INTUITION IN SOFTWARE DEVELOPMENT

P. Naur
Datalogisk Institut
Copenhagen University
Denmark

Abstract

A characterization of the pervasiveness of intuition in human conscious life is given, followed by some remarks on successes and failures of intuition. Next the intuitive basis of common notions of scales, logic, correctness, texts, reasoning, and proofs, is described. On this basis the essential notions of data models of human activity and of software development, as built on human intuition, are discussed. This leads to a discussion of software development methods, viewed as means to overcoming the hazards of intuitive actions. It is concluded that programmers' experience and integrity are more important than their use of methods.

1. Introduction

The purpose of the present discussion is to clarify the manner in which software and some of the notions and techniques entering into producing it are grasped by the human being involved. More particularly, an attempt will be made to make clear that immediate human apprehension, or intuition, is the basis on which all activities involved in software development must build.

The immediate reason for taking up the question of intuition is that in current discussions of programming there is a clear tendency to speak of intuition as an inferior human trait which is the cause of major difficulties in program development and whose influence on programming therefore should be eliminated. In the view to be presented here such a notion is based on a fundamental fallacy. What will be claimed is that intuition enters directly into any of the mental activities that together constitute software development. This holds in particular for the use of techniques such as special notations and modes of argumentation, which do not eliminate the influence of intuition, but rather must be regarded as aids by which certain risks of errors of intuitive insight may sometimes be reduced.

The questions to be taken up here are peculiar in that while they relate mostly to matters that, it must be supposed, are experienced by

every mature human being continually during all his or her waking hours, yet tend either to be ignored or overlooked, or when they are taken up for scrutiny to lead to controversy.

In the present discussion the notion of intuition will first be taken up for clarification. This is followed by discussions of several notions that enter into software development, such as scales, classifications, logic, correctness, syntax, semantics, and proof. This leads to a final discussion of software development and software development methods.

2. The notion of intuition

Intuition is explained by dictionaries in such words as "immediate apprehension by the mind without reasoning; immediate apprehension by sense; immediate insight". Thus it is used with two different, but related, senses, either about something happening, or about insight obtained as a consequence of that kind of happening.

Intuition tends to be overlooked because it is so all-pervasive, a so basic constituent of anything a person does, and works so smoothly and effectively. As we go through our lives, we see, hear, feel the touch of, things and persons around us, continually, throughout our waking hours, recognizing them and reacting to them. Only upon reflection will we realize that by far the most of the actions and insights involved in these transactions are intuitive, according to the definition. In fact, a person's intuition embraces his or her experience, knowledge, and memory. Each of these items of the person's mental possession are intuitive, in so far as they are readily available to the person himself or herself.

Part of our intuitive ability is to make sense of the impressions that we receive through our senses. However, the intuitive insight or grasp we have of our surroundings cannot be separated from that of language and of theories, nor can it be separated from our consciousness or world view.

A person's intuitive insight changes throughout life, by additions and modifications. One important part of a person's life is to add to his or her fund of intuitions, to the kinds of things that he or she can recognize intuitively.

Because of its pervasiveness, intuition tends to be referred to only when it is unexpected. For example, when some people talk about "feminine intuition" they clearly refer to insight had by women where they expect none. In quite another context, intuition is referred to by Medawar [1] in talking about "the generative act in scientific enquiry, 'having an idea'.[...] Intuition takes many different forms in science and mathemat-

ics, though all forms of it have certain properties in common: the suddenness of their origin, the wholeness of the conception they embody, and the absence of conscious premeditation". Popper [2] expresses himself similarly: "My view of the matter, for what it is worth, is that there is no such thing as a logical method of having new ideas, or a logical reconstruction of this process. My view may be expressed by saying that every discovery contains 'an irrational element', or 'a creative intuition', in Bergson's sense. In a similar way Einstein speaks of the 'search for those highly universal laws ... from which a picture of the world can be obtained by pure deduction. There is no logical path', he says, 'leading to these ... laws. They can only be reached by intuition, based upon something like an intellectual love ('Einfühlung') of the objects of experience'".

The view that intuition can refer only to extraordinary situations is so common that Quine [3] finds it necessary to defend himself against it: "Twice I have been startled to find my use of 'intuitive' misconstrued as alluding to some special and mysterious avenue of knowledge. By an intuitive account I mean one in which terms are used in habitual ways, without reflecting on how they might be defined or what presuppositions they might conceal".

3. Successes and failures of intuitive insight

As perhaps the most important characteristic of intuition, with normal, adult persons it is enormously successful. In the course of their lives such persons perform vast numbers of actions and interactions with the world, each involving huge numbers of muscular and nervous activities in incredibly complicated interplay. All this goes on day after day, and for by far the greater periods of time each human organism manages quite successfully in the activity at hand. We stretch out our hands, get hold of things, handle them; we move our body about among the multitute of other people and things, quite successfully; through our senses we get in contact with more distant parts of surroundings, recognize things and people known or unknown, without trouble most of the time.

But our intuition is not perfect, occasionally it does let us down, make us drop the cup we are holding, bump into something, make a mistake in recognizing a face, or in dialing a telephone number. Such failures can be taken as a warning that unaided intuition may lead us astray.

A further reason for distrusting intuition is that it is so temptingly convenient. Making the reaction or giving the answer that comes first to our mind is bound to be less troublesome than any other procedure.

An additional complication is that over a large range of a person's

reactions to certain common situations of life it is impossible to classify them as either successes or failures. There may thus be no basis for criticizing the reaction that comes to us intuitively. This holds particularly for reactions that involve understanding of language. Our understanding is, primarily, entirely intuitive, and furthermore develops throughout life, partly so as to keep up with the development of the communal language itself.

4. Bounding the failures of intuition

A main point of the present discussion is that all human activities must always remain entirely dependent on intuition, and that failures of intuition cannot be entirely eliminated. At best the ill consequences of them can be reduced. In the present section some basic patterns of how to achieve such reduction will be discussed.

Certain failures of intuition are in themselves irreversible, for example dropping a cup so as to break it. Other failures, such as dialing a wrong telephone number, can usually be put right by a simple corrective action. Even just these two examples suggest two important general issues in making our intuitions successful, first the recognition of the difference of importance between different kinds of failures, and second, the procedure, in certain situations, consisting of reaching a certain goal by repeating an action until success is achieved. Other action patterns along this line are achieving success by repeating a certain action and comparing the outcomes for sameness, doing actions carefully and slowly, and having other people check the outcome of an action.

What should be noted about these patterns of actions is that in their meaningful application they all depend on intuitive insight into the wider context of the action with which they are concerned. Thus one may say that they indicate ways in which intuitive insight can be made to support other intuitive insight, in other words a self-conscious mode of proceeding. Thus they conform directly to the scientific manner, described by Quine [3] in the words "science is self-conscious common sense". Closely related, they depend on self-criticism and an undogmatic attitude, with readiness to admit the limited validity of any insight.

5. Intuition, scales, logic, and correctness

The previous discussion of intuition has deliberately been phrased in terms of the vague concepts success and failure, without any attempt to clarify more precisely what these stand for. In this manner of speaking it is suggested that in our intuitive apprehensions we are perfectly

capable of dealing with the world and its quality without basis in criteria or scales of values.

It is further suggested that any kinds of well-recognizable categories, counts, and measures, in their application to the world depend entirely on our intuitive understanding. This means in particular that we may accept the application of certain categories, counts, and measures, without it being clear what their limitations are, neither with respect to their clarity of definition nor the scope of their application. This holds in particular for binary categorizations, such as right and wrong. For example, when we go on a trip we may say that we got on the right train and reached the right city, since trains and cities usually, although not always, are distinct enough. However, we are not normally prepared to say whether the weather turned out to be right or wrong, or whether the kindness of the population was agreeable or disagreeable, or even just put the quality of the weather and the kindness of the population on any kind of scale. Even so our notions of the weather and the kindness are perfectly meaningful intuitively.

These considerations show that any description of the world in terms of strict categories and scale values is no better than our intuition will make it. From this it further follows that such human failures or errors that depend in their very notion on strict categories are, in a sense, produced by the making of the categories as much as by the particular erroneous human action. Thus if there is no highway code there can be no traffic misdemeanour.

In using logic on matters of the real world we find the same dependence on intuition. This can be illustrated by looking more closely at a common logicians' claim, that our knowledge is made up of predicates. Take for example the statement, Paris is the capital of France. The logician will claim that this is essentially a predicate, that is something that may be true or false. As everyone knows, as part of our intuitive knowledge, the statement means far more than that. If we ask for a justification why we accept the statement, we do not ask for a proof. Rather we may mention many different kinds of evidence for the validity of the statement, without ever claiming neither their necessity nor their sufficiency. For example, the president of France is resident in Paris, the ministeries of the French government are located there, as is the chamber of the Deputies, every book about France will say that Paris is the capital, etc. etc.

In ordinary use of language we rarely deal with something close to the logician's truth and falsity. To claim that our knowledge of the fact that Paris is the capital of France is equivalent to accepting the truth of a predicate is misleading. Our intuitive knowledge, which ex-

tends without a clear bound into scientific knowledge, is a connected whole of innumerable items of insight and knowhow, continually adjusting us to the changing situation. Our acceptance of the statement is connected with our knowledge of how human beings arrange themselves into cities and states and may be supported by our having a theory of government and the significance of capitals. With this knowledge we will be able to give reasons why we find the statement acceptable, and might also indicate circumstances under which it would cease to be so. In this there arises no question of truth or falsity, in the intuitive sense. Such a question is relevant only in certain specialized situations, where the issue is whether someone has told a lie.

The dependence of logic on intuition is not confined to logic in its application to the world, but extends right into logic itself. In fact, even the notions of the best established formal theories depend on their being grasped intuitively by a human being. Thus as mentioned by A.N. Whitehead [4, page 266] the most basic constituents of formal logic, such as Proposition, are used with many different meanings in treatises written by highly acute modern logicians. This statement itself, it may be noticed, depends entirely on human intuition, since sameness of meaning has no sense independent of its being decided intuitively.

The situation of correctness is similar to that of truth. We do not ordinarily refer to the manner of operation of mechanisms we deal with by the word correct. We do not ask whether a car works correctly, but rather we may say that it is working order. This does not imply a set of sharp criteria to be satisfied, and indeed is compatible with our awareness that it has minor defects. What it means is that the car is in such a state that under a reasonably wide class of circumstances it will provide the kind of service we expect from a car. If we try to make the matter more precise we will find that we cannot. To a large extent the performance of the car depends on characteristics that do not admit such sharp limits that are needed in order to formulate sharp criteria. Again, if we adopt arbitrary limits on characteristics so as to establish a criterion, this does not guarantee that a car which is perfect according to the criteria will not be found to break down under certain conditions of stress that the user still would call normal.

This observation has relevant bearing on the question of program correctness. The situation may be described as follows. The user of a program clearly wants the program to work, that is to give the right results when it is used. However, this desire is not, and cannot be made, the basis of a once-for-all strict description of user requirements. Indeed, while there will undoubtedly be results of the program that the

user would unhesitatingly describe as correct or incorrect, most commonly there will be whole classes of results or reactions from the program to which the user would be unprepared and uncertain. To a considerable extent the user might, with good reason, respond to queries concerning such cases with the suggestion that many of them ought never to arise and so are irrelevant.

In response to this state of affairs some computer scientists insist that where no usable criterion of correctness is at hand it is the programmer's task to introduce one. For this purpose the programmer will describe the action of the program to be developed in terms of a set of strictly defined specifications. A program is then "correct" if its actions satisfy the specifications, a matter which is strictly defined. However, this matter of operating does not alter the dependence of the compatibility of texts and other circumstances of the world on being intuitively grasped. Thus the extent to which the texts of the specification and the matters of the world that are of concern to the user are compatible, remains a question that can only be ascertained intuitively.

6. Intuition and text

Software development to a considerable extent depends on the programmer's use of texts, where text is understood to include any data on readable form, including ordinary language prose, programs, and formulae of any kind.

As one of its important functions in the present context, text may serve as an aid to reducing the hazards of intuition, being a relatively stable record of something the unaided memory might not retain. This function of text directs our attention to the manner in which text is intuitively comprehended, or made to influence the knowledge had by a person.

In much recent discussion of text and language there is a strong tendency to take for granted that in dealing with texts one has to distinguish between things called syntax and semantics, and a corresponding unquestioned belief that the reading of a text must involve separate syntactic and semantic analyses. In computer oriented environments such a notion is further supported by the fact that in both the design and the structure of compilers for programming languages a division into syntax and semantics makes useful sense. From this background it would appear to be obvious that the distinction between syntax and semantics must likewise appear in some way in the manner human reading of texts takes place.

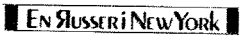
It must be pointed out that the assumption of an inherent distinc-

tion between things like syntax and semantics rests on several doubtful claims. First, that in analyzing a text it is generally possible to perform a syntactic analysis independent of the semantic analysis. Second, that the intuitive, human reading of a text is based on such a distinction.

On the first of these issues it should be noted that the concepts syntax and semantics, and several others, primarily have been introduced as aids to a scientific description of language phenomena. However, what concepts to use for describing any particular language is by no means given in advance. Rather, the choice of concepts will be a matter of descriptional simplicity and convenience in relation to the particular language at hand, and will moreover reflect the point of view adopted by the linguist. For example, Jespersen in his classical description of English [5] mentions that "grammar is usually divided into two parts: *accidence* - also called morphology - i.e. the doctrine of all the forms (inflexions) of the language, and *syntax*, i.e. the doctrine of sentence structure and the use of the forms." But he then goes on to explain his reasons why this type of division has been disregarded in his book.

More generally, the essential point about such concepts as syntax and semantics is that they are not issues in terms of which a language may first be presented to a person. Rather, they may, at best, serve to bring out aspects of a language to someone who already has the language, intuitively. Applied to the analysis of a text of the language the concepts can make sense only to someone who already understands the text. This point, as related to the teaching of mathematics, has been made Kline [6], who says: "The proper pedagogical approach to any new subject should always be intuitive. The strictly logical foundation is an artificial reconstruction of what the mind grasps through pictures, physical evidence, induction from special cases, and sheer trial and error. The theory of the calculus is about as helpful in understanding that subject as the theory of chemical combustion is in understanding how to drive an automobile."

The unity, or wholeness, of the manner in which a human being's intuitive grasp of a text takes place may be illustrated strikingly by the success of reading even in situations where an application of rules would fail. For example, the title of a film on show in Copenhagen at the time when this is written appears in the newspaper advertisement as follows:



Presumably any normal Danish reader will grasp this immediately to say, in Danish, "A Russian in New York". However, clearly any analysis of the text by means of rules would stumble even at the level of recognition of the single letters, since the first letter of the second word is not a Danish letter at all. The proper, intended understanding of the text depends on an interplay of what in an analysis by means of rules are entirely different levels, in addition to recognition of similarity or analogy which lies beyond any rules.

The essential unity of a person's intuitive comprehension of a piece of text is confirmed in the study by Ledgard et al. [7] on interfaces of an interactive text editor. As emphasized by the authors of this report, the subjects in the experiment "made no distinction between syntax and semantics. They simply could not conceive of editing power or function as something different from the appearance of the actual commands. To them, the actual commands embodied the editor to such an extent that many were surprised when told after the experiment that the two editors were functionally identical".

The most important point of the present discussion is to make clear how deeply a person's reading of a text depends on intuition, which cannot be understood in terms of applications of rules, such as rules of syntax. This point can be brought out most clearly if it is realized that a person's reading of a text can only be understood as one indivisible action, where the decisions that what is present before the eyes is in fact a text, that that text is relevant to the purpose at hand, say, the development of a piece of software, and that the text contains such and such statements, or formulae, or whatever, these decisions cannot be separated, as premises of an argument, but rather are consequences of the intuitive insight. The justification of this claim is best based on the well-known infinite regress of rules which would be invoked if the decisions had to be made separately, from rules. For example, take the first decision, that what is present before the eyes is in fact a text. If there were to be separate rules behind this decision the question would immediately arise how to decide that those particular rules are actually relevant. This new decision problem would then raise yet another need for rules about how to decide what rules are relevant, etc. in an infinite regress. Since this is absurd the initial assumption that the decision that a text is in view can be rule-based must be dropped.

The conclusion to be drawn from the discussion of the present section is that for the effective comprehension of a text by a person no particular aspect of the text can in general be said to be more im-

portant than any other. Thus, in particular, the common use of the designation syntactic sugar to denote a certain aspect of a notation suggests a contrast to matters supposedly more important that tends to be misleading. Consequently in evaluating the merit of a language or notation for use in supporting software development, the full range of its actual application by the actual programmers must be considered. In other words, the criterion that the notation works cannot be replaced by any criterion based on scale values or formal characteristics.

7. Reasoning, proof, and intuition

For the present discussion a crucial issue is the connection between reasoning, proof, and intuition. Reasoning, according to the dictionary, is the activity of forming or trying to reach conclusions from premisses by connected thought, silent or expressed. A proof, correspondingly, is the result of a successful piece of reasoning. A main point of the present discussion is that reasoning and proof, far from excluding or being opposed to intuition, are completely dependent on intuition. For a person to do reasoning and for someone to grasp the resulting proof, it is necessary that he or she has intuitive knowledge of each of the premisses, of their mutual connections, and of the total pattern by which they support the conclusion. This is in addition to an intuitive understanding of what a proof is. Thus a proof is an expression of intuitive insights patterned so as to show the reasoning supporting a conclusion.

As an illustration of a proof concerning such real matters as are also the concern of applied software, consider the following proof that there are now four apples in the basket. Proof: a minute ago I saw that the basket was empty; since then I have seen first Susan and then Barbara each put two apples into it, and nothing else has happened; and twice two make four.

This illustration, simple as it is, demonstrates all that goes into a proof about matters of the real world, and in particular exhibits the dependence on intuition throughout. Thus there is only intuitive insight behind deciding the emptiness of the basket, behind seeing the two girls with apples in their hands, behind relating that sight to the natural numbers, and behind the knowledge that the operation of multiplication is relevant to the situation. In a more careful analysis one might ask for an enumeration of the conditions for the validity of the proof. One may then be told that the basket is supposed to be an ordinary one, without a hole in the bottom, and that the validity is limited in time, until one of the apples has rotted away. But a very cautious respondent may want to add that the proof depends basically on the as-

sumption of a certain continuity of the world which is entirely beyond enumeration.

As also shown by the illustration, the proof depends on having a logical model of an aspect of the world, in this case amounting to using the whole numbers for counting such items of the world as apples. This model and the implied correspondence between certain things, apples in this case, and the logical constructs numbers can only be known intuitively. Part of this intuitive knowledge is that the model applies to apples, at least under certain circumstances. That this is significant insight may be seen from the fact that a similar model will not always work. It does not work for drops of water, for example. If we let first two and then again two drops of water fall into an empty cup it is unlikely that we will then find four drops in the cup.

Proofs concerned exclusively with the properties of mental constructions, such as those of mathematics, have a special character in so far as, since mental constructions may be endowed with eternal unchangability, they may establish results of eternal validity. This of course is a unique quality in a world which otherwise has nothing but inconstancy. This quality does not make the corresponding proofs less dependent on intuition, however. Both the mental items entering into the proof and their connections have to be grasped intuitively for the proof to be established.

While reasoning is entirely dependent on intuition, intuitive insight is only incidentally dependent on reasoning and proof. Throughout our lives our intuitive knowledge develops incessantly, the impressions from our senses combining with the knowledge already had to form ever new insight. Predominantly the combination of the new insights and observations with the previous knowledge is felt to follow without special effort, without appeal to laws, principles, or algorithms. Even so, if examined closer the activity involved cannot be distinguished sharply from that of reasoning. We speak of reasoning when the number of relevant circumstances and their connections is not too small and the conclusion follows only through their somewhat subtle combination, to such an extent that the formulation of the proof seems a worthwhile effort.

Making oneself familiar with a proof may contribute to adding the insight into the conclusion to one's intuitive knowledge, but is neither necessary nor sufficient for that purpose. One may accept the steps of a proof and yet fail to grasp the conclusion intuitively. Conversely, one may accept a statement without proof. Sometimes one may retain a conclusion, but have forgotten the proof of it. For example, to someone who has taken certain elementary courses of mathematics the statement "a quadratic equation has at most two different roots" may well have

become part of the intuitive knowledge, but the person may have forgotten how to prove it. On the other hand, in order to understand the statement the person must necessarily have an intuitive knowledge of such items as quadratic equations, roots, differences between roots, in addition to the knowledge of numbers.

Intuitive knowledge acquired with some aid from a proof is not essentially different from such knowledge that has not been connected with a proof. What the proof may contribute is intuitive knowledge about the connections between certain items of intuitive knowledge. Insight into the manner in which the premisses are connected together in a proof is significant in its own right. This follows from the fact that a person may be fully aware of all premisses and yet fail to draw the conclusion. This is a common failure, even in activities that put high premium on invention.

In summary, reasoning and proof do not provide insight independently of intuition, being in fact intuitive insights patterned in special manners.

8. Data models of human activity

Software is developed with the purpose of supporting human beings in their activity. Although the manner of support may take many forms and be concerned with many different aspects of human life, a common underlying principle of software solutions is the use of a model of the human activity in the form of data and data processes. As a way of showing the connection between the notions of data, data processes, specifications, data representations, formalization, and proof, a human activity and its support by a model will be briefly sketched. The activity chosen, although simple, will be found to bring out most of the typical problems arising in designing data models.

The activity to be considered as illustration is stockkeeping in an ordinary household. In one manner of solving this problem all kinds of supplies are obtained at regular intervals, once in every shopping cycle. This means that the shopper of the household in each shopping trip must undertake two main actions, first, establish the shopping needs on the basis of the remaining stocks and the expected consumption for the period of the following shopping cycle, and second, buy the goods needed and bring them home. In implementing the shopping according to this general manner it may be helpful to make use both of specifications and formalized descriptions. The stocks that should be established in each shopping trip can be specified as a stock plan, that is a list giving the minimum and maximum amount for each kind of goods, determined

from the average consumption of the household, the storability of the kind, and the storage capacity of the household. In the first action of the shopping trip the shopper will combine this specification with data on the remaining stock of each kind of goods and so produce a shopping list, that is a formal description showing what kinds of goods to be bought and how much of each. In the second action of the shopping trip the shopping list is used in selecting the goods from the shelves of the shop.

An examination of principles and actions of this household stocking procedure will show that human intuition has to be brought in at all stages of both the design and execution of it. One may want to claim that the procedure can be derived systematically from one single, overall requirement, such as that the household must at no time run out of supply of any kind of goods. In stating this it should be noted, however, that in its very statement the requirement definition depends on an intuitive understanding of the total situation of the household and such items as supplies and kinds of goods. In establishing the principles of the procedure to be applied at each shopping trip we depend on an intuitive understanding of the consumption of goods in a household and the relevance of the formal concept average consumption. This intuitive understanding in particular is necessary in establishing the connection between the shopping procedure and the overall requirement of the procedure. It is further decisive in determining the limitation of the validity of procedure, and in designing margins and safeguards in the procedure, so as to make it valid even for certain classes of abnormal situations. At the same time it must be obvious that no procedure will be able to ensure the satisfying of the requirement under any conceivable circumstances. For example, if the household stock of any particular kind of goods disappears from theft or is destroyed by fire, then at that moment the requirement ceases to be satisfied.

During the execution of the household stocking procedure intuition enters every time a human being deals with an item of goods in any manner whatsoever, first of all in identifying it, and possibly in determining it more closely with respect to quantity and quality. The intuitive insight obtained by this direct contact with the goods may then, again intuitively, be related to the formal descriptions of the stock plan or the shopping list.

The possible application of a formal proof as part of the design of the household stocking procedure again depends on intuitive insight in several ways. For example, the design might include an algorithm for determining the minimum and maximum amount of goods, for use in establishing the stock plan. The design might include a proof that the over-

all requirement is satisfied when this algorithm is used. For this step in the design to be valid it must be intuitively clear that the proof criteria do indeed justify the conclusion drawn from the proof. Moreover, the proof will undoubtedly depend on premises that express restrictive conditions on the situation. It will be a matter for the intuitive insight of the designer to establish whether or to what extent these restrictive conditions are or can be satisfied.

9. Software development

For the present discussion of the importance of intuition in software development, a vital issue is the proper view of the development process considered as an interplay of intuitive knowledge had by the programming person and the real world accessible to that person, which includes the texts used or produced by the person. In the following discussion the word knowledge will be used to indicate intuitive knowledge had by a person. A major source of difficulty, and possibly of confusion, is the need to talk of parts of that knowledge at the same time as it must be recognized that a person's knowledge at any time is an indivisible whole. In the following discussion this recognition is reflected in that the identifiers used to indicate the person's knowledge, K1, K2, ..., refer not to parts of that knowledge, but to that total knowledge as it is increased in the software development process.

The real world, which is open to inspection by the programmer and other persons, will be denoted W. Certain parts of it are recognized intuitively by the programmer and other persons as having the special nature of texts of some language or notation and will be denoted WT1, WT2, ...

The simplest conceivable software development activity can now be described in terms of the following items:

- W. The real world, as accessible for inspection to several people.
- WT1. Text describing the programming language.
- WT2. Text describing the problem to be solved by the program to be developed.
- WT3. Text of program and its documentation.
- K1. Knowledge of the world, of ordinary language, and of the programming language to be used.
- K2. K1 with the addition of knowledge of the problem to be solved.
- K3. K2 with the addition of knowledge, or theory, of the programmed solution.

The software development activity or process typically proceeds as fol-

lows. Initially the programmer has K1, and W and WT1 are available. Here K1 and WT1 must be assumed to have a certain close relation of intuitive compatibility, in the sense that the person, having K1, upon inspection of WT1 will find at least most of it familiar and understandable. On the other hand, it would be misleading to claim that K1 includes merely a kind of copy of WT1. In acquiring K1 the person may have used WT1 as an aid in getting to know the programming language, but establishing K1 has also depended to a large extent on previous knowledge of programming and of the world and language generally, and on the accessibility of W.

The software development activity may be initiated by making WT2 available to the programmer. This will be a decisive source in the programmer's acquiring K2, which, however, inevitably will also depend strongly on K1. Indeed, even just understanding a problem of programming depends on the person's having a background knowledge of programming and of a programming language.

Of similar importance, in acquiring K2 the programmer must relate the contents of WT2 to his knowledge of the world, which is part of K1, and may also have to refer to W. These combinations of parts of the real world and knowledge are only conceivable as manifestations of intuitive insight. Indeed, a text, such as WT2, can be related to the world, W, only through on the one hand, understanding and interpretation of the text, and on the other hand, selection of and assignment of significance to the features of the world. All of this can only make sense if understood as elaborate, purposeful actions undertaken by the programmer, depending on intuitive insight at every turn.

The actual software development, if pursued strictly as a sequence of phases, consists in the programmer's forming the theory of the solution, and thus in acquiring K3, and then in the writing of WT3, the text of the program and its documentation. On such a view the program text, WT3, is produced purely from the fully formed knowledge K3, as an expression of a certain aspect of that intuitive knowledge.

Probably software development will rarely, if ever, proceed by the phases suggested in the previous paragraphs. Correspondingly, the strict distinction between K2 and K3 probably rarely is possible. Rather, the development will proceed as a continued interplay between the programmer's growing insight, both into the problem and its solution, and the real world items, W, WT2, and WT3. The growing, intuitive insight will tend to be supported by the gradual production of the documentation and program, WT3, and at the same time new uses of the accessible W and WT2 will contribute to forming K2. Occasionally the programmer may also need to refer to WT1, the description of the programming language, although

to be at all effective he or she must be able to rely mostly on K1 for knowledge about the programming language.

Whether the software development proceeds more closely as a sequence of distinct phases or as one more complicated phase, the overwhelming importance of the programmer's intuition must be clear. At all stages the activity depends on combining items of the world that have no inherent affinity apart from the programmer's intuitive understanding, interpretation, and notion of purpose and significance.

10. Software development methods and intuition

In the discussion of this section a software development method is any set of rules designed to influence how the programmer proceeds in his or her task, beyond the rules inherent in the development activity described in the previous section. The rules of a method will normally be mutually dependent and supporting. The matters of concern of a method can be divided into three major areas, as shown below.

1. Activity. Rules about what should be done or produced, in addition to what is covered by the minimal software development actions. Typical items under this heading are: produce descriptions of variables; produce assertions of invariant properties of the variables of the program execution; produce descriptions of the program logic in other forms than the program itself, such as for example specifications of the relation between input and output; produce proofs of the consistency or compatibility of certain expressions; perform particular check operations; perform walk-throughs.

2. Forms of expression. Particular notations or languages that should be employed for various purposes in the software development.

3. Ordering of activities. Imposition of particular orders in which the activities of the software development should be undertaken, such as top-down, or stepwise refinement.

A major issue of any particular software development method is the extent to which it is effective, i.e. whether and how much it contributes to improving the software development in which it is adopted. In principle it would be desirable to have the effectiveness of methods determined empirically, by means of observations of actual software development activities. In practice such an approach meets difficulties of several kinds: In trying to detect improvements, what software development activity can be taken as basis for the comparison? How can the improvements be evaluated or measured? How is it possible to make sure that such improvements that are in fact detected are the result of using the method? Quite generally, any empirical study of a method de-

signed to improve conditions of a poorly developed kind of activity is faced with a difficulty described by Bernard Shaw [8] under the heading The Surprises of Attention and Neclect: when an activity is in a state of neglect, any attention given to it, even such that are based on entirely misguided notions, is likely to improve matters. Clearly, under such circumstances an improvement brought about by the application of a method is no sure evidence of the efficacy of the method.

As an alternative or supplement to empirical evaluations, software development methods will here be evaluated on the basis of their relation to intuition. What can be noted, first, is that, as is the case for all other software development activities, the use of the rules of a method depends directly on the programmer's intuition in deciding when each rule of the method applies and how.

As the second major issue, the various kinds of rules of methods must be related to the shortcomings of the programmers' intuitive insight and behaviour. These shortcomings are of two kinds: (1) *omissions* to do what the programmer is intuitively aware should be be done, and (2) *flaws* in the actions taken intuitively.

Considering first omissions, the first area of concern for methods, activity, clearly is highly relevant and potentially useful. Even just a simple check list of activities that experience shows may be relevant in software development may be effective in avoiding omissions. Having such a check list and insisting that it is used might well be part of a methods in the area of activity. The second area of concern of methods, forms of expression, is of no direct importance to omissions. The third area, ordering of activities, is slightly relevant, in so far as even a very simple activity aid, such as a check list, must be used at not too late stages of the software development, if it is to be useful at all.

Turning to flaws in the actions taken intuitively, the activities prescribed by a method are highly important in so far as they insist on the programmer performing checks. The checks may be just that, doing the same work another time and verifying that the results are the same. However, the aspect of check is implicit in several other kinds of activities, including the production of descriptions of the program logic in other forms than the program itself. Comparing direct and implicit checks, each kind has its merits and limitations. The advantage of direct checks, based on repetition of actions, is that there is no question about what it is they verify. The limitation is that if done by the same person there is a strong risk that the same flaw of intuition will repeat itself. This limitation may be overcome by having the check done by another programmer. Implicit checks avoid repetition of flaws of intuition, but pose new problems of the extent to which they are ef-

fective in verifying solutions. Usually only certain aspects of a solution will be verified, and typically such aspects as the spelling of variable names is left almost wholly unchecked.

The relation between flaws of intuition and the imposition of particular forms of expression of a method is a complicated one. The claim is often made that certain forms, or formalizations, will guarantee the absence of flaws of arguments. What seems to lie behind such claims is the fact that by the use of certain kinds of formalizations it is possible to formulate the connections between statements corresponding to a proof in terms of rules for manipulating the statements. While this property is of great interest as a matter of principle, and also is the necessary basis for mechanical proof construction and verification, and occasionally is used in the reasoning carried out by people, it provides no guarantee for the absence of flaws in the arguments used in software development making use of formalizations. In fact, the compatibility of descriptions used in developing a piece of software and the matters of the world that are supposed to be modeled by them remains a matter for human intuition in any case. Avoiding flaws in that modelling undoubtedly depends to some extent on the form or language used for the description. However, what is the most suitable form or language for any situation will depend not only on the degree of formalization of the description, but equally on the character of the aspect of the world to be modelled and the background experience of the programmer who has to establish the description. Thus to claim, for example, that the mathematical properties of a form of notation have to be decisive in choosing it for software development, implies a gross disregard for the importance of human intuition. It must further be concluded that rules of a mathod that impose the use of particular, restricted forms of expression on the programmer may in fact contribute to introducing flaws in the software product. For further discussion of this issue, see [9].

Relating, finally, flaws of intuition to the third area of concern of methods, ordering of activities, it seems likely that imposed orderings will not help to avoid flaws and may in fact contribute to them. Indeed, the task of software development involves the programmer's dealing with complicated patterns of interconnected restrictions and concerns and deriving new, relevant conclusions from them, intuitively. Avoiding flaws of such derivations depends to a considerable extent on the programmer's concentration in retaining complicated patterns. Such concentration may well be disturbed or interrupted by externally imposed demands on the ordering in which certain matters should be attended to.

As a summary of the above discussion of methods and the failures

of intuition, methods appear to be useful primarily by providing check lists that may help programmer's to avoid flaws of omission, while the rules of methods in the areas of form of expression and ordering of activities are of doubtful utility.

The conclusion just suggested raises additional questions concerning the most pressing problems of software development. Indeed, in evaluating a method the question is not only what advantages might accrue from using the method, but also how can one make sure that the programmers in a particular project do in fact follow the rules of it? And more generally, how can one make sure that the programmers make proper use of relevant, well-known results and techniques? The question may be asked whether the flaws of software systems are caused primarily by lack of methods, or perhaps rather by failures to make proper use of such techniques that are generally well known, by simple neglect. The suggestion implied in this question gains some support from experience gained in other large scale human construction activity, such as the construction of buildings. For example, an investigation made in 1984 of buildings constructed in the year 1979 in Denmark reveals a large number of flaws of construction that predominantly are such that could have been avoided by application of perfectly well established techniques, which have been neglected from sloppiness.

The indication of such experience supports the above discussion of intuition in the conclusion that the problem of high quality software development cannot be solved by rules and methods, which essentially assume that the programmer acts like a machine for producing programs. As an alternative it will here be suggested that the primary task of the programmer is to build theories of the way the problems at hand can be aided by a computer program, an idea discussed in more detail elsewhere [10].

11. Conclusions

The major conclusion of the present discussion is that software development in all its phases, and irrespective of the techniques employed in its pursuit, must and will always depend on intuition.

The fundamental way of reducing failures of human intuition is to apply multiple work and check. Rules for guiding the software development depend on intuition to decide where and how they apply. Consequently a view of software development that makes the application of rule--based methods and notations the basic issue is misguided. The deeper problem of software development is the programmer's building of theories of the computer-based solutions.

Finally, to the question of this conference, what is the role of semantics in software development? Answer: neither that of the composer, nor of the librettist, the conductor, the hero, or the heroine, but that of the prompter, who does nothing but tell the actors things they know already, but that may momentarily have slipped from their minds.

References

1. Medawar, P.: *Pluto's republic*. Oxford University Press, Oxford 1982.

2. Popper, K.R.: *The logic of scientific discovery*. Hutchinson, London, 1959.

3. Quine, W.v.O.: *Word and object*. M.I.T. Press, Cambridge, Massachusetts, 1960.

4. Whitehead, A.N.: *Adventures of ideas*. Pelican Books, Harmondsworth, Middlesex, England, 1942.

5. Jespersen, O.: *Essentials of English grammar*. George Allen and Unwin, London, 1933.

6. Kline, M.: *Why the professor can't teach*. St. Martin's Press, New York, 1977.

7. Ledgard, H., Whiteside, J.A., Singer, A., Seymour, W.: The natural language of interactive systems. *Comm. ACM* 23 (10), pp. 556-563, 1980.

8. Shaw, B.: *The doctor's dilemma*. Penguin Books, Harmondsworth, Middlesex, England, 1946.

9. Naur, P.: Formalization in program development. *BIT* 22 (1982), 437-453.

10. Naur, P.: Programming as theory building. Microprocessing and Microprogramming, in preparation.