How to Eat a Cake Without Having It:

An Evaluation of the IFIP Working Group 8.1 Report
on Information Systems Methodologies

Roel Wieringa
Wiebren de Jonge

Department of Mathematics and Computer Science
Vrije Universiteit
De Boelelaan 1081
1081 HV Amsterdam
The Netherlands

uucp addresses: roelw@cs.vu.nl, wiebren@cs.vu.nl

ABSTRACT

This paper reviews some aspects of Olle et al. [1988]. After an analysis of the terms "method" and "methodology", it is concluded that the book is an exercise in information system (IS) development methodology. We then argue that the aim of the book, the provision of a framework which should help in understanding existing IS development methods, is only partly reached, and that much work towards this goal is still to be done. Finally, we indicate some of the improvements which can be made to the text with respect to the presentation of the material as well as to the attainment of the goal.

Contents

1. Introduction

The appendix of the book [1] lists 34 different IS development methods. Clearly, as the authors state in the preface, there is a need for fundamental research in the nature of these methods, research which attempts to answer such questions as what the similarities and differences between these methods are, whether there are criteria for choosing between different methods, and whether there are relations between the development methods to be used and the kind of system to be developed. The attempt to study these questions is to be applauded. It is not an easy task, not only because of the sheer number of development methods or because of the bewildering variety in terminology current in these methods, but also because the criteria for a good system development method are not yet clear at all and because the environments in which these methods are used, organizations, are complex, vague, ill-defined, dynamic, sensitive to the presence of the system developer, and at times chaotic, to mention only a few of the problems.

It is therefore not to be expected that a first attempt at providing a framework for system development methods will succeed in all respects. In this paper we will look at the extent to which the attempt in the book has succeeded and, more interesting because it indicates future work, how it has failed to reach the goal. In order to get to grips with what the book is about, in section 2 we criticize the use of the word "methodology" in the book. In section 3 we then look at the stated aim of the book, in particular at the meaning of the word "framework." This gives us a preliminary standard by which we measure the achievement of the book. In section 4 we discuss some of the deficiencies in detail. Section 5 summarizes the conclusions.

2. Method versus methodology

As the title indicates, the book is about information systems methodologies. Use of the term "methodology" instead of "method" is motivated by reference to the common practice over the past decade (page 1). This is to be regretted, for use of the term "methodology" to mean method has caused genuine methodology to go nameless and therefore ignored (Jackson [1983]). We will use "system development method," or "method" for short, with the meaning of a way of working, thinking and writing in the development of a system. "Methodology" will be used with the classical meaning of a study of (IS development) methods.

Note that we do not intend to correct English as it is used. Instead of legislating the use of words, we want to have two different words for different and important concepts. Which words are actually used is less important than that the distinction is made. If "methodology" is used to mean method, another word must be found to be able to talk about methodology. In this paper, we stick to the classical meaning of "method" and "methodology."

A method is thus a way of organizing the system development process, a way of organizing thought about the universe of discourse (UoD), and a notation in which to represent and communicate the structure of the UoD as it is explicated by following the method. A method is normative; it prescribes how work ought to be organized. Similarly, a methodology may be normative, for it studies the logical structure of one or more methods and may justify certain methods in terms of the way they facilitate the development of "correct" systems.

We will substitute "development method" for "methodology" in quotations where we think development method is meant, and indicate this by enclosing "development method" in square brackets. The original quotation can thus be regained by substituting "methodology" for "[development method]".

Applying this to the title of the book, we get "Information Systems [Development Methods]",

1. In this paper, the unqualified phrase "the book" refers to Olle et al. [1988]. All page references, unless otherwise stated, refer to the book.

which cannot be what is intended. What is meant is probably "Information Systems Development Methodology," which is a small change as far as syntax is concerned, but a huge change in semantics. It would tell us that IS development methods are studied, just as the authors say in the preface of the book. Note that with the book's interpretation of the word "methodology," the title suggests that the book contains IS development methods.

2.1. An essential difficulty with methodology

The study of method is essentially more difficult than the study of other kinds of UoD, as an analogy with methods and methodology of physical science will make clear. Popper's [1959] theory of falsification is a theory about the proper method to be followed in physical science. Simply put, Popper's theory says that any scientific theory should be rejected as soon as one of its predictions is falsified, i.e. contradicted by observations. As a theory about methods, this can be tested by confronting it with the methods actually used by researchers. However, it turns out that researchers actually do not behave as Popper describes/prescribes: they do not always reject a theory when it is falsified.[2] There are now two options: either Popper's theory of method is wrong and should, according to its own criteria, be rejected, or his theory is correct and the actual method used by scientists is wrong and should, in the future, be improved according to the norm laid down by Popper's theory.[3] Whatever methodology we adhere to, there are always two options when a theory of method clashes with actual methods: we can reject the theory or we can reject the methods. The availability of these two options makes confrontation of a theory of methods with actual methods inherently more difficult than confrontation of a theory of a physical universe with actual behavior in that universe, because in the case of a physical theory we can never conclude that the UoD behaves in a way in which it ought not to behave.[4] Whereas a theory of method contains a normative element, a theory of the physical universe does not. The study of methods (i.e. methodology) is therefore always more difficult than the study of other kinds of UoD.

2.2. The notion of a correct system

In the case of the methodology of IS development, there is an additional problem, concerning what is meant by a "correct" system. In physical science, a correct system is a system which truthfully represents a UoD. Such a system is an abstract representation of entities and processes in a UoD, i.e. it is a *theory* of a UoD. For the development of theories about physical UoD's, empirical methodology prescribes the empirical method, which is a cycle of observations, hypothesis formulation and testing by observations.

In information systems (IS) development, a correct system should not only truly represent the UoD, it should also be able to function as part of the UoD. If the UoD is a human enterprise, it may take over some of the tasks formerly executed by people. This complicates the development method and its justification. Ultimately, political processes may influence the choice of one development method rather than another, and may cause some aspects of a development method to be omitted or distorted. The methodology of IS development, which is the study of methods of IS development, is

---

2. The classical reference for this is Kuhn [1970].

3. Here another fundamental difficulty of methodology emerges: if Popper's theory is wrong, we do not have to follow its prescription of rejecting a falsified theory. In particular, Popper's theory, though falsified, need not be rejected. But if we do not reject it, we ought to follow it, which means that we should reject falsified theories. The reader can complete the argument for him- or herself. In general, a theory of the method of theory development applies to the method of its own development.

4. But there are other reasons for complication here. If a theory of a physical universe is contradicted by an observation, we may conclude that the observation is erroneous, or an anomaly, or not worth the trouble of accounting for. In general, as long as there is no alternative theory which, according to some standard, is at least as good as the existing theory, the latter will not be rejected no matter how bad it accounts for the facts. See Lakatos [1970] and Laudan [1977].

thus fundamentally more complicated than the methodology of empirical science, which is the study of the development of theories about physical UoD's.

3. About the aim of the book

When we open the book, our expectations that it contains methodology, not a particular method, are confirmed. In the preface we read that it does not intend to provide a new system development method, nor to integrate existing methods, but that

> "… the book defines a framework into which many existing [development methods] can be fitted" (page vi),

and that

> "The aim of the book is to present the techniques used in information system [development methods] in a way that shows how the various techniques interact with each other and how such techniques may be integrated." (page viii)

A *technique* is defined on p. 11 as "a part of an information systems [development method] which may employ a well-defined set of concepts and a way of handling them in a step of the work." Some examples given of techniques are entity-relationship modeling, data flow analysis and transition nets.

We will discuss first the meaning of the definition of "technique," then we explicate the aim in the book, and finally we evaluate the extent to which the aim is reached.

3.1. The definition of "technique"

The definition of "technique" contains a number of imprecisions which we think are not intended. It defines a technique as a part of a development method which *may* employ a well-defined set of concepts. This implies that there are techniques that may as well do something else, such as employ an ill-defined set of concepts, which presumably is not what is intended. Secondly, a well-defined set of concepts is not the same thing as a set of well-defined concepts, which is probably what is intended. Thirdly, a *part* of a development method must probably be construed as a logical part and a *step* as a temporal part, i.e. an interval in the sequence of actions to be performed. With these clarifications, a technique is defined as a logical part of an IS development method which employs a set of well-defined concepts and a way of handling them in a step of the work.

This does not yet make sufficiently clear what a technique is. Any part of a development method ought to use well-defined concepts which are handled in a development step. The examples given do not clarify which logical part of a development method is meant, for ER modeling and data flow analysis are activities, not techniques. It may very well be that examples of techniques are interview techniques and/or representation techniques. If *representation techniques* are meant, then the examples of techniques given concern ER diagrams, data flow diagrams and transition diagrams. In this case, a technique is a graphical notation method, i.e. a way of writing, which is a logical part of a method. The relation of techniques to development steps is then that a technique is used to represent the result of a development step. If on the other hand *interview techniques* are meant, then the examples given of techniques cannot really be understood, and a technique is a way of working. The relation to development steps is then that an interview technique is used to obtain the information needed in a development step. If representation- as well as interview techniques are meant, and perhaps other techniques besides as well, then this is not made clear in the text. More clarification is needed in this respect.

---

5. According to one view (which we endorse), a method consists of a way of writing, a way of working and a way of thinking.

3.2. The aim of the book

Ignoring the concept of technique, the aim of the book can be read as the provision of a *framework* that helps in understanding IS development methods. What is a framework for development methods?

Any framework is not a framework of an individual entity but of a certain *kind* of things, a set of possible things which we may call the UoD of the framework. In the case at hand, the UoD of the framework is the set of all actual and possible IS development methods.

However, a framework is not the same thing as its UoD. A framework for development methods should contain definitions of *key concepts* relating to system development, so that actual (and possible) development methods can be related to the framework, and through the framework to each other. The key concepts thus *describe* the area, and implicitly provide *yardsticks* with which to compare actual and possible development methods. A framework may also contain explicit normative statements like "a development method of a system of type X should distinguish analysis from design" and "in any method, the results of analysis should be verified with users of type Y and Z in the UoD." Even if no particular method is actually compared with the yardsticks, it should be possible to do so. A framework thus contains at least descriptive, but often also normative statements.

Thus construed, a framework is a *conceptual model* of a UoD which contains

1. descriptive, and often also,
2. (implicit or explicit) normative statements about the UoD.

Thus, the task of providing a framework for IS development methods is the task of providing descriptions of and norms for IS development methods. Such a framework provides a conceptual model of IS development methods. Constructing the framework is an exercise in methodology. Because the word "methodology" has been disowned in the book, the previous statement cannot be formulated this way in the vocabulary of the book.

We agree with the authors that development of a framework for IS development methods is not the construction of a new method. However, the framework should be applied to particular methods in order to test it and, if confirmed well enough, it should be applied to particular methods in order to evaluate them. During the *development* of a framework for development methods it is confronted with actual methods primarily to test the framework's descriptive qualities (accuracy, scope, etc.). After the framework is reasonably confirmed, it can be *used* to evaluate different methods. We emphasize here that part of the difficulty of developing a framework for IS development methods is that there is no firm criterion for the degree of confirmation of such frameworks, so that there is also no sharp dividing line between the period of development and of use of a framework for IS development methods. No matter how well the framework is confirmed, there is always the option of rejecting the framework instead of the method to which it is applied. This brings us to the question to what extent the aim of the book is reached.

3.3. Is the aim reached?

We saw that one of the stated aims of the book is the presentation of the techniques used in information system development methods in a way that shows how the techniques interact and how they may be integrated. A cursory glance through the book shows that this aim is not reached: only one representation technique is used throughout the book.

Secondly, we argued that as an exercise in methodology, the framework should contain definitions of key concepts (which have a descriptive and often implicitly a normative character). This aim is reached to the extent that chapter 1 defines concepts applicable to a variety of IS development methods. In section 4 we will see that success in this respect is at most partial.

Thirdly, we argued that the framework should contain descriptive statements about development methods. The book contains numerous descriptive statements of a rather vague nature, such as

"Many ways [of business analysis] call for a specification of the ATTRIBUTES of each ENTITY TYPE" (page 66; capital letters are used in the book for names of entity types in the conceptual model of development methods which it presents). For any of the four groups for which, according to the preface, the book is intended, viz. students, teachers, practitioners and researchers, this is a piece of non-information. What is required in a book on methodology is a discussion of the concepts of attribute and entity, a survey of the way these concepts are used in different IS development methods, and a definition of the way(s) these concepts ought to be used motivated by this discussion and survey. What we find instead are statements like

> "There is frequent confusion about the use of the term 'attribute' and some approaches to data analysis use the term as independent of ENTITY TYPE - implying the assertion that 'the same attribute can belong to more than one ENTITY TYPE'. This latter interpretation of the term 'attribute' is explicitly excluded here." (page 66)

This can hardly be called a clear explanation of the different ways in which the term "attribute" is used, nor is it a motivated choice for one of these uses.

This brings us to our fourth point, which is that a framework usually also contains (implicit or explicit) normative statements. These can be found in the book as well, such as the statement quoted above, or the (implicit) statement that analysis ought to be separated from design (but see the discussion below in subsection 4.6). However, these statements are rather vague and are insufficiently motivated, or not motivated at all. Moreover, some belong to the level of particular development methods, not to the level of a theory about methods. For example,

> "Each ATTRIBUTE *must be* an ATTRIBUTE of an ENTITY TYPE. This means that each of the above ATTRIBUTE NAMES *should be assignable* to one of the eight ENTITY TYPES already identified." (page 73-74, italics added)

This statement belongs to a particular method, not to methodology. Methodological questions concerning the quoted statement are which concepts of entity and attribute are presupposed, and whether the requirement expressed in the statement is a valid one, given these concepts of entity and attribute.

Our fifth point is that the concepts, descriptions and norms developed in a methodology should be applied to particular development methods. Here there remains a lot of work to be done, for the book contains no application of the framework to even one particular development method. What is worse, where statements pertaining to a particular development method appear, such as above, they are not recognized as such.

As a further substantiation of the claim that the above standards by which we measure the achievement of the book are reasonable, consider the questions raised in the preface (page vii) and in the introduction of chapter 1 (page 2). These are identified as some of the "more significant" questions to be answered in the provision of a framework for IS development methods, so the reader has the right to expect them to be answered, or at least discussed, in the book. These questions are (page 2):

1. "Are there really so many substantially different ways to design a computerized information system?
2. If not how are these ways similar?
3. Should we be using one [development method] for one kind of information system and a different [development method] for another?
4. If so, which [development method] should we use for which kind of system?"

However, we have not found an explicit discussion of these questions in the book.

We conclude that the aim of providing a framework for IS development methods is not reached in the book in that definitional, descriptive and normative matters are either not attended to or treated only partially and sloppily. The last statement is substantiated by presenting a few examples in

section 4 of this paper.

4. Presentation of the material

In this section, we give some comments on the presentation of the material. The order in which this is done, and the length of the treatment of each topic, stands in no relation to the relative importance of each topic. As a preliminary to the rest of this section, we first discuss the diagram technique used in the book.

4.1. Diagram techniques

The diagrams use squares to indicate entity types and Bachman arrows to indicate many-one relationships. A Bachman arrow points in the many direction, as illustrated in figure 1a. Figure 1a shows that a car has precisely one color, but that each color can belong to zero or more cars.

Figure 1a.

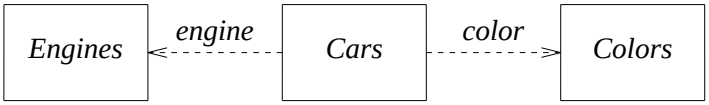

Figure 1b.

We view attributes as functions and represent them by *functional arrows.* For example, *color* is a function *Cars* $\longrightarrow$ *Colors* and is represented as in figure 1b. Note that the functional arrow has the reverse direction from the Bachman arrow. We draw functional arrows dotted, to emphasize their distinction from Bachman arrows.

The attribute *color* is a function applying to all *Cars* and has its value in the range of all possible *Colors*; for each car there is exactly one color which it has, and for each color there are 0 or more cars whose color it is. Likewise, the attribute *engine* is a function from *Cars* to *Engines*; for each car there is exactly one engine. Using other terminology, *engine* is a *foreign key* pointing to an entity of *Engines*.

We will use diagrams with functional arrows to clarify some examples. This implies no adherence to the functional data model, nor a fundamental objection to the Bachman arrows used in the book. However, we prefer not to use the Bachman arrow because it may be confusing to people used to a number of other arrows. For example, in many-one relationships, Bachman arrows point from the one to the many direction, which is opposite to the direction of the arrows used between similar boxes in ER diagrams. The direction of Bachman arrows is also opposite to that of the arrows in functional data models as well as to that of arrows in diagrams of functional dependencies. Finally, foreign keys are most naturally represented as arrows in the many-to-one direction, which is again opposite to that of Bachman arrows. All of this suggests that the use of functional arrows is more consistent with other uses of arrows than that of Bachman arrows.

4.2. The didactic strategy used

The order of presentation of the material in the book could be improved. For example, the diagram technique is first used on page 20, for a rather complex metapurpose, the representation of the concepts defined in the introduction. Explanation of the symbols used, however, is given only on page 49. There is no illustration on page 49, and another application of the technique for a metapurpose is given on page 61. Finally, on page 69 the explanation of the symbols is illustrated with a simple example taken from the UoD of inventory control. Clearly, this order of presentation is not optimal.

4.3. Results versus methods

The diagrams in the book come out of the blue as far as the description of the UoD and of the analysis of the UoD is concerned. What is needed in a book on IS development methodology is an illustration of the theory developed in the book with one or two development methods. Such an illustration proceeds by showing how, according to a particular method, a certain UoD is analyzed and a conceptual model of the UoD is designed. What we find instead is an illustration of representation techniques (diagrams, tables) which may be used to present results obtained in applying a method, but are not a method in themselves. So it is illustrated *what* one can get, but not *how* to get it. This is a serious flaw in a book which claims to study IS development methods.

4.4. A modeling mistake.

As noted earlier, the diagram technique used in the book is first used about 30 pages before the meaning of its symbols is explained, and is used for a rather difficult diagram (page 20). Part of this diagram is reproduced in figure 2.

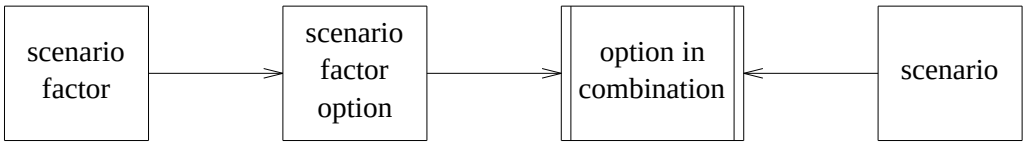

Figure 2.

The vertical bars in a box say that the box is a "cross-reference", which, although not explained in the text, we take to mean a many-many relationship modeled by means of two many-one relationships. Looking at the definitions in the text, we find the following. A *scenario* is a scenario of system development, and is characterized by a number of *factors* such as the scale of the project, the well-definedness of the environment, and the quality of the developer. For each factor there are a number of *options.* For example, the scale can be large or small, the environment can be well-defined or ill-defined, etc. Representing this in a functional diagram, we get that each scenario factor is an attribute of a scenario, and a scenario factor option is a possible value for a scenario attribute (figure 3). Figure 3 is not at all analogous to figure 2 in that it is not obtained from figure 3 by simply reversing the arrows. What has gone wrong in figure 2 can be found out by changing to a less exotic example and replacing the Bachman arrows by the reverse, functional arrows. Figure 4a is the analogon of figure 2 in the domain of cars and figure 4b is the corresponding functional diagram obtained by reversing and labeling the arrows.

Apart from being outlandish and confusing, figure 4b contains three mistakes. The first is that it does not model that *color* is the only possible value for *is_value_of* (*blue*). As far as figure 4b is concerned, *is_value_of* (*blue*) could be *engine*. Correspondingly, figure 2 does not model the fact that each scenario factor option is an option of precisely one particular scenario factor. It just models the fact that a scenario factor option is an option of any one scenario factor.

The second mistake in figure 4b is that it does not model the fact that *color* has possible values

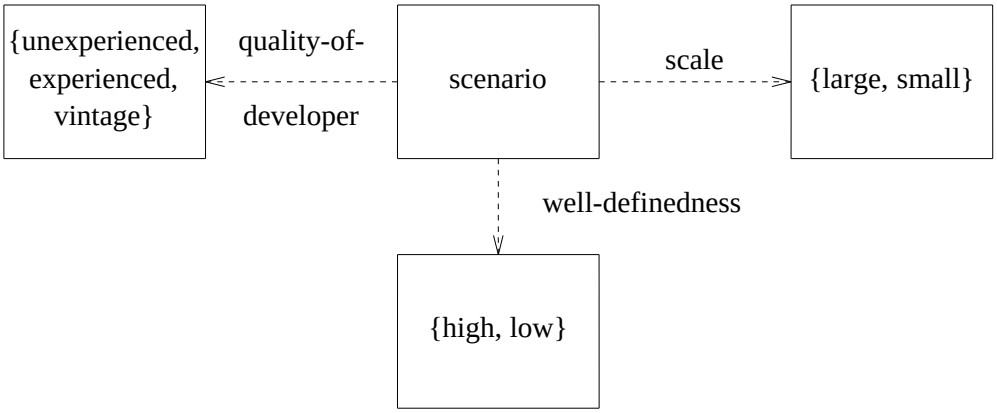

Figure 3.

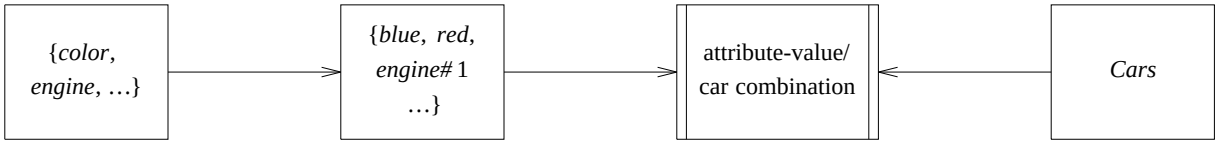

Figure 4a.

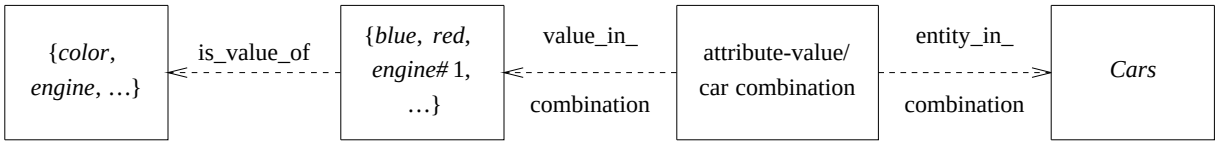

Figure 4b.

blue and *red*, but not *engine# 1* etc. Correspondingly, figure 2 states merely that to each scenario factor belong 0 or more of all possible scenario factor options, rather than that to each scenario factor belongs a particular set of scenario factor options.

The third mistake in figure 4b is that it does not model that each car has exactly one color, one engine, etc. Correspondingly, figure 2 does not model the fact that in each scenario, exactly one option is realized for each scenario factor. For example, the scale of a scenario is either large or small, and the environment is either well- or ill-defined, etc. Instead, figure 2 merely says that there is a many-many relationship between scenario factor options and scenarios, which does allow a scenario to have, for example, both the options large and small.

4.5. The concept of design

On page 2 the *design process* is defined as "the route traveled in reaching the target," where the target is a design product. On page 10 the *design product* is defined as "a set of components which can be given to a builder so that the information system can be constructed according to the design." We take the modality expressed by "can" in this definition to be one of restriction on the design product, not of the possibility of giving. It is not meant that after a design process, the design product may, but need not be given to the builder. Instead, what is meant is that the design product must satisfy certain requirements, such as being complete and consistent, so that a system builder needs nothing but the design product in order to build the system. This definition of the key concept of design

process leaves us in the dark as to where exactly the design process starts. Turning to the diagrams does not help very much. Instead, it causes the concept of design product to be obscured as well.

Figure 1.1 on page 3 of the book illustrates the concepts of design, analysis, and design product. According to it, the design process produces a number of diagrams such as a representation of a terminal, paper report and mass storage. However, these symbols seem to represent implementation components, not components of a design product. For this part of the diagram, we therefore suggest figure 5 as a clarification. In drawing figure 5 we have assumed that figure 1.1 uses the diagram convention that round angles indicate processes and square angles indicate results. This is nowhere indicated in the text, but is significant, as we show below.

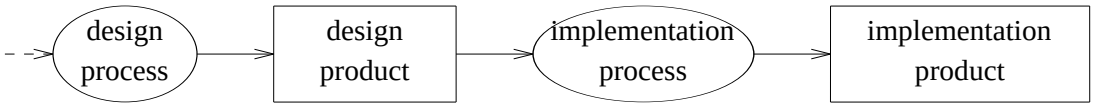

Figure 5.

Moreover, figure 1.1 says that the design product consists of the results of design *and* the results of analysis. This contradicts the statement on page 10 that the design product is given to the builder, for the builder receives (or ought to receive) only the results of design, not the results of analysis.

The start of the design process is not defined on page 2, but on page 10 we find the phrase "the whole design process, namely, the results of the work of the analyst and the designer, ..." Now, the book usually carefully distinguishes a *process* from a *result,* so that this phrase is a bit odd. However, we obtain an indication of the start of the design process from it, for it implies that the design process includes analysis as well as design. This agrees with the rather confusing statement made in figure 1.1 that the design product includes the results of design as well as the results of analysis. On the other hand, it contradicts another part of figure 1.1, which, by the (unexplained) diagram convention for processes and results, states that the design process starts *after* the analysis process. The resulting muddle is not only confusing, but also inconsistent, and we therefore suggest to drop the equation

design product = results of analysis + results of design

and restrict the meaning of "design process" to the process starting from the results of analysis and leading to the results of design, which then form the design product.

5. Conclusions

In conclusion, we repeat that the aim of the book is valuable and that it is not an aim which is easy to reach. We agree with the authors that the aim of the book is to provide a methodology, not a particular method, but we find it unfortunate that the book uses "methodology" to mean method. In their terminology, methodology is nameless, though the meaning they implicitly assign to the word "framework" comes close to the original meaning of "methodology."

We argued that a methodology marks the domain of research by giving a list of definitions of key concepts, and then contains descriptive and possibly also normative statements about methods. Moreover, a book on methodology should contain applications of the theory to particular methods. Applications to particular methods are not found in the book. Although the book contains a list of definitions of key concepts pertaining to IS development methods, and some normative and descriptive statements about methods, all of this is vague and preliminary, and substantiation based on first principles or on examples of system development methods lacks. Therefore, we have to conclude

that in the book, we eat the cake of methodology without it having been served.

Finally, the presentation of concepts and techniques is at places so confusing, inconsistent, or both, that it is not clear at all what is being defined or illustrated. We have presented only a few examples of such deficiencies.

6. References.

Jackson, M. [1983]

System Development, Prentice-Hall.

Kuhn T. [1970]

The Structure of Scientific Revolutions, second edition, University of Chicago Press.

Lakatos, I [1970]

"Falsification and the Methodology of Scientific Research programs," in I. Lakatos & A. Musgrave, *Criticism and the Growth of Knowledge*, Cambridge University Press, 1970, 91-196.

Laudan L. [1977]

Progress and its Problems. Towards a Theory of Scientific Growth, University of California Press.

Olle, T.W., J. Hagelstein, I.G. Macdonald, C. Rolland, H.G. Sol, F.J. Van Assche, A.A. Verrijn-Stuart [1988]

Information Systems Design Methodologies, A Framework for Understanding, Addison-Wesley.

Popper K. [1959]

The Logic of Scientific Discovery, Hutchinson.