Evaluating Formal Specifications:

A Cognitive Approach

Ricky Jay Vinter

Technical Report No: 320

Department of Computer Science
Faculty of Engineering and Information Sciences

July 1998

Acknowledgements

I would like to thank Professor Martin Loomes, the project's principal supervisor, for his general guidance and support, many helpful criticisms and belief in the value of this research.

I would like to thank Doctor Diana Kornbrot, the project's second supervisor, for helping to shape the psychological methodology used in this research, her help with statistical procedures and her many thought provoking discussions.

I would like to thank David Boniface, Principal Lecturer in Statistics, and Ken Ryder, Research Support Statistician, for their help with statistical tools and procedures.

I am indebted to those friends and work colleagues who reviewed the documents produced as a result of this research: Carol Britton, Ben Potter, Richard Ralley and Jane Simpson. I would like to thank the Economic and Social Research Council and the University of Hertfordshire for their provision of funding and resources in support of this research. Thanks are also due to all of the staff, students and software professionals who participated in the project's experiments.

Finally, I would like to thank John and Alice Vinter, to whom this thesis is dedicated, and without whose encouragement, patience and support this programme of research would not have been possible.

RJV

Evaluating Formal Specifications: A Cognitive Approach

Doctor of Philosophy

Ricky Jay Vinter

The University of Hertfordshire, 1998

This thesis explores a new approach for supporting software engineering claims with empirical evidence and investigates whether the human potential for error when reasoning about formal specifications can be reduced. The cognitive science literature has shown that people succumb to various forms of systematic error and bias when reasoning about natural language statements containing logical connectives. A series of empirical studies are reported, which use the framework of the Z notation to test whether people are liable to commit the same errors and biases when reasoning about formal specifications. The results suggest that the ways in which people reason about formal expressions are influenced by grammatical properties of the specification, such as the degree of thematic content or the polarity of logical terms, and by attributes of the reasoner, such as length of experience or degree of expertise. The empirical data is then recast into a tentative predictive model for quantifying how far properties of formal specifications are likely to evoke erroneous inferences. The model may be used as an empirical basis for assessing some of the claims associated with formal methods, or as a basis for engineering formal specifications with less propensity for eliciting erroneous development decisions.

Contents

Chapter 1

A False Sense of Security?

> "It is, indeed, the common fate of human reason to complete its speculative structures as speedily as may be, and only afterwards to enquire whether the foundations are reliable. All sorts of excuses will then be appealed to, in order to reassure us of their solidity, or rather indeed to enable us to dispense altogether with so late and so dangerous an enquiry" (Kant, in Smith, 1993, p.47).

The success of our decisions in everyday life often depends on the accuracy of the reasoning processes which lead to them. We regularly make spontaneous decisions, neglectful or indifferent to the cognitive processes upon which they are based (Evans et al., 1993; Johnson-Laird and Shafir, 1993). Even when inappropriate reasoning strategies are adopted we somehow manage to muddle through using trial and error, and we often receive immediate feedback informing us whether or not our choices are correct. This is not usually the case in the context of software engineering, however, where the criteria against which our reasoning is assessed becomes much more strict and the consequences of our errors more apparent. Most key development decisions tend to be made at an early stage in a software project and developers

do not receive feedback on the accuracy of these decisions until near the project's completion, by which time it may transpire that the initial decisions were incorrect and that additional work is required to rectify the mistakes incurred. The entire history of software engineering has shown us that the cost and effort required to rectify such mistakes tends to increase along with the time taken to recognise them (Cohen, 1989a; McMorran and Powell, 1993; Sheppard and Ince, 1993). It therefore seems critical to the success of software projects that early development decisions are carefully deliberated and founded on sound reasoning wherever possible. Besides incurring the costs and delays associated with correcting erroneous work, incorrect development decisions which go undetected can also lead to the introduction of defects in software systems. Given the increasing complexity and criticality of software (MacKenzie, 1992), this is a genuine reason for concern.

1.1 Project Motivation

Historically, failure to interpret or reason correctly with specifications has caused developers to make incorrect development decisions which, in turn, have led to the introduction of faults or anomalies in software systems (Fenton and Pfleeger, 1996; Potter et al., 1996). Much of the software community argues that such problems stem from designers' near exclusive use of natural language based notations, which are notoriously prone to imprecision (Gehani, 1986; Meyer, 1985) and verbosity (Barroca and McDermid, 1992; Norcliffe and Slater, 1991). This claim is supported by the linguistic literature which reaffirms that natural language descriptions can be vague and ambiguous (Empson, 1965; Turner, 1986). It is also supported by the cognitive literature which suggests that natural language sentences containing certain common forms of logical construct are prone to elicit human reasoning errors and, consequently, to cause erroneous decisions (Braine and O'Brien, 1991; Johnson-

Laird, 1977; Johnson-Laird and Tridgell, 1972; Lakoff, 1971; Newstead et al., 1984).

The past two decades have seen some striking new developments in software technology, including the advent of "formal methods"; languages and tools with precise mathematical underpinnings designed to overcome some of the intrinsic weaknesses of natural language based notations. Advocates of formal methods claim distinct advantages over natural language including increased precision and concision, and increased levels of insight and confidence gained through the use of a mathematical approach (Ince, 1992; Liskov and Berzins, 1986; Wing, 1990). Supposing that formal specifications are indeed more precise than their informal counterparts and that audiences are more likely to interpret them correctly, it follows that people would be more likely to reason correctly about formal specifications, at both a formal and informal level (Thomas, 1995). This possibility is worthy of special scrutiny because improved reasoning could lead to more accurate development decisions which, in turn, could lead to the introduction of fewer defects in software systems. Aside from being overwhelmingly appealing, the possibility that a major source of software defect could be eliminated through formalisation appears, for many, to be plausible.

Reichenbach (1966, p.3) argues "It is true that simple logical operations can be performed without the help of symbolic representation; but the structure of complicated relations cannot be seen without the aid of symbolism". This assumption seems central to the mathematical argument that it is easier to reason about problems expressed in formal logic than those in natural language (Ince, 1992; Lemmon, 1993). The claim that users are more likely to reason correctly about formal specifications than their natural language counterparts is not often made explicitly in favour of formal methods. It appears, nonetheless, to be a popular intuition amongst members of the software community. Providing this intuition is correct and the formalists' argument is proven, the software community stands to make

tremendous gains by adopting formal methods because they could provide a key to the development of safer systems.

Despite clear symbolic differences, most formal notations contain logical operators with roughly equivalent meanings to those same natural language constructs which have been shown to elicit incorrect decisions in cognitive studies. A question that the software engineering community must ask itself is: Are the same non-logical errors and biases that people without formal training exhibit when reasoning about problems in natural language also liable to occur when trained software developers reason about logically equivalent statements in formal specifications? The fact that people currently err at all in the specification process is a reason for concern, but the possibility that they will continue to do so even after having adopted a formalism is especially disconcerting given the business or safety critical nature of the projects to which these are usually applied (Barroca and McDermid, 1992; Bowen and Stavridou, 1993a). If software developers assume that formal methods promote error-free reasoning when in fact this belief is inaccurate, then Farr (1990, p.30) is correct when he argues "reliance on formal methods alone may be dangerous, in that they may engender a false sense of security, in all senses of the phrase".

The software engineering community has acquired a reputation for failing to support the claims relating to its emerging technologies with scientific evidence (Glass, 1996; Loomes et al., 1994). This might be attributed to the difficulties or costs involved in validating such claims. It has meant, however, that potential users have had little choice but to accept or reject emerging technologies at face value. The claims associated with formal methods are no exception. They are invariably based on personal conviction, isolated surveys or case studies from which results can be difficult to generalise (Craigen et al., 1995; Fenton, 1996). Empirical software research has yet to produce substantive evidence which might refute these claims and this appears, for many, to support the case for formalisation (see for example:

Bowen, 1988; Hall, 1990; Potter et al., 1996). Irrespective of how plausible these claims might appear, many of the community's beliefs relating to formal methods rest on anecdotal, rather than empirical, grounds.

The belief that human operators are often responsible for system faults or failures has led designers, most notably in the manufacturing industry, to adopt increasingly sophisticated technologies and to automate existing work practices as far as possible, in the hope that this will minimise the opportunity for human error. The ways in which such technologies are applied, however, can radically transform the traditional role of humans, reshape their cognition and behaviour and, thereby, create a potential for new forms of error. In order to determine how design could give rise to faults or latent system failures, it is important that we explore the possible impacts of technological change on human cognition and behaviour (Woods et al., 1994). Formal methods may be regarded as a new technology in its own right because it promises to solve some of the problems surrounding the software development process and to modify that process irreconcilably (Cohen, 1989a). It is therefore possible that a situation analogous to that in the manufacturing industry is gradually emerging in the software community, whereby the introduction of formal technology is creating many new and unforeseen opportunities for human error.

Human error is divisible into two categories of failure; "slips" tend to result from unmeditated behaviour, when subconscious actions fail to fulfill their intended goal, whereas "mistakes" tend to result from deliberated behaviour, when conscious intentional actions are inadequate to achieve their goal (Reason, 1990). In comparison with slips, mistakes tend to have more serious consequences and are more difficult to detect and rectify (Norman, 1988). Although the potential for both forms of error exists in the formal specification process, in this research we are mainly concerned with systematic mistakes made by many people, rather than one-off slips made by one individual under exceptional circumstances. We are concerned with

those conditions under which an individual intends to draw a correct judgement but the mental strategy upon which his or her reasoning rests is either inadequate or inappropriate, especially where the mistake is attributable to the influence of a heuristic or bias. We focus on erroneous decisions drawn from formal specifications that the decision maker believes to be correct, even after careful deliberation, because these appear to be the root cause for many of the faults or anomalies that are introduced into software systems. Systematic reasoning errors of this nature are particularly problematic because they are also likely to be endorsed by the human checkers involved in manually reviewing or certifying software systems.

1.2 Research Aims

The psychological literature documents a wealth of research aimed at exploring the cognitive processes that contribute to human error in everyday decision making (Norman, 1988; Reason, 1990; Senders and Moray, 1991). The computing literature documents a substantial amount of work aimed at reducing the chances of human error in the software development process (Cohen et al., 1986; Sommerville, 1992; Woods et al., 1994). A driving philosophy behind this research was to bring these two bodies of knowledge together, albeit within the limited domain of formal specification, by applying theories and procedures from cognitive science to the software specification process. In so doing, a cognitive approach to evaluating formal specifications is adopted. The intention here was to raise awareness of some of the often overlooked human difficulties associated with formal specification and to subject some of the claims associated with formal methods to empirical analysis. A central tenet of this thesis is that the software engineering community stands to benefit by taking on board relevant research findings from alternative disciplines in this manner and by supporting its claims with empirical evidence; something it has been

slow to do in the past (Fenton, 1996).

The overall aim of this research is to explore the use of a new approach for supporting software engineering claims with empirical evidence and to investigate whether the human potential for error when reasoning about formal specifications can be reduced. Three subsidiary aims were established to meet this overall aim:

1. Identify through a review of the cognitive science literature those key factors which significantly affect human reasoning performance in natural language contexts.

2. Test through empirical study whether the non-logical errors and biases that reasoners exhibit in natural language contexts are liable to transfer into the formal domain.

3. Formulate a system of metrics for quantifying how far different combinations of these factors are likely to affect human reasoning performance in formalised contexts.

Cognitive science research suggests that human reasoning in natural language contexts can be affected significantly by a range of factors including: the type of inference to be drawn (Evans, 1977a), the degree of realistic problem content (Johnson-Laird et al., 1972), the ordering of logical terms (Begg and Harris, 1982) and the believability of conclusions to be inferred (Evans et al., 1983). The first subsidiary aim seeks to identify those common factors from cognitive studies of natural language based reasoning which could also give rise to non-logical errors and biases in formalised contexts. It focuses in particular on those grammatical constructs and linguistic conditions from natural language that may also occur in formal specifications.

Based on results from cognitive studies of natural language based reasoning, the second aim pursues the main line of inquiry by manipulating combinations of grammatical constructs and linguistic condition in formalised experimental conditions and observing the effects on the reasoning performance of users with various levels of expertise. The intention here is to yield insights into the ways in which psychological processes combine with properties of formal specifications to influence human judgement in software development contexts. The results of these experiments enable us to identify a range of erroneous inferences made by the users of formal methods and the conditions under which they are likely to occur. It should be noted that the second subsidiary aim encompasses the central empirical hypothesis of this research. The possibility that non-logical heuristics and biases might be employed when trained software developers are reasoning about formal specifications is a genuine reason for concern, and one which potentially has far-reaching implications for both the computing and cognitive communities.

Although computing research has proposed numerous models for predicting likely sources of human error in a wide range of software development contexts, the predictions of many are overgeneralised and contentious (Fenton, 1991; Fenton and Kaposi, 1989; Senders and Moray, 1991; Sheppard and Ince, 1989). Our aims are much more modest; we aim only for a tentative model to characterise very specific types of human reasoning under highly specialised conditions. Although we do not expect the predictions of the model to be completely accurate, they might be considered an improvement on crude human intuition for predicting likely sources of human reasoning error and bias in formal specifications. The fact that they are supported by empirical evidence from repeatable experiments should also lend them rather more credence than that given to many of the highly publicised claims relating to formal methods. Although the model built under the third subsidiary research aim is admittedly limited in its scope of application, the way in which

it is formulated demonstrates the feasibility of a methodology that may be used to subject many of the software community's claims to independent and objective scrutiny. The model does not aim to "settle" the massive debate concerning formal methods, but it is hoped that this research raises awareness of some of the key issues as a basis for discussion and provides a framework on sufficiently solid foundations upon which future research endeavours might build.

1.3 Research Methodology

The claims relating to software engineering tools and working methods have historically been based on anecdotal forms of evidence such as personal opinion, surveys or case studies (Fenton, 1996; Loomes et al., 1994). This trend is problematic, however, in that we are limited in what can be generalised from subjective, isolated forms of evidence. The arguments that are valid for a tool or method in one development context may not necessarily apply in others. Unfortunately, software engineering seems to lack the necessary empirical apparatus with which to test its theories. Once a new technology has been proposed, we accept or reject it at face value and it is unclear whether the claims made in its favour are ever really validated, even after applying it. This trend is especially noticeable in the formal methods community (Craigen et al., 1995; Glass, 1996). A possibility which is explored in this thesis is to test how far data can be generated to populate cognitive models designed for software engineering contexts and, at the same time, to test claims relating to formal methods using empirical means. The research methodology is novel in the sense that such an approach has not been attempted beforehand.

The Popperian "underlying pattern of continuous development" (1974; 1992) proceeds by subjecting initial theories to empirical analysis, eliminating errors, then speculating more accurate theories in their place. This approach ensures that the

scope of an investigation becomes more specialised as tests are conducted and theories are refined to become consistent with the emerging results. The methodology used for controlling this research has a "scientific" basis in the Popperian sense. Rather than basing our theories on subjective belief, we use a link to well supported theoretical knowledge from the domain of cognitive science. In particular, we pursue lines of inquiry stemming from natural language based studies of human reasoning. Rather than using isolated case studies, we also borrow standard experimental procedures from cognitive science in order to subject our theories to empirical scrutiny. This approach is advantageous over conventional software research methodology in that it generates specialised theories "grounded" in the observed data (Glaser and Strauss, 1968), which can subsequently be used to refine hypotheses and generate new theories in a fashion analogous to Popperian scientific methodology. A rationalisation of the actual research methodology follows.

1. A review of the cognitive literature was used to identify particular grammatical constructs and a range of linguistic conditions from natural language which could also give rise to non-logical heuristics in formalised contexts. This gave rise to a general theory: "In addition to attributes of the reasoner, such as expertise and experience, the ease with which a user can draw an inference from a formal specification is influenced by linguistic properties of the specification and the type of inference to be drawn".

2. An exploratory study was used to identify factors which might evoke non-logical reasoning in formalised contexts.

3. Based on the results of the exploratory study, the initial theory was refined into a well founded set of hypotheses based on existing bodies of knowledge; namely, cognitive theories of error and bias in natural language contexts. Although this opened up several lines of inquiry relating to a range of logical operators,

this research focussed only on a subset of those for which cognitive science had produced evidence of systematic human fallibility.

4. Three main experiments were conducted in parallel, each of which tested users' abilities to reason about formal specifications containing different combinations of those properties hypothesised to correlate with reasoning performance.

5. The three lines of inquiry were brought back together as the results of the main experiments were summarised in the form of a descriptive model. This model forms the basis for the finalised, if tentative, theories of this research with regard to the likely rates at which the users of formal methods are liable to commit non-logical reasoning errors. It also provides an empirical basis for testing some of the claims associated with formal methods because, unlike the anecdotal evidence obtained from case studies, one of the main advantages of an empirical model is that it can be evaluated, modified or extended by others in a scientific way.

1.3.1 Selecting a Formal Notation

In order to conduct the experiments and formulate metrics it was necessary to use a notation with a single, concrete syntax and a sufficiently large user base. The range of formal notations developed for the purpose of software specification has been steadily increasing during the past decade, however, and the range of notations now available vary widely in their mathematical foundations, purpose and popularity. A systematic review of twenty formal notations (Vinter, 1996) was conducted against pre-determined criteria in order to identify a suitable grammatical framework for meeting the research aims. The reasons for choosing the Z notation (Spivey, 1992) were as follows:

1. The fact that Z is one of the most popular notations used in academia and industry (Parkin and Austin, 1994) eased the task of finding adequate numbers of suitably skilled participants for the experiments.

2. The mathematical calculi underlying the Z notation, predicate logic (Lemmon, 1993) and set theory (Johnstone, 1987), provide the grammatical basis for many other formal notations. The research findings might therefore generalise to other notations sharing the same logical foundations: Gypsy (Ambler, 1977), Larch (Guttag et al., 1985), RAISE (RAISE Language Group, 1992) and VDM (Jones, 1989).

3. Z is reputed to be one of the more readable formal notations (Bowen, 1988; Jack, 1992). It is expected that the measures yielded by our model, for quantifying the likely levels of correctness for Z users, will therefore reflect the results of a "best case" analysis.

4. Having been applied in industry for developing a diverse range of applications, Z is one of the more well established and commercially viable formal methods (Barden et al., 1992). This has favourable implications for the credibility of the model's theoretical basis and suggests that the model may be genuinely useful after further validation work has been conducted.

5. Perhaps owing to the current drive for a rigorous deductive proof system and an international standard for Z (Brien and Nicholls, 1992), not to mention its increasing acceptance in industry (Hall, 1996; Nix and Collins, 1988; Nicholls, 1987), the Z notation represents a highly active area of academic research and one in which much still remains to be explored.

1.4 Presentation of the Research Findings

This research is truly interdisciplinary insofar as it employs theories and procedures from the domains of psychology, computing, linguistics and statistics. In view of the fact that it addresses problems in the software engineering domain via the application of theoretical knowledge and standard procedures from cognitive science, however, this research has been written up in the style of a joint computing and cognitive science project. Every attempt has been made to make the thesis accessible to members of both communities by developing the main intellectual topics of concern from first principles and relating them to both bodies of literature.

This chapter has set the scene for the discussions to follow by describing the motivation for the research, its aims and methodology. It has outlined the hypothesis that, despite the much publicised claims associated with formal methods, reasoning with specifications will always be susceptible to the fallibility of human judgement. This seminal hypothesis will be revisited on many occasions as we test its empirical credibility for a range of variables hypothesised to influence human reasoning in formalised contexts and we explore the implications of the results.

Chapter Two describes some relevant principles of software measurement and outlines how a system of metrics, based on empirical data, could be used to mitigate against some common causes of error in the software development process. This leads onto a description of the role of specification in the development process and a brief introduction to formal methods. We consider why much of the software community now appears to be seriously contemplating the use of formal methods, and review some of the major claims made in their favour. We then explore the central hypothesis of this thesis and speculate as to why the users of formal methods may be liable to err in ways similar to those exhibited by people without formal training in natural language based contexts.

Chapter Three considers the research aims from a cognitive perspective and introduces some relevant principles from cognitive science. It begins by discussing the role of inference in human reasoning and that of deduction in formal specification. This is followed by a discussion of the possible relations that may exist between mathematical logic and human deductive competence. The chapter concludes by speculating on some likely conditions under which software developers are likely to err when reasoning with formal specifications. The speculations give way to empirical hypotheses based on findings from cognitive studies of logical reasoning in natural language contexts. These hypotheses form a theoretical basis for our formalised studies.

Chapter Four describes how an initial study helped, first, to identify key properties of formal specifications that are particularly likely to influence human reasoning performance, second, to test some popular software engineering claims pertaining to formal methods and, third, to refine the methodology used for this programme of research. It concludes with a description of how the findings from this initial study gave rise to three more specialised studies of human reasoning in formalised contexts.

Chapters Five, Six and Seven report the three main formalised experiments, each of which focus on the ability of individuals to reason about logical properties of the Z notation under different grammatical conditions. The aim of these experiments is to identify specific combinations of formal construct and linguistic condition which are particularly likely to evoke error and bias when users are reasoning about formal specifications in software engineering contexts.

In Chapter Eight the results of the three main experiments are synthesised into a system of metrics. Although tentative and limited in its scope of application, the system may be used, first, as a means for identifying properties of formal specifications which are likely to cause human reasoning errors and, second, to

subject some of the anecdotal claims associated with formal methods to empirical scrutiny. A brief demonstration of the model is followed by a discussion of how far the methods used in its construction satisfy generally accepted software measurement validation criteria.

Chapter Nine begins with a description of how far the research aims were met and reflects on the research methodology used. This is followed by discussion of the research results and their implications for the software engineering and cognitive science communities. During this discussion a number of possible directions for future research are proposed. Finally, some concluding remarks summarise the main findings from the programme of investigation.

The appendices contain material supplemental to the main substance of the thesis. Appendix A contains the task sheets distributed to participants during the project's experiments. Appendix B describes the documents that have been published as a result of this research.

Chapter 2

An Empirical Basis for Assessment

> "It would appear the claims of formal methods advocates are primarily based on bias, belief and anecdote. Mind you, formal methods advocates are not alone in making such claims; most software engineering claims are based on dubious scientific grounds. This is not to say that the claims are necessarily false, only that they are scientifically unproven" (Craigen et al., 1995, p.407).

This chapter introduces a theoretical basis for a measure designed to predict the likelihood that a reasoner will draw particular inferences under given conditions. It describes how a system of metrics based on this notion could be applied to formal specifications in order that potential sources of interpretative or reasoning error might be identified and reduced before they have chance to elicit erroneous decisions. It also describes how such a system might be used by the software engineering community to support or refute some of its "psychological claims" relating to formal methods with empirical evidence.

2.1 Principles of Software Measurement

The term "measure", in the software measurement literature, refers to a number or symbol assigned to characterise an attribute of an entity (Fenton and Pfleeger, 1996; Kaposi and Myers, 1994; Kitchenham, 1991). Aside from its unfortunate connotations with decimalisation and mathematical functions, the term "metric" has, in the context of software engineering, become synonymous with "measure". Although the term "metric" might suggest both a measure and an underlying model or theory, this is not a view that reflects the ways in which metrics are normally perceived or applied (Sheppard and Ince, 1993). It is for this reason that no such distinction is maintained in this thesis, where the terms "measure" and "metric" are used interchangeably.

Numbers may be used to characterise the quality of both products and processes, and there exist two corresponding divisions of software metric (Fenton and Pfleeger, 1996; Ince, 1989; Roche, 1994). "Product metrics" are oriented towards quantifying the tangible outputs from development activities such as requirements documents, system design or program code. Typical examples include the number of executable lines of code and the percentage of comments per program module. "Process metrics" are oriented towards quantifying the development processes themselves, such as program design, implementation and testing. Typical examples include the predicted costs of development and the number of defects found during a phase of program testing.

Software metrics can be applied as both descriptors and predictors of quality, which respectively mirror the reactive and proactive ways in which they can be used (Ince, 1989; Kaposi, 1991; Kitchenham, 1991). "Descriptor metrics" are used to describe existing products or processes. Typical examples include product measures such as the number of possible routes that may be taken through a program module,

and process measures such as the total expenditure on testing. "Predictor metrics" are used to estimate final characteristics of products yet to be developed or to produce estimates of descriptor metrics. Typical examples include product metrics such as the estimated number of lines of code that will comprise the final system, and process metrics such as the projected costs of an entire project based on the amount spent so far and the work yet to be completed.

Software metrics can again be further categorised according to whether the attribute to be measured is internal or external (Fenton and Pfleeger, 1996). The "internal" attributes of a product or process are those which can be measured purely in terms of the product or process itself. The length of source code, for example, can be measured directly without reference to its behaviour. The "external" attributes of a product or process are those which are measured in terms of how it interacts with its environment. The reliability of a system, for example, can be measured in terms of the number of times it fails to fulfill a valid service requested by its users. It is a common misconception that the numeric value yielded by an external metric provides a definitive and all encompassing account of the quality of a product or process; the higher the value, the more quality is assumed to exist. External metrics tend to be defined in terms of only a small subset of internal attributes, which are weighted and combined in just one of numerous possible ways.

The metrics to be developed during the course of this thesis may be classified as predictive, external and product based. They are predictive because they relate to possible future events; namely, the types of conclusion likely to be drawn by human reasoners in response to formal specifications. They are external because their values are not calculable directly from formal specifications alone, but also from the ways in which people have reasoned about similar specifications in the past. The metrics are product based because they yield measures oriented towards grammatical properties of specifications such as the presence of specific logical operators, the degree of

realistic and believable content, and the types of logical statement that may be inferred from combinations of these. Whilst it may be helpful to think of the system of metrics in this manner, the ways in which measures may be classified under such a taxonomy are not always clear-cut.

2.2 Benefits of Early Software Measurement

Reliance upon source code metrics alone, such as those proposed by Halstead (1977) and McCabe (1976), has proven unsatisfactory for identifying software development problems because these can only be applied at a late stage in the development process, once code has been implemented. By this time, the effort and expense of producing a system has already been incurred and it can be a costly exercise to backtrack through specification, design, coding and testing to rectify even the slightest mistake or omission (Sheppard and Ince, 1993). It would seem more beneficial to apply metrics earlier in the development process, where they are likely to help realise greater savings (Bainbridge et al., 1991; Fenton and Kaposi, 1989; Sheppard, 1988). This view is supported by Curtis' (1979, p.103) argument that "it will be more valuable to estimate metric values earlier in the life cycle in order to acquire a better understanding of how various design decisions will affect resulting system characteristics".

To complement existing reactive forms of software quality metric, software engineering requires a more proactive form of measurement that enables developers to predict likely sources of development error before the system has been built. Earlier measures could provide feedback to software developers and act as a basis for making more informed development decisions before errors have had chance to propagate through to subsequent development work. Predictive measures aimed at specifications could be especially useful in this respect because specifications tend

to cause the most expensive and difficult development problems (Sheppard, 1990). Wordsworth (1992, p.68) argues "in a formally developed product the aim is defect prevention rather than defect detection", and it is indeed in the kinds of critical engineering context where formal methods are applied that predictive measures could help to prevent some of the potentially more harmful software defects.

2.3 Measuring the Complexity of Specifications

The software measurement literature informs us that complexity metrics are useful indicators of other software quality attributes (Kitchenham, 1991; Sullivan, 1975). For example, complexity is generally related to reliability because complicated documents are more likely to contain residual errors, and to maintainability because complicated documents tend to require more time and effort to understand before incorrect sections can be located and revised. There is a tendency in the software industry, however, to perceive complexity metrics as precise indicators of the complete range of attributes that contribute towards the quality of a product or process, such as readability, structural simplicity, maintainability and robustness, when in fact they are based on measures taken from only a narrow subset of such attributes (Fenton, 1992). McCabe's (1976) model, for example, claims to provide indications of cognitive complexity, program defects and maintenance costs. Given that its predictions are based only on the number of logical decision branches in program code, however, it is debatable as to whether or not these notions are actually within its descriptive power (Fenton, 1991; Sheppard, 1988; Whitty, 1997).

Formal methods research has historically been directed primarily towards developing or improving tools and notations, that is, it has been oriented towards the supporting technology rather than its human users. But assuming that the human potential for error will remain after formalisation of the specification process and

that inaccurate human reasoning about formal specifications will continue to lead to the introduction of defects in software systems, it seems appropriate that we strive to predict errors in human judgement before these have chance to cause erroneous behaviour. This view is supported by Senders and Moray's (1991, p.66) argument that "we need to know the probability of the mistaken decision even more than the probability of the 'incorrect' actions stemming from it". One way of achieving this is by introducing a model for predicting likely sources of psychological complexity in formal specifications so that likely sources of erroneous human decisions can be avoided. Unlike those models proposed by the software community which claim to yield generalisable measures of complexity, our aims are much more modest. We aim only for a tentative model to characterise very specific types of human reasoning under highly specialised conditions. The intention is not to formulate a general model with a wide scope for application, but rather to demonstrate the feasibility of the approach. Although considerable work will be required to develop the model to a genuinely useful "tool", the potential benefits of such a tool are great.

In view of their central role in project documentation and communication, it is important that we aim to eliminate impediments to the readability of formal specifications. Complexity metrics can be used in support of this aim to highlight those parts of specifications which could give rise to human comprehension or reasoning difficulties. The model which we are working towards is concerned with quantifying "inferential complexity", which might be viewed as a form of psychological complexity aimed at quantifying the ability of people to reason about given statements. This research is concerned with statements taken from formal specifications. It is not claimed that the model is indicative of other attributes belonging to formal specifications, such as maintainability or reliability, although the model does incorporate the notions of interpretability and representability. This claim is based on the assumption that, in order to reason about any form of written text, a

reader must interpret its meaning and maintain some internal representation of it. The proposed model is therefore psychological in nature.

2.3.1 Psychological Versus Computational Complexity

A study of software complexity might be approached from one of two possibly interrelated directions: computational or psychological (Curtis, 1986; Melton et al., 1990; Öry, 1993). A model of the computational complexity in, for example, source code might aim to quantify the algorithms embedded in its mathematical calculations or the structure of its program modules. The measures from such a model could be used to improve the run time efficiency of a software system. A model of psychological complexity might aim to quantify the extent to which attributes of, say, a software specification could cause difficulties in its creation or usage. These attributes might include: the symbology of the notation, the meanings of its constructs or the style of writing employed by its designer. The measures from such a model could be used for constructing a specification that is easier for people to work with. A distinguishing feature of psychological complexity is the interaction between product characteristics and individual differences, such as grammatical constructs and language expertise. Unlike computational complexity, however, modelling psychological complexity requires much more than measures of internal product attributes. Models of psychological complexity must accept as parameters measures relating to both the product and the people who interact with it.

> "Assessing the psychological complexity of software appears to require more than a simple count of operators, operands, and basic control paths. If the ability of complexity metrics to predict programmer performance is to be improved, then metrics must also incorporate measures of phenomena related by psychological principles to the memory, information processing, and problem solving capacities of programmers" (Curtis et al., 1979, p.103).

The psychological complexity of software products and processes has generally been overlooked in favour of quantifying the computational attributes of the development process. All of the specification metrics published to date are calculable purely in terms of explicit grammatical properties of specifications, such as dependencies between modules or decision branches, and do not account for the human developers involved in the specification process (see for example: Bainbridge et al., 1991; Kaposi and Myers, 1994; Samson et al., 1987). Despite claims to the contrary, the predictions of such models are limited from a psychological perspective. It is perhaps due to the increasing complexity and criticality of software generally (MacKenzie, 1992) that the emphasis is changing. In addition to algorithmic efficiency, the software measurement community is becoming increasingly concerned with factors which impair human performance in engineering contexts (Fenton, 1991) and, in particular, sources of psychological complexity in formal specifications (see for example: Finney, 1996).

2.3.2 Statistical Prediction of Human Error

It is usually a complex task to predict errors of human judgement with a fair degree of accuracy because the reasoning processes which influence human judgement involve numerous psychological issues few of which are fully understood, even by professional psychologists. These include the ways in which reasoning is affected by: personality traits, prior beliefs, motivational states, the structure of a problem and its presentation. Moreover, errors of judgement are frequently ascribable to multiple causes and it is rare indeed that all of these causes are evident, even to their perpetrators. This complicates the task of predicting when and where similar errors might happen in future.

"Errors result from the normal operation of the human information processing system, along with effects arising from the environment, the various pressures and biases influencing the actor, and the latter's mental, emotional, and attentional states. [...] In principle, if we knew all these factors, we could predict errors precisely. In practice, since we cannot know all the factors, we will always have to resort to statistical prediction" (adapted from Senders and Moray, 1991, p.61).

Faced with the problem of predicting future events, such as manifestations of human error, our predictions will, where possible, be based on full knowledge of the causal factors which lead to the event occurring. Such knowledge is unlikely to be available or accessible, however, for predicting the psychological causes of error in human judgement. In the absence of such knowledge, our predictions might be based on the frequencies with which the same errors have occurred for similar people in the past. A practical alternative available to us is to make predictions based on probability. There is likely to be an element of uncertainty associated with our predictions in any case given that we can never predict with absolute certainty what will happen in future possible worlds (Springer et al., 1966). Our predictions are therefore "founded upon the assumption that the future will bear a resemblance to the past; that under the same circumstances the same event will tend to recur with a definite numerical frequency" (Aristotle, in Ross, 1949, p.244).

The intuitive heuristic methods which people use to assess the probabilities of future events are notoriously inadequate for making consistent and reliable predictions (Kahneman et al., 1991). Human predictions, including those made by professionals with the relevant training or domain knowledge, are liable to focus on irrelevant factors or neglect relevant ones (Evans et al., 1993), to assign incorrect or inconsistent weightings to factors (Nisbett and Ross, 1980), or to be affected by emotional beliefs and motivational states (Dawes, 1971). It is for these reasons that

we cannot rely solely on humans to predict human errors and, more specifically, that we cannot rely upon human developers alone to pre-empt the sources of their own, and their colleagues', reasoning errors. One possible solution explored in this thesis is to support developers' intuitions with a mathematical means for assessing the potential of specifications for admitting reasoning errors, which is independent from the subjective opinion of those people performing the reasoning.

Although the psychological determinants of error are often less than clear, this does not necessarily mean that explicit mathematical models cannot be used to predict the human propensity for error. This view is supported by Meehl's (1973) argument that the complexity of the human mind should not preclude the use of mathematics in decision making. It is claimed, moreover, that statistical models, particularly regression based models, yield more accurate predictions than the naive intuitions of so-called "experts" with relevant training (Dawes, 1971; 1979; Goldberg, 1970; Slovic and Lichtenstein, 1971).

> "Human judges are not merely worse than optimal regression equations; they are worse than almost any regression equation. Even if the weights in the equation are arbitrary, as long as they are nonzero, positive, and linear, the equation generally will outperform human judges" (Nisbett and Ross, 1980, p.141).

This claim is based on the assumption that statistical methods abstract away from the processes they represent and are not influenced by the kinds of cognitive state or heuristic process which affect human prediction such as: aptitude, fatigue, bias, motivation and habit. Regression based models consistently attach the same fixed weightings to causal factors, regardless of such extraneous variables, and can generate quantifiably precise estimates of the extent to which each contributes towards an event's occurrence (Garnham and Oakhill, 1994). People are notoriously reluctant to allow automated procedures such as these to replace their judgement;

a problem which is pronounced by their overconfidence and frequent tendency to make erroneous decisions based on intuitive guesswork. Although the use of statistical models are claimed to have a "dehumanising" effect, human and statistical predictions share two important commonalities: they are both based on induction because they generalise from past occurrences, and they are both fallible because they rest on probabilities, whether these be implicitly or explicitly defined. Whilst there may be some opposition to the use of statistical models, however, it should be remembered that predictions based on statistical formulae generally agree with those given by human judges. The intention in this thesis is not to replace human judgement, but to complement it with a more objective means of assessment.

2.4 The Role of Software Specification

A software specification is an abstract description which seeks to delineate the software components that a desired system will eventually comprise, and is often used as a basis for evaluating the correctness of finished systems. A specification is normally employed at various stages in a project including: contract negotiation, planning, design or code derivation, and program maintenance. Owing to the nature of these activities, it is likely that the specification will need to be understood by different audiences, each with varying degrees of expertise. A specification document is typically produced as a joint venture between developer and customer, between whom it represents a form of contractual agreement (Cohen, 1989a; Imperato, 1991), but is intended to be read primarily by members of the development team for whom it also serves as a constant source of reference during the system's construction and maintenance. It is generally accepted that the specification process is an essential part of the development life cycle and the potential benefits that stem from a well written specification are widely documented in the computing literature (Cohen et

al., 1986; Potter et al., 1996; Sommerville, 1992).

Although certain forms may be parsed, verified or animated by machines, software specifications are written predominantly for human audiences. For the members of a development team, a system specification represents an account of the software operations that must be implemented in order to meet a customer's requirements. Experience has shown that the errors which developers make when interpreting or reasoning about specifications are liable to manifest themselves in, and propagate throughout, subsequent design and code work, leading to the appearance of faults or anomalies in the system developed (Fenton and Pfleeger, 1996). Serious defects often go undetected until integration or testing, by which time the costs of backtracking through design, code and specification to rectify them have increased dramatically (Cohen, 1989a; McMorran and Powell, 1993; Sheppard and Ince, 1993). The costs of failing to discover the defects could be much higher, however, particularly where the system under development is business or safety critical in nature.

It is clearly important that a specification captures a customer's requirements in a full and comprehensive manner. In view of their central role in project communication (Barroca and McDermid, 1992; Imperato, 1991), it seems equally important that they are written in ways which are easily accessible to their intended audiences with minimal potential for admitting erroneous human reasoning. The early stages of software projects are often critical to their success and software specifications have caused some of the most costly and intractable development problems (Cohen, 1989a; Sheppard, 1990). It would therefore seem reasonable to expect that expending additional effort at the early specification stage, in particular, would substantially reduce development effort overall (Hall, 1990; Potter et al., 1996). But perhaps more importantly, it could help to reduce the numbers of latent defects that find their way into "finished" software systems.

2.5 Formal Methods in Perspective

A formal specification is a description of a proposed system written in a notation with "an explicitly and precisely defined syntax and semantics" (Liskov and Berzins, 1986, p.4). Most formal methods comprise two main components: a formal notation and a deductive apparatus. The formal notation, comprising grammatical symbols and rules, enables designers to build abstract models of desired systems. Formal notations are often underpinned by a formal semantics which define in precise mathematical terms the meaning of syntactically valid statements written in them. The deductive apparatus, comprising axioms and rules of inference, helps users to reach valid deductions about properties of the system without reference to any specific interpretation of the meaning of the formulae being manipulated (Woodcock and Loomes, 1988). This apparatus may be used to test whether users' intuitive theories about a system are correct, or to determine whether selected properties of the system model are consistent with a customer's requirements.

2.5.1 Overcoming the Doubts of Industry

Although formal methods have been available for over thirty years, industry has been reluctant to adopt them because of doubts surrounding their commercial viability. The increasing interest now being shown in formal methods (Bowen and Hinchey, 1994; Oakley, 1990), however, suggests that some of the initial scepticism surrounding their use has been overcome and they are gradually gaining industrial acceptance (for a review see: Bowen and Stavridou, 1993a; Clarke and Wing, 1996). This can perhaps be attributed to the increasing complexity and criticality of software generally, and the growing need for assurance that systems will not fail at the most inopportune times, which cannot be given by existing methods (Craigen et al., 1995; Norcliffe and Slater, 1991). That the software industry now appears to be

contemplating seriously the switch to a formal approach may be ascribed to several notable developments in the formal methods community during the past decade.

- *Improved tool support.* The lack of supporting tools, or deficiencies in existing tools, has discouraged many potential users (Cooke, 1992; Holloway and Butler, 1996). Although much of the technology supporting formal methods is still only experimental or is aimed at highly trained users, proponents argue that it has now evolved to a stage where it can fulfill a useful role in industry (Cohen, 1989a; Sheppard, 1995; Thomas, 1993).

- *Industrial scalability.* The question of whether formal methods are capable of meeting the demands of large scale industrial projects has been a source of contention for many years (Holloway and Butler, 1996; Garlan, 1996). It is argued, however, that the increasing application of formal methods in industry is helping to resolve the debate in their favour (Hall, 1990; Nix and Collins, 1988; Thomas, 1993).

- *Supporting standards.* For many, the lack of standards supporting the use of formal methods gave the impression of a technology that was immature and unsettled. Independent regulatory bodies, however, have recently begun introducing generic standards mandating the use of formal methods on certain types of project, and have also begun standardising the semantics underlying formal notations themselves (for a review see: Bowen and Stavridou, 1993a).

- *Changes to traditional work practices.* Formal methods are widely perceived as a complex technology both to learn and apply (Cohen, 1989a; Craigen et al., 1995; Potter et al., 1996). Academia now teaches formal methods theory, tools and notations as part of standard computing curricula, thereby ensuring that potential users are equipped with relevant skills before entering industry.

- *Financial viability.* One of the main reasons why industry has been reluctant to embrace formal methods is because of doubts surrounding their cost effectiveness. Recent surveys, however, suggest that the cost of adopting a formal approach is becoming comparable with that of adopting conventional development technologies (Craigen et al., 1995; Larsen et al., 1996).

Historically, software engineering has lacked the kind of theoretical foundations underlying other engineering disciplines, such as mechanics, whereby scientific theories can be represented via abstract models and subjected to mathematical analysis (Cohen, 1989a; Nicholls, 1987; Norcliffe and Slater, 1991). It is argued that programs are analogous to scientific theories (Hoare, 1986; Loomes, 1991; Naur, 1985a), however, which can be subjected to independent and objective mathematical testing then accepted or refuted, leading to refined programs. A similar scientific methodology appears to accompany formal specification. Perhaps the main reason why formal methods have recently attracted so much interest, therefore, is that they appear to add a scientific basis to a previously unscientific practice. Indeed, for many, formal methods tend to evoke notions of carefully deliberated technological progress and have close associations with scientific culture (Cohen et al., 1986; Hall, 1990; Hoare, 1984).

2.5.2 The Claims for Formalisation

Proponents have been keen to point to numerous possible advantages of formal methods over conventional, informal methods of specification. In this research we discriminate between those claims which rest directly upon psychological assumptions and those which do not. We concentrate in particular on those claims whose credibility rest upon the use of formal specification as a medium for human communication or as an instrument for supporting human reasoning. This excludes several

notable claims made in favour of formal methods including those relating to: development costs (Thomas, 1993), test case generation (Clarke and Wing, 1996), mathematical verification (Sommerville, 1992), machine animation (Reade and Froome, 1990), automated tool support (Wing, 1990), component reuse (Bowen and Hinchey, 1995) and support for abstraction (Hinchey and Bowen, 1995). Five specific claims which do fall within our scope of concern are discussed as follows.

- *Increased understanding through translation.* The process of writing a formal specification forces a designer to translate a set of informal requirements into mathematical terms. Besides helping to uncover latent errors, ambiguities or incompleteness in the system model (Clarke and Wing, 1996; Meyer, 1985; Thomas, 1993), it is argued that the process of formalising a customer's requirements heightens a designer's understanding of the problem at a relatively early stage in a system's development (Bowen, 1988; Hall, 1990; Wing, 1990), and is often sufficient reason alone to justify a formal approach, even where there is no intention to verify formally the model produced (Cooke, 1992).

- *Increased conciseness.* Based on the view that it usually requires several more cumbersome natural language statements to express the same meaning conveyed by a single mathematical expression, it is argued that formal specifications are more concise than their natural language based counterparts (Barroca and McDermid, 1992; Jacky, 1997; Meyer, 1985). Conciseness can be particularly beneficial because it simplifies the tasks of checking for residual errors, correcting mistakes and referencing required information.

- *Unambiguous specifications.* One of the main problems historically associated with the specification process has been one of communication. Ambiguous specifications are liable to be understood in different ways by different people, with the danger that not every developer will work towards achieving

the same system solution. This problem has generally been attributed to the use of natural language based specifications, which are notoriously prone to imprecision (Gehani, 1986; Imperato, 1991; Meyer, 1985). It is argued that formal methods alleviate these communication problems by giving rise to precise and unambiguous specifications (Imperato, 1991; Jack, 1992). Based on the assumption that the semantics underlying a formal notation give to every statement expressed in that notation a precise mathematical meaning, it is often argued that formal specifications are open to only one form of interpretation (Bowen, 1988; Liskov and Berzins, 1986; Thomas, 1993).

- *Easier error location.* Given the propensity of natural language to verbosity and ambiguity, it is sometimes difficult to determine exactly what is being said in an informal specification, let alone which parts are erroneous (Hall, 1990). Based on the assumption that formal specifications are generally more precise and concise than their informal counterparts, it is often argued that errors are more easily found, which leads to fewer defects in finished systems (Bowen, 1988; Macdonald, 1991; Oakley, 1990).

- *Support for human reasoning.* Human reasoning can either be formal, and based on well defined mathematical laws with explicitly recorded intermediate steps, or informal, and based on belief or intuition with implicit or undefined steps (Jacky, 1997). Whereas formal reasoning abstracts from the content of arguments, informal reasoning tends to take into account both content and context in drawing conclusions (Salmon, 1991). Formal reasoning might involve checking via mathematical proof that relationships between a specification and its implementation hold, whereas informal reasoning might involve checking via subjective judgement that a system model corresponds to a customers' requirements. Based on the assumption that formal specifications are

more concise and less ambiguous than their natural language counterparts, it is argued that formal methods encourage disciplined reasoning (Rushby, 1995) and that it is easier to reason about formal specifications, even at an informal level (Thomas, 1995). It is argued, moreover, that "due to limitations of the human brain, many insights are not possible without adequate formal notation" (Kneuper, 1997, p.383).

2.5.3 The Potential for Human Fallibility

In the context of software engineering, the term "method" suggests an ordered, prescriptive set of procedures which can be followed to guide the development of software, such as stepwise refinement or object oriented design (Cooke, 1992; Hinchey and Bowen, 1995). In the context of software specification, the term "formal method" seems to suggest a precisely defined set of procedures for developing or manipulating specifications. Yet the processes of writing, refining or verifying a formal specification rarely comprise any predefined systematic steps of action whatsoever. It is in fact only the notations in which formal specifications are written that are formally defined, owing to their precisely defined grammatical foundations (Woodcock and Loomes, 1988). The development processes associated with formal methods are guided mainly by the spontaneous judgement and ingenuity of individual developers (Bowen and Hinchey, 1995; Jacky, 1989; 1997; Oakley, 1990), and their lack of support for developmental methodology in this sense has led much of the software community to regard the term itself as somewhat misleading (Barden and Stepney, 1993; Bowen and Hinchey, 1994; Jacky, 1997; Woodcock and Loomes, 1988). The term "formal method" is used in this thesis to refer to a formal notation, and the set of formal, or informal, processes associated with its use.

> "So 'Formal Methods' (only) provides a framework in which programs can be developed in a justifiable way. It does not dictate, or even advise, on how manipulations should be applied. There is still a need for the program developer to make decisions and to determine appropriate programming strategies" (Cooke, 1992, p.420).

Formal methods provide developers with a selection of tools that can be used to create, and possibly verify, specifications. But like any other form of tool, it is possible for formal methods to be misused (Bowen and Hinchey, 1995). Software engineering has always been driven predominantly by human judgement and no conceivable developments in the formal methods community are likely to change this. So despite their mathematical foundations and much reputed scientific basis (Cohen et al., 1986; Hall, 1990), errors will continue to arise in, and from, formal specifications simply because of the natural fallibility of their human users (Bowen and Stavridou, 1993a; Hinchey and Bowen, 1995). Although supporting tools, guidelines and standards might help to reduce the numbers of errors committed, they cannot prevent errors from arising completely (Cohen, 1989b). In this light, Hoare's (1984) vision of future software engineering practice, in which mathematical techniques are used to guarantee that specifications can no longer give rise to software defects and conventional testing methods are discarded in favour of formal reasoning, now seems unrealistic and unachievable (Loomes, 1991). The possibility that incorrect development decisions will continue to emanate from specifications, however, is disconcerting in view of the increasingly critical engineering questions being asked of the software community (MacKensie, 1992) and the high degrees of confidence that tend to be placed in systems developed using formal methods (Wing, 1990).

"Formal methods cannot guarantee correctness; they are applied by humans, who are obviously error prone. Support tools - such as specification editors, type checkers, consistency checkers, and proof checkers - might reduce the likelihood of human error but will not eliminate it. Systems development is a human activity and will always be prone to human whim, indecision, the ambiguity of natural language, and simple carelessness" (Bowen and Hinchey, 1995, p.60).

Formal methods research has historically focussed on developing new notations and supporting tools. This research has mostly been conducted in purely academic environments without a full understanding of development problems experienced by industry (Garlan, 1996). Aside from resulting in a proliferation of tools that are awkward to use, unreliable and unsuited to the types of problem faced by industrial practitioners (Holloway and Butler, 1996), this near exclusive preoccupation with formal methods' supporting technology has distracted computing research from one of the most important agents in the design process; the human users of formal methods themselves. It seems appropriate that computing research should at least pause to consider the cognitive implications of using this technology. Empirical research has, in particular, failed to question the extent to which the human potential for error will remain after formalising the specification process.

2.5.4 Subjecting the Claims to Empirical Assessment

The increasing interest in formal methods being shown by the software community (Bowen and Hinchey, 1994; Oakley, 1990) may be attributable to the highly publicised claims that the use of formal methods will lead to greater benefits than are realised with informal methods. Many of these claims, however, are based on subjective belief or isolated case studies from which results can be difficult to generalise (Fenton, 1996).

"One difficulty we encountered in determining the relative advantage of formal methods is the lack of strong scientific evidence that the technology is, in fact, effective. Various surveys have provided reasonably systematic anecdotal evidence of effective industrial use of formal methods. However, none of the formal methods application projects has used strict, scientifically based, measurement data" (Craigen et al., 1995, p.407).

That software research has yet to produce substantive evidence which might refute the formalists' claims appears, for many, to support the case for formalisation (see for example: Bowen and Hinchey, 1994; Hall, 1990; Potter et al., 1996). But irrespective of how plausible they might appear at face value, such claims rest on anecdotal, rather than empirical, grounds.

"In the absence of a suitable measurement system, there is no chance of validating the claims of the formal methods community that their models and theories enhance the quality of software products and improve the cost-effectiveness of software processes" (Fenton and Kaposi, 1989, p.293).

This thesis reports a series of empirical studies aimed at identifying properties of formal specifications which are particularly likely to elicit human reasoning errors and biases. The results of these studies are subsequently recast in terms of a descriptive model designed to yield quantitative measures of inferential complexity in formal specifications. The model provides an empirical basis for assessing two of the psychological claims associated with formal methods upon which a great deal of the formalists' case rests; namely, that formal specifications are open to only one form of interpretation (Bowen, 1988; Liskov and Berzins, 1986; Thomas, 1993), and that it is easier to reason about formal expressions than their informal counterparts (Ince, 1992; Thomas, 1995; Wing, 1990).

2.6 Summary

This chapter has argued that one of the primary roles of software specification is as a medium for supporting human reasoning and that the ways in which specifications are written must take into account the fallibility of human judgement. It has introduced the notion of inferential complexity and outlined how a system of metrics based on this notion might be used, first, to subject some of the claims associated with formal methods to empirical scrutiny and, second, to identify potential sources of human error in the formal specification process. Finally, it has pointed to evidence which suggests that, despite the use of formal methods, the human potential for reasoning error and bias in the specification process is liable to remain, and has indicated why such errors and biases are a special reason for concern in software development contexts.

Chapter 3

Searching for the Logic in Human Reasoning

> "To err is human. No matter how well we come to understand the psychological antecedents of error or how sophisticated are the cognitive [...] devices to aid memory or decision making we eventually provide for those in high-risk operations, errors will still occur. Errors are [...] the inevitable and usually acceptable price human beings have to pay for their remarkable ability to cope with very difficult informational tasks quickly and, more often than not, effectively" (adapted from Reason, 1990, p.148).

The previous chapter explored some of the key issues involved in measuring psychological complexity and reviewed some claims commonly associated with formal methods. In this chapter we review some cognitive theories of human reasoning which provide further impetus to our research methodology. In view of the logical calculi underlying most formal notations and the frequent requirement that software developers reason logically about formal specifications, we concentrate on the possible relations that may exist between logic and human reasoning. This discussion

provides pointers to those common properties of natural language that are likely to be a source of reasoning difficulty in formalised contexts.

3.1 Inference as a Central Cognitive Process

An "inference" is the mental process that we undergo when formulating a conclusion in response to given premiss information. This process effectively results in an "argument" whose believability normally rests on the strength of the link between its premisses and conclusion. Inference underpins the cognitive faculty for reasoning and is central to the notion of human intelligence (Rips, 1984). Virtually all everyday human activities require recourse to inference including elementary cognitive activities such as: language comprehension (Clark, 1977), visual perception (Eysenck and Keane, 1990), and decision making (Mullen and Roth, 1991). Cognitive activities such as these tend to be taken for granted to such an extent that we are seldom conscious of when we are inferring at all; we are only "aware of the results, not the mechanisms" (Johnson-Laird and Shafir, 1993). A chain of reasoning typically involves making a sequence of connected inferences, whereby the conclusion of each feeds into the premisses of subsequent inferences towards some overall conclusion. Reasoning might therefore be regarded as thinking with a view towards establishing a convincing implicative chain of argumentation.

Some common forms of inference are exemplified in Figure 3.1. Induction (Holland et al., 1986) involves generalising from a few specific instances to reach a more informative conclusion which might only be plausibly valid. Deduction (Rips, 1988) proceeds by making explicit information already given in the premisses to arrive at a less informative, but necessarily valid, conclusion. Probabilistic inference (Kahneman et al., 1991) uses explicit probability values from given premiss information as the basis for deriving a conclusion whose validity rests on statistical

likelihood rather than certainty. Calculation (Johnson-Laird and Byrne, 1991) might use values supplied in the given premisses and the laws of arithmetic as the basis for deriving a mathematically valid conclusion. A pragmatic inference (Levinson, 1983) might involve making additional assumptions based on popular belief or convention to reach a plausible conclusion. A plausible inference (Collins and Michalski, 1989) might use implicit probabilities suggested by the given premiss information as a basis for deriving conclusions which are plausible in the circumstances.

Inductive inference:	Deductive inference:
The first mature student passed	All scores over 60% pass the test
The second mature student passed	John scored over 60%
$\therefore$ All mature students passed	$\therefore$ John passed the test
Probablistic inference:	Calculated inference:
10% of lazy students pass tests	10 male students have passed
80% of diligent students pass tests	10 female students have passed
$\therefore$ A diligent student is more likely to pass	$\therefore$ 20 students have passed
Pragmatic inference:	Plausible inference:
Scores of 60% or more will pass	Diligent students normally pass
John scored 55%	John is a diligent student
$\therefore$ John has failed	$\therefore$ John will pass

Figure 3.1: Six common types of inference

Reasoning appears to operate in a field of its own and this is how it has generally been studied in the cognitive literature. Two fields which are sometimes perceived to have close links with reasoning, however, are those of ordinary thought and decision making. Reasoning may be distinguished from ordinary thought in that the latter operates by associating past experiences with one another and is therefore reproductive, whereas the former leads to new assertions and is therefore productive (James, 1950). Perhaps the most distinctive difference between reasoning and ordinary thought is that the latter does not have an explicit goal; that is, "it is not directed toward solving any problem or reaching any conclusion" (Johnson-

Laird, 1988, p.430). Reasoning may also be distinguished from decision making in that, besides making inferences, the latter also involves accumulating and aggregating evidence, speculating hypotheses, evaluating alternatives and applying criteria (Galotti, 1989). There is normally a close interdependence between reasoning and decision making because much of the reasoning that we perform in everyday life is used as a basis for reaching decisions (Johnson-Laird and Shafir, 1993; Shaw, 1996).

The human capacity for reasoning enables us to apply our knowledge to given situations. Indeed, if we were incapable of inferring new information from existing knowledge then it is difficult to see how we could ever learn anything; knowledge would have to be entirely specific or factual in order to be useful (Evans et al., 1993). In general, the greater our knowledge the more accurate our everyday reasoning and decision making is likely to be. The recall of misleading prior beliefs, however, can sometimes hinder cognitive performance, as we shall see.

3.1.1 Error and Bias in Human Reasoning

In an ideal world people's reasoning performance would equal, or even exceed, their reasoning competence. Whilst people often do know the relevant procedures to help them reach the correct conclusion, however, they occasionally fail to employ them. The chances of error-free reasoning in many decision making activities are remote because there exist only a few ways of arriving at the correct conclusion, and each inference in a chain of reasoning provides extra opportunity to stray in an inappropriate direction, to misapply procedures or to apply the wrong procedures altogether. Although much of the reasoning which is conducted on an everyday basis is adequate for its purpose, people have a notorious propensity for erroneous reasoning, sometimes at the most inopportune times, and "if it is impossible to guarantee the elimination of errors, then we must discover more effective ways of mitigating their consequences in unforgiving situations" (Reason, 1990, p.148).

Flawed reasoning exhibited by competent people may be attributed to numerous possible causes including: irrational thinking (Kordačová, 1994), inappropriate assumptions (Henle, 1962), absence or inaccessibility of the relevant procedures in memory (Inhelder and Piaget, 1958), application of inappropriate procedures (Rips, 1994), failure to consider all possibilities (Erickson, 1974), sound reasoning but from the wrong premisses (Henle, 1962), lack of experience or expertise (Reimann and Chi, 1989), misjudgements of value or probability (Kahneman et al., 1991), poor motivational states (Channon and Baker, 1994), and misleading prior beliefs (Barston, 1986). Regardless of how well psychology comes to understand the cognitive mechanisms responsible for human error, it is unlikely that the underlying causes of erroneous decisions will ever be completely eliminated. It is therefore important that we strive to reduce the consequences of reasoning errors in those areas which are particularly intolerant to them.

According to Reason (1990), slips occur when subconscious actions fail to achieve their intended goal and mistakes occur when intended actions are inadequate to achieve their goal. The term "error" is therefore meaningful when interpreted in relation to intentional cognitive states and is meaningless when applied to non-intentional human behaviour (Senders and Moray, 1991). In this thesis we focus on errors in consciously deliberated reasoning where the classification of inferences as "erroneous" depends on the "normative system" (Evans, 1993b) against which they are assessed. This system may comprise a reasoner's set of intentions or a completely independent system. Reasoning which conflicts with the dictates of logic, for example, may not necessarily be classified as erroneous if it led to a conclusion which is pragmatically sanctionable and the criteria used to assess its validity comprised real world knowledge and personal beliefs, rather than the formal rules of some logical calculus.

Human biases exist in a variety of forms, many of which hinder cognitive performance by encouraging people to ignore relevant parts, or focus on irrelevant parts, of the problem being solved (for a review see: Evans et al., 1993). Although bias frequently contributes to human error, a certain degree can nevertheless be helpful in certain situations. As methodical reasoning places considerable demands on cognitive effort, bias can occur in everyday experience as a consequence of people's natural tendencies to minimise cognitive processing load (Perkins, 1983; Perkins et al., 1991) and to disambiguate the interpretation of new information with recourse to prior beliefs and context (Gilovich, 1991; Sperber and Wilson, 1995). Our concern is with those human biases which are liable to arise in formalised contexts and cause reasoners to depart from deductive lines of thought. We are concerned, in particular, with those situations where non-logical biases or heuristics gain favour over logical deduction in determining whether speculative conclusions are accepted.

3.1.2 The Role of Deduction in Formal Specification

A deductively valid inference is one of the strongest possible forms of argumentation in the sense that, providing its premisses are true, its conclusion will also be true by necessity (Walton, 1993). Deduction proceeds by making explicit items of information which are already contained, albeit implicitly, in given premisses. The main limitation of the deductive form is that it does not allow reasoners to reach entirely new assertions. This is because a deduced conclusion is obliged to be more specific than its premisses and never to increase their scope. Furthermore, although deduction may be used to verify the logical validity of an argument, it cannot be used to determine whether the premisses upon which the argument rests are accurate pragmatic assertions.

"You need to make deductions in order to formulate plans and to evaluate actions; to determine the consequences of assumptions and hypotheses; to interpret and to formulate instructions, rules and general principles; to pursue arguments and negotiations; to weigh evidence and to assess data; to decide between competing theories; and to solve problems. A world without deduction would be a world without science, technology, laws, social conventions and culture. And if you want to dispute this claim, we shall need to assess the validity of your arguments" (Johnson-Laird and Byrne, 1991, p.2).

Although the ability to reason deductively is an important part of human cognition and a core component of rational thought (Rips, 1988), our everyday reasoning is guided mostly by informal inductive strategies based on probability, guesswork and intuition (Galotti, 1989; Shaw, 1996). Given that we often need to look beyond given information to draw new assertions, our everyday inferences are mostly based on probable truths and contingency rather than absolute truths and necessity. Our deductive cognitive processes do come to the fore, however, when we are conducting pure mathematics or logic (Russell, 1994; Strawson, 1966), and arguing critically in explicit steps (Walton, 1993).

The process of interpreting or reasoning about a formal specification requires recourse to numerous forms of inference, including those based on deduction, induction and pragmatics. Yet the deductive form is of particular relevance to this research, first, because it is one of the strongest and most prominent forms in the formal specification process, and, second, because cognitive science has reported a range of systematic human deductive errors and biases in informal contexts which could help to explain software engineering errors in formalised contexts.

Most formal notations are based on logical calculi, such as the predicate calculus (Lemmon, 1993), which are particularly amenable to deductive lines of proof and argumentation. Indeed, the ability to subject formal specifications to rigorous

logical analysis is commonly purported as one of their main strengths, particularly in the context of formal verification (Rushby, 1995; Wing, 1990). This may not, however, preclude the users of formal notations from employing non-deductive forms of inference even when these are clearly inappropriate and lead to errors.

3.2 The Role of Logic in Reasoning

Formal logic provides an objective, content independent means for evaluating the validity of deductive inferences (Adams, 1984). A system of formal logic comprises two main components: a symbolic notation whose grammar is defined precisely in mathematical terms, and a deductive apparatus comprising axioms and rules of inference (Woodcock and Loomes, 1988). Given that the syntax of the notation is comprised entirely of mathematical symbols, it is argued that this allows for more precision (Lemmon, 1993) and more concision (Barroca and McDermid, 1992) than can generally be achieved in natural languages. The inference rules of a logical system, when applied correctly, ensure that arguments with true premisses lead only to true conclusions. A formal deductive apparatus thus allows for the evaluation of an argument's form without recourse to any particular interpretation of its meaningful terms, and it is indeed with the structure of arguments that logic is concerned, rather than their semantic content (Lemmon, 1993; Macnamara, 1986).

> "To reason logically is to link one's propositions that each should contain the reason for the one succeeding it, and should itself be demonstrated by the one preceding it. Or at any rate, whatever the order adopted in the construction of one's own exposition, it is to demonstrate judgements by each other. Logical reasoning is always a demonstration" (Piaget, 1928, p.1).

Historically, reasoning competence has been equated with logical ability to such an extent that people shown to reason illogically were claimed to exhibit symp-

toms of irrationality (Evans, 1993b). Reber (1985, p.376) defines as "irrational" any kind of human thought which is "in violation of the rules of logic". This is not a view which is universally accepted in the cognitive community. Others propose that definitions of "rationality" might be better oriented towards the ways in which people behave in order to achieve goals or their personal assessments of subjective utility (Kahneman et al., 1991). The ability to make logical deductions is nevertheless regarded as a strong indication of rationality (Johnson-Laird and Byrne, 1993).

3.2.1 Logic as a Theory of Deductive Competence

The origins of formal logic can be traced back to the doctrine of the syllogism propounded by Aristotle (in Ross, 1949) in Ancient Greece to model the simplest form of deductive inference and to provide a content independent means for evaluating the validity of argumentation. For nearly two thousand years the authority of Aristotle remained unquestioned whilst it was universally accepted that deductive reasoning was reducible to syllogistic form and the syllogism was regarded as an accurate model of people's deductive thought processes. It was not until the advent of psychology as an experimental science in the nineteenth century that the validity of this assumption was brought into question and its psychological weaknesses began to be exposed (for a critique see: Beth and Piaget, 1966; Strawson, 1966). The possibility that Aristotelian logic might serve as a candidate model for accurately representing human deductive thought processes has been discredited in the cognitive literature because syllogistic reasoning rarely seems to occur in everyday argumentation (Newstead, 1989; Perkins, 1983), and in mathematics which is almost entirely deductive (Russell, 1994). Indeed, the translation of a formal or informal argument into syllogistic form would be artificial and might even confound matters by disguising the real nature of the original problem.

"The laws of thought, in all its processes of conception and of reasoning, in all those operations of which language is the expression or the instrument, are of the same kind as the laws of the acknowledged processes of Mathematics" (Boole, 1854/1958, p.422).

The models built from formal logic by several nineteenth century mathematicians aim to reflect as closely as possible the actual constituent laws and processes of human thought. These models, despite their mathematical origins, encapsulate theories of human reasoning and make strong psychological claims. The model developed by Boole, for example, aims to "investigate the fundamental laws of those operations of the mind by which reasoning is performed" (1854/1958, p.1), while the model developed by Mill aims to provide "a new theory of the intellectual operations" (1874/1986, p.iii). The works of these logicians gave rise to the theory of "psychologism" which claims that it is possible to find mathematical systems which are analogous to the laws of human thought. The question of how far such formal models reliably characterise human thought, however, remain to be answered. Although psychologism was widely condemned at the time (Frege, in Bynum, 1972; Husserl, in Findlay, 1970), the empirical tools and procedures necessary for testing its predictions did not become available until the twentieth century, following the development of cognitive experimental psychology. It has only been during the past three decades that cognitive science has returned to address the question of whether the human mind comes equipped with an inbuilt system of logic that guides reasoning, or logical systems can be built to mirror reasoning processes.

3.2.2 The Theory of Mental Logic

To many theorists, the notion that the core elements of human thought are anything other than logical seems incredible because, if they were not, it is unclear how people would be able to communicate, understand one another or work together (Henle,

1962; Cohen, 1981). In seeking more accurate representations of human reasoning, cognitive science has propounded numerous theories which argue that people are fundamentally logical in nature and has reiterated, to an extent, the claims of the nineteenth century logicians that reasoning is based on procedures akin to those found in systems of logic (Beth and Piaget, 1966; Braine, 1978; 1994; Braine et al., 1984; Henle, 1962; Inhelder and Piaget, 1958; Macnamara, 1986; O'Brien, 1993; 1995; Piaget, 1928; Osherson, 1975; Rips, 1983; 1994; Sperber and Wilson, 1995).

> "Deductive reasoning consists in the application of mental inference rules to the premises and conclusion of an argument. The sequence of applied rules forms a mental proof or derivation of the conclusion from the premises, where these implicit proofs are analogous to the explicit proofs of elementary logic" (Rips, 1983, p.40).

Proponents of "mental logic" theory argue that the human mind contains an inbuilt set of abstract inference rules which are used for constructing mental derivations of conclusions in a wide range of problem contexts, in a manner similar to the "natural" style of proof developed by Gentzen (in Szabo, 1969). Advocates of the theory claim that the types of inference rule stored in reasoners' mental repertoires are analogous to those found in "standard logic", whose roots lie in the propositional and predicate calculi, and whose rules relate to the connectives "if", "or", "and", "not", and quantifiers "all" and "some". Advocates claim that people's repertoires correspond only in part to the inference rules of standard logic (Macnamara, 1986), however, and that people are capable of drawing only some of the inferences sanctioned by standard logic (Braine and O'Brien, 1991). Advocates also claim that people who have acquired a high degree of deductive competence would, if logically possible, always employ the correct inference rule at the appropriate time in order to derive a valid conclusion (Henle, 1962; Inhelder and Piaget, 1958; Osherson, 1975).