A Dynamic Framework for Classifying Information Systems Development Methodologies and Approaches

Author(s): Juhani Iivari, Rudy Hirschheim and Heinz K. Klein


Published by: M.E. Sharpe, Inc.
Stable URL: http://www.jstor.org/stable/40398498
Accessed: 22-08-2014 22:48 UTC

Your use of the JSTOR archive indicates your acceptance of the Terms & Conditions of Use, available at http://www.jstor.org/page/info/about/policies/terms.jsp

JSTOR is a not-for-profit service that helps scholars, researchers, and students discover, use, and build upon a wide range of content in a trusted digital archive. We use information technology and tools to increase productivity and facilitate new forms of scholarship. For more information about JSTOR, please contact support@jstor.org.
A Dynamic Framework for Classifying Information Systems Development Methodologies and Approaches

JUHANI IIVARI, RUDY HIRSCHHEIM, AND HEINZ K. KLEIN

JUHANI IIVARI is Professor of Information Systems in the Department of Information Processing Science at the University of Oulu. Formerly, he held a professorship at the University of Jyväskylä, Finland. He served as the national coordinator of the Finnish Doctorate Programme in Information Systems 1993–1994. Iivari’s research has broadly focused on theoretical foundations of information systems, information systems development methodologies and approaches, including object-orientation, organizational analysis, implementation and acceptance of information systems, the quality of information systems, and knowledge work. Iivari has published in journals such as Communications of the ACM, Data Base, European Journal of Information Systems, Information & Management, Information and Software Technology, Information Systems, Information Systems Journal, Information Systems Research, MIS Quarterly, Omega, Scandinavian Journal of Information Systems, and others.

RUDY HIRSCHHEIM is Professor of Information Systems and holds the Tenneco/Chase International Chair in the C.T. Bauer College of Business at the University of Houston. He has previously been on the faculties of McMaster University (Hamilton, Ontario), the London School of Economics and Political Science (University of London), and Templeton College, Oxford (University of Oxford). He has also worked as a Senior Consultant with the National Computing Centre in Manchester (U.K.). Professor Hirschheim has held visiting Professorial positions at the University of New South Wales (Sydney, Australia), Monash University (Melbourne, Australia), and the University of Warwick (Coventry, UK). Since 1984, he and Richard Boland have been the Consulting Editors of the Wiley Series in Information Systems. He is on the editorial boards of the journals Information and Organization (formerly Accounting, Management and Information Technologies); Information Systems Journal; Journal of Information Technology; and Journal of the Association for Information Systems; and has previously been on the boards of European Journal of Information Systems and MIS Quarterly. He is currently the Vice President for Publications for the Association for Information Systems.

HEINZ K. KLEIN earned his Ph.D. at the University of Munich and in 1998 was awarded an honorary doctorate by the University of Oulu for his academic contributions to the development of IS research in Finland. He is currently Associate Professor of Information Systems in the Fox School of Business and Management at Temple University. He has previously led the IS group in the School of Management of the State University of New York at Binghamton. He has also held research and teaching appointments at major universities in Germany, Canada, Finland, Denmark, New Zealand, and South Africa. He has made contributions to the philosophy of IS research, foundations of IS theory, and methodologies of information systems development, and
has written articles on rationality and the emancipatory ideal in ISD, principles of interpretive field research, and intellectual foundations of alternative approaches to information systems development. His work has been published in journals such as Communications of the ACM, MIS Quarterly, Information Systems Research, Information and Organization (formerly Accounting, Management and Information Technologies), Information Systems Journal, Information Technology and People, Decision Sciences, and others. He has also coauthored or edited several research monographs and international conference proceedings in IS and served on the editorial boards of scholarly journals and the Wiley Series in Information Systems.

ABSTRACT: This paper proposes a four-tiered framework for classifying and understanding the myriad of information systems development methodologies that have been proposed in the literature. The framework is divided into four levels: paradigms, approaches, methodologies, and techniques. This paper primarily focuses on the two intermediate levels: approaches and methodologies. The principal contribution of the framework is in providing a new kind of “deep structure” for better understanding the intellectual core of methodologies and approaches and their interrelationships. It achieves this goal by articulating a parsimonious set of foundational features that are shared by subsets of methodologies and approaches. To illustrate how the framework’s deep structure provides a better understanding of methodologies’ intellectual core, it is applied to eleven examples. The paper also introduces and illustrates a procedure for “accommodating” and “assimilating” new information systems development methodologies in addition to the eleven already discussed. This procedure provides the framework with the necessary flexibility for handling the continuing proliferation of new methodologies.

KEY WORDS AND PHRASES: deep structures, information systems development, systems development methodologies.

Both the practitioner and academic literature continue to devote considerable attention to the issue of how information systems development (ISD) can be improved through the application of new tools, techniques, principles, and methods or methodologies (ISDMs). The unabated proliferation of new methods and tools for developing information systems is clearly visible. Indeed, Jayaratna [55] notes the existence of over 1000 such ISDMs. Avison and Fitzgerald [4] have gone so far as to refer to this as a “methodology jungle,” a seemingly impenetrable maze of competing ideas and notions. This paper takes the view that different methodologies, even though not necessarily widely used, may contain valuable knowledge about ISD. Currently, this collective knowledge remains elusive and difficult to assimilate, because it is so widely dispersed in a plethora of methodologies.

In order to cope with the confusion created by the proliferation of ISDMs, it is desirable to construct an organizing structure that reduces the complexity of the myriad of ISDMs. Is it possible to find an order that not only systematizes different ISDMs, but also captures their spirit and essence? We feel this can be done, but it requires a new way of thinking about ISDMs. Fundamental to this new way of thinking is the
idea that each ISDM is merely one instantiation of a more general abstract class. This class has basic features that are then inherited by all the ISDMs belonging to that class. The class to which kindred ISDMs belong we term an “approach” (ISDA). But ISDAs themselves, we contend, are not the highest level of abstraction. In fact, the most abstract essences of different ISDAs are captured by the characteristics of the paradigm with which they are most closely associated. In summary, our new way of thinking suggests a four-tiered structure. At the top (the fourth and highest level of abstraction) are no more than a handful of paradigms. ISDAs (the third level) inherit fundamental features (assumptions) typically from one or two dominant paradigms. ISDMs (on the second level) instantiate the features of ISDAs and add further detail. Techniques at the bottom (at the first level) are components of specific ISDMs.

The paper’s first purpose is to explain this four-tiered framework in detail. The framework extends the previous distinction between approach and methodology [51] to a four-level structure. The concepts of “approach” and “paradigm” are the central elements of the proposed framework, and therefore we focus our attention on the “essences” (or essential features) of ISDMs rather than their “accidental features.” (This issue is taken up in the “Conclusions” of the paper). This four-tiered framework provides a context for the remaining part of the paper, which focuses on the two intermediate levels and especially on the relationship between ISDMs and ISDAs. The second purpose of the paper is to illustrate the organizing and simplifying power of the concept of “approach.” This is done by applying the framework to a representative cross section of ISDAs currently known through their publication in the scholarly literature. This cross section reflects a systematic selection of eleven ISDAs from multiple paradigms. In this way, the paper demonstrates that hundreds of ISDMs can be reduced to a few dozen ISDAs characterized by their essential features. Lastly, because of the continued evolution of the ISD area, our proposed framework must be dynamic. Thus the paper’s third purpose is to provide a procedure for dealing with newly emerging ISDAs. Taken in total, we believe the paper contributes a conceptual structure that organizes the field of ISD in a more comprehensible way.

The most fundamental implication of the paper is to shift the discussion of alternative ISDMs and tools to a higher level at which the IS community’s thinking can benefit from comparing and contrasting features at the level of ISDAs rather than ISDMs. In this way the framework serves as an organizing and simplifying vehicle. It also allows us to capture the knowledge and insights embedded in the existing ISDM base. In addition to these intellectual contributions, the paper also has practical implications because, at best, only a few ISDMs are likely to be mastered in practice by any one individual. The simplifying concepts of an ISDA may allow practitioners to widen their “methodology” repertoire. An additional point is that when adopting a particular ISDM, the methodology user consciously or unconsciously adopts the underlying assumptions of the ISDM. This can be expected to have implications for the system that is ultimately developed (cf. [7]). Therefore the choice of which ISDMs to study or use is important and the assumption analysis conducted in this paper can be an effective analytical tool to help make such choices. The proposed framework brings these assumptions to the forefront. From this angle the paper adds more detail to our
earlier work on mapping the terrain of the complex literature on IS development (cf. [40, 41, 42, 43, 47, 51, 69]).

A further implication of the four-tiered framework concerns the practical use of ISD methodologies. Existing research on ISDM use in practice [15, 34, 113] has consistently found that ISDMs, as far as they are used, are applied adaptively, customized to fit the organization and project in question. This can be regarded as an argument for the need for methodology engineering [63]. The paper also contributes to this discussion by suggesting that the concept of ISDA supports flexible generation and adaptation of ISDMs. The examples of ISDMs as instantiations of ISDAs help one to understand the underlying assumptions of methodologies in methodology engineering. In this context, the four-tiered framework specifically helps to sensitize a methodology architect to the issue of incompatibility when choosing features from different approaches as building blocks in the assembling of a customized methodology.

The contributions of the paper can be summarized by five points, which are of relevance to academic researchers and practitioners alike. First, and as its major contribution, the paper develops and illustrates the idea of the four-tiered structure as a dynamic framework for classifying information systems development methodologies. The structure is based on the idea of distinguishing the essential features of methodologies from their less essential ones by associating them with approaches and paradigms. Second, the developed framework helps make sense of the "methodology jungle," pointing to the existence and significance of methodologies that do not belong to the mainstream that happens to be popular at any given time. This supports pluralism in IS research and practice, that is, the belief that there is always more than just one reasonable way of developing information systems. Third, the framework contributes to methodology comparison by pointing out similarities and differences between methodologies. In particular, it allows recognizing and modeling genealogical dependencies of methodologies. Fourth, the framework supports an alternative and complementary view of methodology engineering. In this complementary view methodologies are generated as instantiations of existing ISD approaches rather than just combinations of existing methodology fragments. Also, using our framework would sensitize method engineers to the underlying assumptions of ISDMs, which render methodology fragments more or less compatible with each other. Finally, the four-tier framework forms a condensed knowledge representation schema of ISD process knowledge. This is likely to help practitioners widen their methodology repertoire, thereby supporting more knowledgeable and reflective practice of ISD.

The organization of the paper is as follows. The following section reviews the literature upon which the arguments of this paper build. Next we explain the four-tiered framework. It also illustrates the framework by applying it to a systematic selection of ISDAs representative of the spectrum of currently known approaches with some of their associated methodologies. Then we introduce the dynamics of the classification structure by describing how to accommodate and assimilate a new ISDM. Finally, we conclude with several notable implications of our work, partly extending the results of previous work and partly pointing the direction for future research.
Review of Related Research

WE DIVIDE THE REVIEW OF THE RELEVANT RESEARCH LITERATURE into two streams. The first places the paper in the context of the antecedent discussion of the role and significance of methodologies for IS development in general. It should be noted that this antecedent discussion is not without controversy. Knowing where we stand on this controversy will help us to interpret the purpose of the paper and the development of arguments. The second stream is important because it summarizes the intellectual and conceptual bases upon which this paper directly builds.

Background and Motivation

There are two countervailing trends in the current discussion about ISDMs, and our paper can be viewed as a contribution to both. The first trend is that the usefulness of ISDMs is being questioned because of their lack of practicality. The main criticism is that for a number of reasons they do not seem to have accomplished their intended purpose: to organize and guide the work of systems developers [7, 25]. Even though there is a dearth of empirical research on the actual use of ISDMs, existing evidence (e.g., [15, 34, 95]) suggests that their use is limited in practice, and as far as they are used, they are not literally applied (cf. [107]). Chatzoglou and Macaulay [15], for example, report that nearly half of the projects (47 percent) did not use an ISDM in their survey of 72 projects within the U.K., while another British survey [34] reports a significantly different figure (18 percent) for the non-use of ISDMs. Hardy et al. [34] also report that 38 percent of ISDMs were developed in-house and were customized in 88 percent of cases. Similarly, Wynekoop and Russo [113] report the findings of a survey of over 100 organizations that indicated that 65 percent of organizations had developed their own ISDM in-house and 89 percent of the respondents believed that formal ISDMs should be adapted on a project-by-project basis. Mathiassen [72] went so far as to question the practicality of ISDMs in guiding the work of seasoned practitioners, suggesting that ISDMs are primarily intended for beginners as the primary vehicle by which they are initiated into the field (cf. [1]). More moderately, Unhelkar and Mamdapur [103] propose a metaphor of a “road map” for ISDMs, suggesting that an ISDM may not be able to recognize all situational factors (e.g., roadblocks) and is more useful for a foreigner (beginner) than for seasoned practitioners [38].

It is our contention that before proceeding to the empirical world one should be aware of the conceptual distinction between an “ISD methodology” and an “ISD approach.” This is so because ISDM use may take place at the level of the guiding principles and fundamental concepts of approaches—possibly supported by related techniques—rather than at the level of specific ISDMs. One should also pay attention to alternative roles of ISDMs and ISDAs in IS development. Even though not used literally, they may nevertheless become a convenient knowledge representation form, serve as ideals of a “rational” design process [86], provide concepts and metaphors for expert communications, and form useful vehicles of organizational and individual learning as the frames of reference upon which new practical experience may be
contrasted [17]. Hence, despite the noted concerns articulated above, we conclude that ISDMs and ISDAs deserve continued attention even though the focus of this attention may be shifting to new aspects and roles of ISDMs and ISDAs. Therefore, it is important to ask which ISDMs and ISDAs should receive attention in research, teaching, and the discourse of practitioners. Our paper can help shed light on this question.

The second trend, paradoxically, appears to be the exact opposite of the first. There is a resurgence of interest in new ISDMs and ISDAs based on several patterns in the IS landscape: the evolution of information technology platforms from centralized mainframe environments toward distributed client-server architectures embracing the Internet and intranets; changes in user interface technology from character-based to graphical user interfaces, multimedia and the World Wide Web; changes in new applications from transaction processing systems toward systems supporting collaborative work (CSCW); and the use of information technology as an enabler of business process reengineering and redesign (BPR). These changes, coupled with changes in organizations and their operating environments—such as the growth of the network and virtual organizations, internationalization and globalization of organizations, intensified global competition, changes in values such as customer orientation (service quality), and Quality of Working Life—have imposed new challenges for and demands on information systems development. These challenges and demands suggest the need for openness to new, emerging ISDMs and ISDAs.

Second, the recent interest in object-orientation has created a new wave of ISDMs. Since the late 1980s dozens of object-oriented ISDMs have been published (see [48, 74, 105] for reviews). The object-oriented research community has responded to the proliferation by initiating two unification efforts: Unified Modeling Language (UML) [102] and OPEN [84]. Neither of these takes place at the level of approaches, that is, as attempts to define the common core of the object-oriented approach. On the contrary, in principle they continue to proliferate the set of object-oriented methodologies by suggesting additional OO methodologies.2

Third, Computer Aided Systems/Software Engineering (CASE) environments with their ISDM companions [106] have brought ISDMs to the forefront. Most current CASE tools support specific standard ISDMs, such as Modern Structured Analysis or the Object Modeling Technique. The tight coupling between ISDMs and CASE tools was feared to stifle the continuing evolution of ISDMs [13]. To alleviate this problem the latest CASE research has focused on more flexible CASE environments, leading to the idea of proposing “methodology engineering” as a discipline for designing, constructing, and adapting ISDMs [35, 63]. The need for methodology engineering is also supported by the observation that many organizations prefer in-house ISDMs over standard ones and the demand for customizing ISDMs continues to grow.

Fourth, there has been considerable interest in the quality of IS and software products (e.g., [53]). It is generally believed that higher quality is achieved through a more disciplined development process, leading to considerable research and industry interest in software process maturity and improvement [46, 64, 87]. ISDMs can be seen to have a role in this endeavor toward a more disciplined development process.
Our paper contributes to this second trend—that is, reflecting the resurgence of interest in ISDMs—by suggesting how new ISDMs could be constructed and adapted, through a process based on the instantiation of ISDAs. An important aspect of the process is the focus on fundamental principles and concepts of the ISDAs and on their underlying assumptions (paradigms).

The fundamental tenet of the paper is that one should move beyond strict ISDMs to focus on more general ISD approaches, which may comprise a number of more specific ISDMs. These general ISDAs may be conceived of as prototypical classes that share a number of common features with their member (instance) ISDMs. A comparison of object-oriented methodologies in [44] showed that large parts of new methodologies are taken from other methodologies and techniques. This illustrates that specific methodologies may not form the most fruitful units of comparative analysis. Instead of focusing on methodologies per se, one should look at more general principles (such as functional decomposition, incremental entity modeling, encapsulation, etc.) that underline specific methodologies as similarities (e.g., functional decomposition in many structured methodologies and incremental entity modeling in both structured and object-oriented ISDMs) and differences (e.g., encapsulation between structured and object-oriented approaches).

Analysis and Comparison of ISD Methodologies

The previous work on ISDM comparison is largely based on conceptual analysis (e.g., [19, 20, 21, 70, 80]) and more recently formal metamodeling (e.g., [44, 92]). Historically, the CRIS (Comparative Review of Information System Development methodologies) conferences [80, 81, 82, 83, 105] form one of the most ambitious attempts to compare ISDMs. The CRIS project attempted to take stock of the then current methodologies, to apply them to a common case of an IFIP working conference and to conduct a feature analysis with the purpose of identifying commonalties and differences among methodologies.

A second notable attempt to compare information systems and software development methodologies systematically is suggested in Song and Osterweil [99]. They propose a set of “design issues” to model the issues that a methodology or its component addresses, a “type hierarchy” that at the highest level identifies “concept,” “artifact,” “representation,” and “action” to model the internal characteristics of a methodology and its components, and a set of metalanguage formalisms to model a method. This framework is intended for a detailed and formal analysis of methods and their components (techniques). Even though it is not directly applicable to approaches as we define them, it nevertheless has similarities to the framework we suggest to model ISD approaches in the following section (cf. note 3).

This paper goes beyond the earlier analyses of ISDMs. Even though useful, these earlier analyses have mostly taken place at the level of specific ISDMs, focusing on explicit and concrete features without paying much attention to the basic principles and assumptions underlying the ISDMs. The major focus of the present paper lies at
the level of approaches, addressing more fundamental aspects, that is, essences, of ISDAs and ISDMs.

The Four-Tiered Framework

Whereas the terms "technique," "methodology," "approach," and "paradigm" are used extensively in the literature, much of this use is inconsistent. In the following, we shall address this issue by suggesting definitions that interrelate these terms by assigning them to different levels of abstraction. The highest level of abstraction is that which connects research on ISDMs to alternative paradigms that are shared by different research communities. The lowest level of abstraction consists of specific techniques as components of ISDMs. We shall argue the usefulness of an intermediate level of abstraction that organizes ISDMs into clusters sharing a set of common principles and fundamental concepts. The efficacy of this four-tiered conceptual structure is demonstrated through its application to a number of specific ISDAs and ISDMs.

The Relationship Between Methodologies and Approaches

As noted in the beginning of this paper, ISDMs have proliferated in great numbers, leading to an increasingly dense "methodology jungle" and concomitant confusion. Part of this confusion stems from the notion of "method" or "methodology" and "approach." In this paper, we explore these notions, concluding that there is value in moving beyond ISDMs to focus on more general ISDAs.

An ISD methodology (ISDM) has been interpreted as "an organized collection of concepts, methods (or techniques), beliefs, values, and normative principles supported by material resources" [43]. In Iivari et al. [51] an ISDM was defined as, "a codified set of goal-oriented 'procedures' which are intended to guide the work and cooperation of the various parties (stakeholders) involved in the building of an information systems application. Typically, these procedures are supported by a set of preferred techniques and tools, and guiding principles." A technique consists of a well-defined sequence of elementary operations that more or less guarantee the achievement of certain outcomes if executed correctly.

An ISD approach (ISDA), on the other hand, is interpreted as a class of specific ISDAs that share a number of common features. More specifically, we define an ISDA as a set of related features that drive interpretations and actions in information systems development. While there could be more discussion on the best set of features, for the time being we propose the following four: (1) goals, (2) guiding principles and beliefs, (3) fundamental concepts, and (4) principles for the ISD process. The goal specifies the general purpose of the ISDA. Guiding principles and beliefs form the common "philosophy" (cf. [4]) of the ISDA, which ensures that its ISDM instances form coherent holes. Fundamental concepts largely define the nature of an IS implicit in the approach—the focus and unit of analysis in ISD. Principles of the ISD process express essential aspects of the ISD process in the ISDA.
Structured Analysis/Structured Design (SA/SD) [115] can be used to illustrate the characterizing features of ISDAs. “To provide maintainable software” is an example of a goal related to SA/SD; “the design of modules with high cohesion and weak coupling” is one of its guiding principles; a “dataflow” and “cohesion” are two of its fundamental concepts; and “to apply a situation-dependent process model” is one of the principles of the ISD process.

To illustrate the distinction between ISDMs and ISDAs, one can identify a number of ISDAs with their ISDM instances, such as the Structured approach, comprising, for example, the DeMarco [22] and Yourdon [115] methodologies; the Information Modeling approach, including NIAM [77, 104] and IE [71]; the Object-Oriented (OO) approach, including the Booch methodology [10] and Responsibility-Driven Design [112]; the Sociotechnical Design approach comprising ETHICS [75] and Pava’s methodology [88, 89]; and so on. To take the example further, consider object-oriented methodologies as a class of “the OO approach to ISD.” It is well known that there are a number of object-oriented ISD methodologies (see [24, 48, 74, 108] for comparisons) that share a number of features, such as encapsulation, information hiding, inheritance, and communication through messages. More generally, an ISDA may include zero, one, or more concrete ISDMs as its instances. As an example of the first case (i.e., zero ISDM instances), consider the interactionist approach (see below) which has most of the features of an approach but to our knowledge has not been developed into any specific ISDM. On the other hand, the OO approach comprises a number of ISDM instances. Thus the concept of an ISDA makes it meaningful to compare various approaches that may be in quite different stages of their development in terms of the number of their ISDM instances.

Underlying Paradigmatic Assumptions

Our earlier work identified a set of philosophical (paradigmatic) assumptions and beliefs underlying every ISDA and ISDM that allow ISDAs to be grouped into a number of paradigmatic positions. The paradigmatic assumptions can be divided into four categories. A basic ontology (what is assumed to be the nature of IS) and epistemology (what human knowledge is and how it can be acquired). The research methodology states the preferred research methods for continuing the improvement of the ISDA as well as how the ISDA was developed and justified in the first place. The ethics specify the values that ought to guide IS research (bearing in mind that the ISDA was a product of such research). These four fundamental concepts describe the kinds of beliefs that typically characterize an academic community with a shared paradigm. This means that ISDAs are ultimately linked to paradigms through the beliefs held by the creator of the ISDA when developing it. It is generally accepted that such beliefs concern the nature of reality (i.e., the ontological positions of realism or constructivism); how knowledge is acquired (i.e., the epistemological alternatives of positivism or antipositivism); and the values that should guide research investigation (i.e., the ethical positions of neutralism or criticalism). More specifically, epistemology embraces two aspects: (1) the fundamental nature of knowledge
(whether knowledge consists of reality descriptions or social constructions); and (2) the actual methods of inquiry (the specific methods of data collection, for example, either idiographic or nomothetic). (See [47] for further details.)

Burrell and Morgan’s [14] well-known framework elaborates these notions, combining them into four alternative paradigms of sociology and organizational research: functionalism, interpretivism, radical structuralism, and radical humanism. Extending the notion further, Hirschheim and Klein [40] demonstrated that the four paradigms of organizational analysis identified in Burrell and Morgan also existed in the literature on ISD. This suggests that the paradigmatic features of functionalism, social relativism (interpretivism), radical structuralism, and neohumanism (radical humanism) can be used to classify ISDAs.

The Framework

This line of reasoning leads to a four-tiered framework that allows us to organize the “methodology jungle” of hundreds of ISDMs into a comprehensible number of ISDAs (of the order of perhaps a few dozen) and a handful of paradigms (e.g., four suggested by [40] following [14]). The framework includes an inheritance structure in which each ISDA inherits the paradigmatic assumptions of the paradigm it represents. Similarly, each ISDM inherits the features of the ISDA (goals, guiding principles and beliefs, fundamental concepts, and principles of the ISD process), and the paradigmatic assumptions of the paradigm it represents indirectly through the ISDA to which it belongs.

Figure 1 illustrates our four-tiered framework. At the bottom of the Figure we identify examples of specific ISDMs as composites (the “is-part-of” relationships are depicted as diamonds) of specific techniques. Techniques have their detailed concepts and notations. Methodologies more or less successfully integrate these techniques into more coherent wholes. At the detailed level of techniques, methodologies may include features that are accidental (such as notations) or idiosyncratic to the specific methodology (e.g., the concept of qualifier in [94] among the OO methods). The middle of Figure 1 depicts the eleven ISDAs to be discussed and analyzed in the next section. The ISDMs within each ISDA are assumed to share the specific goals, guiding principles, fundamental concepts, and principles of their respective approach (cf. Tables 2 and 3). Finally, Figure 1 notes the four paradigms at the top.

In order to illustrate the intended interpretation of Figure 1, the following example traces the “use case” from the middle of the last line upward through the four levels. “Use case” is an instance of the class “ISD techniques,” and it is part of “OOSE.” OOSE, on the other hand, is an instance of the class “ISD methodologies,” and more specifically of the class “OO,” which itself is an instance of the class “ISD approaches.” “OO” as an ISD approach is an instance of the class “functionalism,” which is an instance of four “ISD paradigms.” Or more briefly, the ISD technique “use case” is part of the ISD methodology “OOSE,” which is an instance of the OO approach belonging to the functionalist paradigm.
Figure 1. The Hierarchy of ISD Paradigms, Approaches, Methodologies, and Techniques (See Table 1 for abbreviations.)
Table 1. Abbreviations for Figure 1 (in Order of Occurrence)

**Approaches**

<table>
<thead>
<tr>
<th>Abbreviation</th>
<th>Interactionist</th>
<th>Methodology</th>
</tr>
</thead>
<tbody>
<tr>
<td>SA/SD</td>
<td>Structured</td>
<td>Interact.</td>
</tr>
<tr>
<td>IM</td>
<td>Information Modelling</td>
<td>SA-based</td>
</tr>
<tr>
<td>DSS</td>
<td>Decision Support Systems</td>
<td>SSM</td>
</tr>
<tr>
<td>STD</td>
<td>Socio-Technical Design</td>
<td>PWP</td>
</tr>
<tr>
<td>Infol.</td>
<td>Infological</td>
<td>TU-ist</td>
</tr>
<tr>
<td>OO</td>
<td>Object-Oriented</td>
<td></td>
</tr>
</tbody>
</table>

**Methodsologies**

<table>
<thead>
<tr>
<th>Methodology</th>
<th>ISAC</th>
<th>Analysis of Changes</th>
</tr>
</thead>
<tbody>
<tr>
<td>SADT</td>
<td>Structured Analysis and Design Technique</td>
<td>Object-Oriented Analysis and Design</td>
</tr>
<tr>
<td>MSA</td>
<td>Modern Structured Analysis</td>
<td>OOAD</td>
</tr>
<tr>
<td>IE</td>
<td>Information Engineering</td>
<td>Object Oriented Software Engineering</td>
</tr>
<tr>
<td>NIAM</td>
<td>Nijssen’s Information Analysis Method</td>
<td>OOSE</td>
</tr>
<tr>
<td>K&amp;SM</td>
<td>Keen and Scott Morton's (1978) Methodology</td>
<td>SAMPO</td>
</tr>
<tr>
<td>S&amp;C</td>
<td>Sprague and Carlson's (1982) methodology</td>
<td>W&amp;F</td>
</tr>
<tr>
<td>ETHICS</td>
<td>Effective Technical and Human Implementation of Computer-based Systems methodology</td>
<td>SSM81</td>
</tr>
<tr>
<td>Pava</td>
<td>Pava’s (1983) methodology</td>
<td>Wilson84</td>
</tr>
</tbody>
</table>

**Techniques**

<table>
<thead>
<tr>
<th>Technique</th>
<th>OM</th>
<th>Object Model</th>
</tr>
</thead>
<tbody>
<tr>
<td>DFD</td>
<td>Data Flow Diagram</td>
<td></td>
</tr>
<tr>
<td>ERD</td>
<td>Entity Relationship Diagram</td>
<td>ID</td>
</tr>
<tr>
<td>STD</td>
<td>State Transition Diagram</td>
<td>CM</td>
</tr>
<tr>
<td>IFD</td>
<td>Information Flow Diagram</td>
<td>CATWOE</td>
</tr>
<tr>
<td>ISD</td>
<td>Information Structure Diagram</td>
<td></td>
</tr>
<tr>
<td>IOA</td>
<td>Input/Output Analysis diagram</td>
<td></td>
</tr>
<tr>
<td>VT</td>
<td>Variance Table</td>
<td></td>
</tr>
<tr>
<td>JSQ</td>
<td>Job Satisfaction Questionnaire</td>
<td>—</td>
</tr>
<tr>
<td>OD</td>
<td>Object Diagram</td>
<td>MC</td>
</tr>
<tr>
<td>SC</td>
<td>Service Chart</td>
<td>—</td>
</tr>
<tr>
<td>—</td>
<td>Use case</td>
<td>—</td>
</tr>
</tbody>
</table>

In the following we shall concentrate on the relationship between the two intermediate levels of Figure 1, illustrating the relationship by analyzing a representative set of eleven ISDAs. The relationship between the top two levels—paradigms and approaches—has been analyzed in a number of previous publications [40, 41, 42, 43, 47, 50, 51], and the mappings between the four paradigms and the eleven ISDAs is based on this extensive prior research. “Methodology engineering” [63] has particu-
larly focused on the relationship between methodologies and techniques. Although we shall not discuss this relationship, we point out that the proposed framework has important implications with regard to methodology engineering (see the section entitled "Methodology Engineering").

Illustration of Eleven ISD Approaches

Tables 2 and 3 summarize eleven ISDAs that are used to illustrate the four-tiered framework. The rationale behind selecting these eleven ISDAs was to have a representative sample of existing ISDAs. Their selection is based on the earlier analyses of the paradigmatic assumptions of a number of "ISD approaches" [41, 47, 51]. From our earlier analyses, we selected six ISDAs that are considered to be fairly representative of the dominant functionalist paradigm (see Table 2). Selecting ISDAs representative of non-functionalist paradigms turns out to be more difficult as they simply are not as plentiful as their functionalist counterparts. Nevertheless, we have chosen five non-functionalist ISDAs that we feel represent the most notable efforts to challenge the functionalist paradigmatic assumptions. Their paradigmatic assumptions were analyzed in detail in [51], and the approaches are described in some detail in [39] (see Table 3).

Table 2 summarizes the six functionalist ISDAs, while Table 3 summarizes the five non-functionalist ISDAs in terms of their goals, guiding principles and beliefs, fundamental concepts, and principles of the ISD. Because of space limitations, the explanations of Tables 2 and 3 are quite condensed and are presented in Appendix 1. (Again, see [39, 47, 51] for more details.)

Maintaining the Framework

THIS SECTION ADDRESSES THE ISSUE of how to deal with the continuing proliferation of ISDMs or ISDAs. For this purpose, one needs to keep in mind that a new ISDA can be defined in two ways: First, it can be abstracted from the details of methodology instances. Second, some ISDAs might have been developed through formulating their own features, as was the case with the Interactionist approach. These approaches might not have any concrete methodology instances. The section first outlines a procedure for the dynamic maintenance of the framework. Next, the procedure is illustrated by inserting OMT and Multiview in the classification structure of Figure 1.

The Procedure for Maintaining the Classification Structure

For our proposed framework to continue to be useful in the future, one must be able to classify whether an ISDM/ISDA to be "inserted" in the classification structure is an approach or a methodology. Once determined, its features must be identified and compared with those already recognized by the existing classification derived from our four-tiered framework. In order to simplify the situation let us focus first on the
<table>
<thead>
<tr>
<th>Structured Approach</th>
<th>Information Modeling</th>
<th>Decision Support Systems</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Goal:</strong> To provide an approach that helps to</td>
<td><strong>Goal:</strong> To provide an approach for enterprise-wide development of information systems</td>
<td><strong>Goal:</strong> To provide an approach for developing systems to support in particular, semi-</td>
</tr>
<tr>
<td>produce high quality (reliable and maintainable)</td>
<td>(databases) that enables coordinated development of integrated application systems and</td>
<td>structured decision making</td>
</tr>
<tr>
<td>software in a productive way</td>
<td>their long-term evolution</td>
<td></td>
</tr>
<tr>
<td><strong>Guiding Principles and Beliefs:</strong> Separation of</td>
<td><strong>Guiding Principles and Beliefs:</strong> Data form a stable basis for information systems;</td>
<td><strong>Guiding Principles and Beliefs:</strong> The purpose of a DSS is to support rather than replace a</td>
</tr>
<tr>
<td>the essential model from the implementation model;</td>
<td>Separation of conceptual and internal schemas/models; The conceptual schema is a theory of</td>
<td>decision; Use of a DSS is interactive; DSS use implies learning; A DSS evolves</td>
</tr>
<tr>
<td>Careful documentation to make the development process</td>
<td>the Universe of Discourse; The conceptual schema forms the core model for an IS;</td>
<td></td>
</tr>
<tr>
<td>visible; Graphic notations; Top-down partitionable</td>
<td>Applications are built on top of the conceptual schema; IS development should be based</td>
<td></td>
</tr>
<tr>
<td>transformation/process models to hide complexity;</td>
<td>on an engineering like rigorous methodology</td>
<td></td>
</tr>
<tr>
<td>Unambiguous, minimally redundant, graphic specification;</td>
<td><strong>Fundamental Concepts:</strong> Universe of Discourse; Information/database; Conceptual schema;</td>
<td></td>
</tr>
<tr>
<td>Balancing of models with high cohesion and weak</td>
<td>Internal schema; External schema; Entity; Attribute; Relationship</td>
<td></td>
</tr>
<tr>
<td>coupling</td>
<td><strong>Principles:</strong> Incremental conceptual schema design; View integration</td>
<td></td>
</tr>
<tr>
<td><strong>Fundamental Concepts:</strong> Essential model vs.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>implementation model; Transformation (process); Data</td>
<td></td>
<td></td>
</tr>
<tr>
<td>flow; Data store; Terminator; Module; Cohesion;</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Coupling</td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Principles:</strong> A step by step process at the detailed</td>
<td></td>
<td></td>
</tr>
<tr>
<td>level of analysis and design activities; Situation</td>
<td></td>
<td></td>
</tr>
<tr>
<td>dependent at the &quot;strategic&quot; level (water-fall,</td>
<td></td>
<td></td>
</tr>
<tr>
<td>prototyping, concurrent)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Sociotechnical Approach</td>
<td>Object-Oriented Approach</td>
<td>Infological Approach</td>
</tr>
<tr>
<td>------------------------</td>
<td>-------------------------</td>
<td>----------------------</td>
</tr>
<tr>
<td><strong>Goal</strong>: To provide an approach for IS development that enables future users to play a major part in the design of the system, to cater to job satisfaction objectives in addition to more technical and operational objectives, and to ensure that the new system is surrounded by a compatible well-functioning organizational system</td>
<td><strong>Goal</strong>: To provide an approach that helps to ensure that the products are delivered to the user on time and within budget, that the products meet user requirements, that the user requests to modify the system and/or fix bugs are responded to in a timely fashion, that increasingly sophisticated products are offered to keep a competitive edge, that changes in standards and delivery technology are kept up and the project team feels motivated and successful</td>
<td><strong>Goal</strong>: To provide an approach that helps to ensure that really needed information systems are developed, systems that give a positive contribution to the organization, that users understand the system, and that the system is easily maintainable, portable, and efficient</td>
</tr>
<tr>
<td><strong>Guiding Principles and Beliefs</strong>: Self-design of a work system; Minimal critical specification; Open-ended design process; Fit between the social and technical subsystems; Joint optimization; Redundant functions</td>
<td><strong>Guiding Principles and Beliefs</strong>: Seamless analysis, design and implementation</td>
<td><strong>Guiding Principles and Beliefs</strong>: Distinction between the infological and datalogical problems; An information system is a model of the object system; The infological problem should be solved before the datalogical problem; User should control the development (especially at the infological level); Levels of modelling and an integrated system of description languages allow effective user participation</td>
</tr>
<tr>
<td><strong>Fundamental Concepts</strong>: Technical system: Social system: Variance; Unit operation; Technical needs; Social needs (job satisfaction)</td>
<td><strong>Fundamental Concepts</strong>: Problem domain vs. implementation domain; Object and class; Encapsulation; Information (implementation) hiding; Inheritance; Polymorphism; Communication between objects</td>
<td><strong>Fundamental Concepts</strong>: Infological problem vs. datalogical problem; Object system; Activity; Material flow; Information flow; Information/message set; Precedence relation; File; File consolidation; Process consolidation</td>
</tr>
<tr>
<td><strong>Principles</strong>: User participation; Socio-technical design; Evolution</td>
<td><strong>Principles</strong>: Iterative and incremental development; Reuse</td>
<td><strong>Principles</strong>: Information analysis based on top-down precedence and component analysis; Design based on bottom-up</td>
</tr>
<tr>
<td>Interactionist approach</td>
<td>SA-Based Approach</td>
<td>SSM Approach</td>
</tr>
<tr>
<td>---------------------------------------------------------------------------------------</td>
<td>-----------------------------------------------------------------------------------</td>
<td>--------------------------------------------------------------------------------</td>
</tr>
<tr>
<td><strong>Goal:</strong> To shed light on the social issues surrounding organizational change and implementation of information systems</td>
<td><strong>Goal:</strong> To provide a methodology for modelling communicative action in organizations, especially speech acts of changes: creating, maintaining, reporting, modifying and terminating organizational commitments</td>
<td><strong>Goal:</strong> To provide a learning methodology to support debate on desirable and feasible changes</td>
</tr>
<tr>
<td><strong>Guiding Principles and Beliefs:</strong> An information system is a social object with social meanings serving different interests; The infrastructure supporting the focal system is critical; Control over the infrastructure is a political process; Commitments of the past constrain the future; IS development is social action of negotiation</td>
<td><strong>Guiding Principles and Beliefs:</strong> Use of notional system models called “human activity systems” to illuminate different Weltanschauungen which may be applied to any social system; An information system is a system to support the truly relevant human activity system</td>
<td><strong>Guiding Principles and Beliefs:</strong> Design of computer support is design of conditions for work; Craftsmanship as the ideal of work; A collective resource approach based on trade union participation</td>
</tr>
<tr>
<td><strong>Fundamental Concepts:</strong> Information systems as institutions; Social use of information systems; Complex and overlapping negotiation context; Nonneutrality of IS resources</td>
<td><strong>Fundamental Concepts:</strong> Speech acts; Illocutionary points; Propositional content; Discourses/conversations</td>
<td><strong>Fundamental Concepts:</strong> Weltanschauung; Human Activity Systems; Root definition; Relevant system</td>
</tr>
<tr>
<td><strong>Principles:</strong> N/A</td>
<td><strong>Principles:</strong> Discourse/conversation analysis; Analysis of the propositional content</td>
<td><strong>Principles:</strong> Stream of cultural analysis; Stream of logic-based analysis</td>
</tr>
</tbody>
</table>

Table 3. Summary of the Five Nonfunctionalist ISDAs
insertion of a specific ISDM. The insertion of an ISDA can be transformed to an insertion of an ISDM, as will be explained later.

One can, in principle, distinguish two situations when adding a new methodology in the classification structure: (1) An ISDM so far not classified by the framework is entirely composed of features that are part and parcel of the approaches already dealt with in the four-tiered framework. (2) an ISDM so far not classified by the framework comprises features that are entirely new in the sense that they are not part and parcel of the approaches already dealt with in the four-tiered framework.

The gist in both situations is to distinguish between a single- and multiple-inheritance hierarchy. In the simplest case of single inheritance, the ISDM to be assimilated can be subsumed under one and only one existing class (ISDA). In such a case, the ISDM is simply added to an existing class as its new instance. A more complex situation of single inheritance is based on the generalization of an existing class. In this case a new generalized class is formed that includes a subset of features of an existing class. The ISDM to be inserted is “assimilated” by inserting it as an instance of the generalized class. Multiple inheritance requires the creation of a new “class” corresponding to the methodology to be inserted. This new class is an intersection class that inherits from the superclasses the features of the ISDM to be assimilated. The superclasses are either existing classes or their generalizations. In order to “accommodate” situation 2, one needs to keep in mind that the procedure inevitably implies adding a new approach. Therefore, if new features are discovered at the level of a methodology, then a new ISDA has to be defined, thus adding a new class into the second (i.e., approach) level of the framework. Of course the candidate ISDM may also comprise features shared by other approaches. The new class can be accommodated in the classification structure either using single hierarchy or multiple hierarchy. Single hierarchy in this case allows that existing classes are not only generalized but that some of their features may also be modified. Accommodation of OMT in the following section illustrates this situation. In the case of multiple hierarchy the intersection class containing the features to be inherited is the repository of the new features of the methodology to be accommodated. Accommodation of Multiview in the section after that illustrates this situation.

Appendix 2 includes a detailed Assimilation–Accommodation (A–A) Procedure for inserting a new ISDM into the classification structure of existing ISDAs. Figure 2 illustrates the logic of the procedure by showing ten different cases. Let us assume in all the cases that the initial situation of the classification structure is that described in the upper left corner. The ovals describe the classes (ISDAs) and the arrows between them describe the inheritance hierarchy between them. The features of each class are interpreted as propositions, depicted by capital letters. For the sake of clarity, inherited features are placed in parentheses.

Assimilation of a new ISDM presupposes that one abstracts the essential features of the methodology in question. They are also described as propositions. The procedure also assumes selection, based on human judgment, of one or more candidate classes, with which the methodology could be associated. The first case in Figure 2 illustrates a situation where one of the existing approaches, (ABD)EF, has exactly the same set
Figure 2. Cases of Assimilating and Accommodating an ISDM in the Classification Structure
of features as the ISDM (M1) to be inserted. In this case M1 can be assimilated as a member of this class. In the second case there is one new feature (P) in M2 that is not in the candidate class (ABD)EF, leading to the creation of a new subclass for the candidate class. This new class includes P as its noninherited feature, and the methodology M2 is accommodated as its member. The third case shows a situation where the ISDM to be inserted has a set of features that is a strict subset (ABDG) of features of one existing ISDA (class). In this case the candidate class (ABD)GH must be generalized to a class (ABD)G so that the methodology (M3) can be assimilated as its member. The generalized class is made a subclass of the new class (ABD)G. The fourth case is similar to the second one, except that candidate class (ABD)GH includes an "extra" feature D that is inherited from the class with features (AB)D. In this case one must modify the inheritance hierarchy following the path from the candidate class to the class from which one of the extra features is inherited. As a result the extra feature is removed from the original class to its subclasses. The procedure is recursively repeated until the candidate class (ABD)GH does not have the "extra" feature. The fifth case extends the previous one by including a feature (D') that is a modification of the features of the candidate class (AB)D. This is resolved by generalizing the candidate class to class (AB)B, with classes (AB)D and (AB)D' as its subclasses. The new methodology M5 can be added to class (AB)D'. The sixth case is similar to the previous one, except that one of the modified features (A') is an inherited feature of the candidate class (AB)D. Because A is the root of the initial class hierarchy (tree), this case illustrates that the procedure may lead to fragmentation of the initial class hierarchy.

The seventh case illustrates a multiple hierarchy where the combined features of two candidate classes (ABD)GH and (AIJ)DK exactly correspond to the features of an ISDM to be inserted (ABDGHJK). Multiple hierarchy always leads to a new class comprising all the features of the methodology to be assimilated. In this simplest case the new class turns out to be an intersection class (ABDGHJK) that inherits all its features from the two candidate classes. The eighth case is similar to the previous one, except the new methodology includes one new feature (P). This will be a noninherited feature of a new class, formed as in the previous case. In the ninth case one of the candidate classes (ABD)GH includes an extra feature H that is not among the features abstracted from the methodology M9. This again leads to generalization of the class (ABD)GH to a class (ABD)G, with the class (ABD)H as its subclass. The final case illustrates a situation of multiple inheritance where there are modified features (D' and G'). There are two obvious candidate classes, (ABD)GH and (AIJ)DK, which share two common features. One of these common features is due to a common superclass A. The second common feature D is one of the features to be modified. The case depicted in Figure 2 assumes that the candidate class (ABD)GH is processed first. It leads to its generalization to a class (AB)H with subclasses (ABD)GH and (AB)D'G'(H). The latter class includes a subset of the features of the methodology to be inserted. So, the class including the features of the methodology can be inserted as its subclass. After this process, the second candidate class is processed. The second candidate class is generalized to a class (AIJK)K with a subclass (AIJK)D,
and the generalized class (AIJ)K is made a superclass of the class (methodology) to be inserted.

Even though the A–A procedure looks like a computer algorithm, its execution requires human judgment, especially an analysis of the features (i.e., goals, guiding principles, fundamental goals, and principles of the ISD process) underlying the methodology to be inserted (step 1), and selection of the candidate classes (step 2). The procedure does not state criteria in the selection of the candidate classes, but obviously genealogical dependencies of methodologies should be considered in addition to sharing of common features. Several other steps (e.g., steps 6, 9, 16, 19, and 22–24) also include significant elements of human judgment. Consequently, the procedure is not deterministic.

The A–A procedure allows “multiple inheritance” in the sense that a class (ISDA) may be an immediate subclass of several ISDAs. “Multiple inheritance” raises the question of consistency between the approaches to be inherited. In other words, should it be required that an ISDM cannot inherit inconsistent goals, guiding principles, fundamental concepts, and principles of the ISD process? Even though we do not necessarily exclude internal contradictions in methodologies, reflecting some sort of “dialectical thinking,” it is obviously useful to be aware of such possible contradictions. When condensing multiple inheritance to essential features, the above procedure leads to identifying possible inconsistencies. One could see where such dialectical thinking becomes one of the principles of the approach in question.

The application of the above procedure may include the renaming of existing classes to better capture the new classification structure. This aspect is illustrated below when the A–A procedure is applied to Object Modeling Technique (OMT) [94] and Multiview [5]. OMT is used to illustrate the revision of a class hierarchy in the case of the object-oriented approach. Multiview, on the other hand, illustrates the case of multiple inheritance.

**OMT as an OO Approach**

To classify OMT requires a careful analysis of its goals, guiding principles, fundamental concepts, and principles of the ISD process (step 1). Even though OMT [94] has influences from the “information modeling approach,” the “SA/SD approach” and the “OO approach,” we will treat the last one as a relevant candidate class in its case (step 2). Step 3 selects the “OO approach” for a detailed analysis.

Let us assume that the “OO approach” considers the features (i.e., goals, guiding principles, fundamental concepts, and principles of the ISD process) listed in Table 2 as fundamental. From the viewpoint of our illustration, an essential feature of OMT is that it does not adhere to encapsulation in the OO analysis (see [49, 93]). Let us call its encapsulation principle “weak encapsulation.” Therefore OMT cannot be considered an instance of the “OO approach” as interpreted in Table 2, assuming that encapsulation there refers to “strict encapsulation,” that is, encapsulation throughout the OO development process (step 4).

The next question is whether OMT could be considered an instance of the class resulting from the deletion of some features of the class “OO approach” (steps 5 and
6). The possible feature to be deleted would be "strict encapsulation." Because OMT adheres to "weak encapsulation," the total deletion of the encapsulation principle does not work in this case. So we can skip steps 7 and 8.

Let us analyze whether OMT can be inserted as an instance of the class "OO approach" when its features are modified properly (steps 9–11). In this case the "strict encapsulation" principle of the "OO approach" must be modified to "weak encapsulation," which does not require encapsulation in the OO analysis (step 9). In order to make the names of classes more descriptive, let us change the name of the existing class, "OO approach," to "strictly encapsulated OO approach," and form a new class, "OO approach," which adopts the goals, guiding principles, fundamental concepts, and principles of the ISD process of the class "strictly encapsulated OO approach," except "strict encapsulation." Let us also form a class "weakly encapsulated OO approach" that includes "weak encapsulation." Assuming that the above generalization seems meaningful, the procedure (step 10) constructs a class structure in which class "OO approach" has two subclasses—"strictly encapsulated OO approach" and class "weakly encapsulated OO approach"—and inserts OMT as an instance of the latter class. The procedure (step 11) cleans the classification structure in the case that the feature "strict encapsulation" is inherited by the class "OO approach." Finally, there is the question of whether a class with features of the "OO approach," which do not include any encapsulation principle, is a meaningful ISDA (step 22). If it is considered meaningful, it will be added among ISD approaches (this might necessitate again the renaming of the class). If not, it is just a conceptual node in the classification structure that is not expected to have methodology instances independent of the methodology instances of its subclasses.

Multiview and Multiple Inheritance

Multiview [5] is a methodology that explicitly attempts to reconcile ideas from several ISDAs, most notably SSM, sociotechnical design, structured analysis, and information modeling. Therefore it provides an excellent case to illustrate multiple inheritance.

Following step 1 of the A–A procedure, Table 4 summarizes the characterizing features underlying Multiview. The characterizations in Table 4 follow closely the vocabulary of Multiview in order to preserve the authenticity of its interpretation. One should also note that Table 4 is the authors' interpretation, because the exposition of Multiview in [5] is only roughly outlined in the original work.

One can easily identify SSM and the sociotechnical (STD) approaches as clear candidate classes in the sense of step 2 of the A–A procedure. A rudimentary comparison of the features of the Information Modeling (IM) approach (Table 2) with Table 4 suggests that the IM approach does not form a reasonable candidate class for Multiview. Multiview has only adopted a few specific techniques (such as the ER model) from IM without other features. The relationship with the Structured approach (SA/SD) is more complicated. Multiview has clearly adopted dataflow diagrams. Even though not stated explicitly, one could insist that it implicitly adheres to the goals and
<table>
<thead>
<tr>
<th>Goals</th>
<th>Guiding Principles and Beliefs</th>
<th>Fundamental Concepts</th>
<th>Principles of the ISD Process</th>
</tr>
</thead>
<tbody>
<tr>
<td>Multiview provides a methodology for exploration into IS development (i:SSM). More specifically, it helps in providing answers to the following questions: 1. How is the IS supposed to further the aims of the organization? (i:SSM) 2. How can it be fitted into the working lives of the people in the organization who are going to use it? (i:STD) 3. How can the individuals concerned best relate to the computer in terms of operating it and using the output from it? 4. What information processing function is the system to perform? 5. What is the technical specification of a system that will come close enough to doing the things that you have written down in the answers to the other four questions?</td>
<td>To develop an information system, which is complete in both technical and human terms, requires multiple viewpoints comprising the viewpoints of human activity, information analysis, sociotechnical aspects, human-computer interface and technical design. The multiple viewpoints should be combined in a reasonably coherent “methodology framework”  <strong>Analysis of human activity system:</strong> To search for a particular worldview, Weltanschauung, to form the basis for describing system requirements (i:SSM)  <strong>Information analysis:</strong> To analyze the entities and functions of the system, independent of how the system will eventually develop  <strong>Analysis and design of the sociotechnical aspects:</strong> To produce a ‘good fit’ design, taking into account people and their needs and the working environment on the one hand, and the organizational structure, computer systems and necessary work tasks on the other (i:STD)  <strong>Design of the human-computer interface:</strong> The dialogue should be related to who will be using the system.  <strong>Design of technical aspects:</strong> Efficient technical design to ensure a quality technical system.</td>
<td>Analysis of Human Activity System  • human activity system, weltanschauung, root definition, relevant system, conceptual model (i:SSM), rich picture  Information analysis: Functional analysis  • function, event, dataflow  Information analysis: Data analysis  • entity, relationship, attribute  Sociotechnical design  • technical objectives, social objectives, technical alternatives, social alternatives (i:STD)  Human-computer dialogue Technical aspects  • processing application, information retrieval, database maintenance, control, recovery, monitoring</td>
<td>Flexibility of the process within Multiview  Five stages of IS analysis and design  Analysis of human activity system: A modification of the seven-stage model of [16]  <strong>Information analysis:</strong> Top-down decomposition of functions based on the primary task; conceptual model; construction of data flow models; verification of functional and entity models  <strong>Analysis and design of the sociotechnical aspects:</strong> Sociotechnical design, user participation (i:STD)</td>
</tr>
</tbody>
</table>
most of the guiding principles of the SA/SD approach. Despite this possibility, we will focus on only two candidate classes: SSM and STD.

Referring to steps 3 and 4 of the procedure, it is clear that Multiview cannot be inserted as an instance of any class in Figure 1. It is also obvious that none of the existing classes in Figure 1 can be generalized, by removing features of the original class (see Tables 2 and 3), to a class in such a way that Multiview could be regarded as an instance of the generalized class (step 5). Neither can the features of any class in Figure 1 be modified so that Multiview could be inserted as an instance of the modified class (step 9). So, let us proceed to step 12 of the above procedure.

Multiple inheritance (step 12) implies that Multiview forms an ISD approach, the features of which have been summarized in Table 3. Let us analyze SSM first as a candidate class (step 13). It is clear that the features of SSM as listed in Table 3 do not form a subset of features of Multiview in Table 4. (In particular, the stream of cultural analysis is missing in Multiview.) Therefore, let us proceed to step 16. The generalization of SSM by removing the stream of cultural analysis leads to a class the features of which form a subset of the features of Multiview. Let us name the new class “SSM-core” and the original class “SSM-1990.” “SSM-1990” forms a subclass of “SSM-core,” inheriting all the features of “SSM-core.” The only noninherited feature of “SSM-1990” is the stream of cultural analysis. Let us insert Multiview as an ISD approach, a subclass of “SSM-core.”

Let us return to STD as a candidate class. According to our interpretation, Multiview does not specifically emphasize guiding principles such as minimal critical specification and open-ended design, concepts such as variance and unit operation, and the evolutionary nature of the ISD process. This leads again in step 16 to a generalization in which a class “STD-reduced” is formed. It includes all the features of the original STD approach except the features mentioned above. The original STD approach, renamed “STD-complete,” includes the omitted features and as a subset of the class “STD-reduced” inherits all its features. The STD ideas of Multiview can be interpreted to cover the features of the “STD-reduced” class. Therefore Multiview can be inserted as a subset of “STD-reduced.”

When finalizing, there is the question of whether the new classes, “Multiview,” “SSM-core,” and “STD-reduced” can be considered as reasonable ISD approaches (step 22). Because “SSM-core” corresponds to the version of SSM of Checkland [16], it can be regarded as an ISD approach. To our knowledge, “STD-reduced” does not have any instance methodologies. Therefore, let us take a position that it does not form an ISD approach in and of itself.

To summarize, the above procedure leads us to consider Multiview as an ISD approach that inherits from “SSM-core” and “STD-reduced.” The inherited features of Multiview are indicated by i:SSM and i:STD in Table 4, respectively. Additionally, Multiview includes a number of other features, for example, the modification of the seven-stage model of Checkland [16] and rich pictures from the SSM origin. Neither of these was interpreted as an essential feature of SSM in the earlier section entitled “Illustration of Eleven ISD Approaches.”
The above example indicates that insertion of a new methodology does not necessarily lead to the most "elegant" class structure. The resultant class structure would have been more elegant had the initial structure more faithfully recorded the genealogy of SSM as an approach. In fact, one could have identified the aforementioned section two versions (subapproaches) of SSM: "SSM-1981" and "SSM-1990," corresponding to Checkland's two books [16, 17]. "SSM-1981" would have included the seven-stage process model as one of its principles of the ISD process. Both of these would be subclasses of "SSM-core" as defined above. In this situation Multiview could have been inserted directly as a subclass of "SSM-1981."

Discussion and Conclusions

As was suggested at the outset, this paper adds another building block to a larger research program concerned with clarifying and organizing the field of ISD. The most obvious contribution of the framework is that it provides a compass and intelligible map to find one's way around the increasingly dense and confusing "methodology jungle." This was illustrated by analyzing eleven ISDAs covering a broad spectrum of research in ISD. More fundamentally, the paper shifts the discussion of alternative ISDMs and tools to a higher level of essential features of ISDAs and underlying paradigms. These essential features provide a "language" with which it is possible to highlight the specific contribution of any new methodology or approach. One might be tempted to say: No longer should there be an excuse for duplicating the same features over and over again in slightly different versions of ISDAs that all belong to the same "class." Rather, new principles could be stated briefly and then the remainder of a methodology could simply be documented by pointing to the building blocks of those already known. In essence, we have provided a body of exemplars on how to identify and summarize the key features and concepts of ISDAs.

In addition to helping "make sense of the methodology jungle," we feel this paper offers a number of important additional insights. In particular, there are several implications of this work, partly extending the results of previously published papers and partly pointing the direction for possible future work. We articulate the implications of the paper in terms of four areas: (1) comparative methodology review, (2) methodology engineering, (3) identifying ISD as knowledge work, and (4) practical and educational implications of our analysis.

Comparative Methodology Review

As noted in the earlier section on "Analysis and Comparison of ISD Methodologies," the CRIS (Comparative Review of Information System development methodologies) conferences [80, 81, 82, 83, 105] form one of the most ambitious attempts to compare ISDMs. The CRIS project attempted to take stock of the then current methodologies, to apply them to a common case of an IFIP working conference and to conduct a feature analysis with the purpose of identifying commonalties and differences among
methodologies. However, the CRIS project failed for a number of reasons. One reason was that it lacked a systematic conceptual framework to make sense of the continuing rapid proliferation of new methodologies. A second reason was its excessive methodology focus. Methodologies as detailed and complex conceptual artifacts did not turn out to be a good unit of analysis for comparing alternative ways of conducting ISD. The CRIS experience triggered the FRISCO (Framework of Information Systems Concepts) project, which set as its objective the definition of a commonly accepted conceptual reference model for information system concepts, and resulted in a detailed reference model of about one hundred concepts [26]. At least in principle, the model facilitates conceptual mapping between different methodologies. In addition to the complexity of the framework and the difficulties of getting it widely accepted, it suffers from two major drawbacks, however. It focuses on quite a narrow, though complex, area of information system concepts, and it is unclear how it can evolve to adopt possible new concepts of future methodologies. In our view there is a need for a more dynamic framework that allows for an accommodation of the evolution of the field.

To such work, which reflects an important stage in the evolution of the ISD discussion, we have added a different and new form of analysis that focuses on a methodology’s “deep structure.” Just as in language theory both deep structure and surface structure are needed to convey the full meaning of a sentence, so too, for the description of a methodology, both its deep structure and its surface structure are needed to convey its full meaning. The “parsing,” representation, and discussion of ISDMs should allow the separation of the two. Of course, as any language evolves, so too must the representation scheme. It will need to be extended in future work. In fact, we would expect that refinements, modifications, and extensions to the classification proposed here would characterize “truly” new ISDMs. Our A–A procedure provides a vehicle for recognizing and tracking such extensions.

From the perspective of the ISDMs' deep structure, two additional insights emerge. First, because ISDAs aim at capturing the common essences of their instance ISDMs, ISDAs form more coherent and appropriate units of comparative analysis than specific ISDMs. With the benefit of hindsight, we can see that the CRIS project was pitched at too detailed a level, that is, the level of methodology instances rather than abstraction classes (approaches), which by nature are more parsimonious. This might very well be a major reason why it could not achieve its purpose of providing an overview of the most important features of methodologies.

Second, our framework also allows meaningful comparisons between ISDAs that are in different stages of their evolution in terms of the number of their methodology instances. The potential contributions of approaches that do not have concrete methodology instances have always been controversial because the procedural steps of the IS development process remain unclear without specific methods and tools. An advantage of our framework is that it permits one to clearly articulate this difficulty and thereby focus IS researchers’ energies on the question of how concrete methodologies could be instantiated rather than on simply denying the relevance of a certain line of reasoning for ISD. For example, the CRIS project would have had great difficulty
in recognizing the potential contribution of most of the nonfunctionalist approaches because they have not yet evolved any specific methodology instances. (Note: This is not only a problem for CRIS but also for other attempts (e.g., [70]) at documenting and comparing methodologies.)

Methodology Engineering

There is increasing empirical evidence to suggest that the systems development process must be adapted to organization-specific, project-specific, and other situational contingencies [76]. This has been used as an argument for the need for methodology engineering, which is normally interpreted as a bottom-up combination of pretested methodology components, fragments, or techniques [11, 35, 63, 101]. The feasibility of such an idea is supported by the finding that large parts of new object-oriented ISDMs are taken from preexisting ones [44]. Even though methodology engineering acknowledges this situational variety, it seems to underscore the commonality of the situations at the level of methodology components and fragments, assuming that a new combination of methodology fragments is able to address the unique features of the new situation. It seems, however, to neglect the underlying common “philosophy” [5] of the resultant methodology. Our proposal is that the concept of an ISD approach allows one to address this concern. An ISD approach has a certain gist that can be expected to assure the “philosophical” consistency (in addition to the formal consistency assured by metamodeling) of the resultant methodology.

The four-tiered framework suggests an alternative and complementary view of methodology engineering as an instantiation of existing ISDAs. This instantiational view extends the focus from methodology fragments to more general features to be inherited from ISD approaches and paradigms. An ISD approach may be interpreted as a template class from which one can instantiate a methodology possibly combining appropriate techniques. In fact, the field study of Smolander et al. [98], reporting that methodologies were mostly used as loosely coupled techniques, can be interpreted as empirical evidence of this. Even though the techniques may be loosely coupled in the formal sense, they may be consistently coupled in the sense that they are used within a single approach (e.g., the OO approach).

A further implication of the above is that the framework could inspire novel combinations of features. However, we do not wish to imply that all combinations of features are equally promising. On the contrary, it is our contention that there are a priori grounds for believing that some features are more likely to be successfully combinable than others. Those that come from the same paradigmatic base should be consistent with one another and therefore pose the least risk. Our rationale for this belief is quite simple: Since paradigms are considered to be consistent—because they have been rationalized and refined by the sheer amount of philosophical debate about their nature—features from the same paradigm would themselves be consistent. However, choosing all features from within the same paradigm is likely to be unrealistic. If one wishes to combine features from different paradigms in a methodology engineering effort it is reasonable to assume that the existing approaches can provide useful point-
ers regarding which features are compatible, since they incorporate a valuable experiential base of evidence by virtue of their prior applications in practice. Their current makeup has been influenced by their past history of successful and unsuccessful uses.

In summary, there are three options in the instantiational methodology engineering effort, each with an increasing level of risk: (1) select features from within a single paradigm; (2) allow multiple inheritance, but only within the confines of existing approaches. In other words, if none of the existing approaches combines feature A from paradigm X, and feature B from paradigm Y, then do not select such a combination either; (3) ignore existing precedence and go your own way at your own risk. Even if this latter option is chosen, our framework should still be valuable because it draws attention to the fact that the methodology architect is breaking new ground by selecting untried feature combinations. But at least the architect is forewarned that he or she is pursuing an ISD strategy with the highest level of risk, although potentially innovative.

It must be noted that it is beyond the scope of this paper to make any normative statements about what methodology fragments should be combined with which others. What has been accomplished so far is twofold: (1) pointing out the need to be sensitive to the underlying assumptions when combining tools, techniques, methods, and principles from a variety of ISDMs in methodology engineering; and (2) providing guidelines for thinking about how to achieve coherence. Taken together, this view of methodology engineering establishes the following research vision for finding out which specific feature combinations might be effective. This vision incorporates an action research program that provides the required experiential feedback. The above discussion is a summary of the research design for such an action research program. In distinction to prior action research projects on ISD, our research design is not aimed at testing a particular methodology, such as Multiview. Rather, it is pitched at a higher level of abstraction. This higher level of abstraction has the intent of finding out which feature combinations are feasible and effective so that they can be reused in an indefinite number of methodology instantiations. This is similar to object-oriented programming language design, which focuses on determining which initial set of objects would provide a good set of primitives.

Systems Development as Knowledge Work

Recently there have been a number of ambitious calls to establish systems development and software engineering as a profession [29, 32]. If one adopts strict criteria for what constitutes a profession, it is unlikely that ISD could be considered a profession in the foreseeable future [85]. Viewing ISD as knowledge work provides an avenue for proceeding toward professionalism without becoming involved in the highly political controversy on what is a proper profession and what is not. But what is knowledge work? It can be characterized as a type of work with the following four characteristics [52]:

1. There must be a clearly identified body of knowledge.
2. Work must be concerned with creating or manipulating representations rather than the physical objects of work.
3. It must require a deep understanding of the objects of work.
4. It must result in products that entail knowledge as their essential ingredient.

What is the body of knowledge in ISD? Applying [30], one can distinguish three components that make up the body of knowledge for ISD: knowledge of information technology; application domain knowledge; and systems development process knowledge. The hierarchical framework of ISD paradigms, approaches, methodologies, and techniques can be interpreted as a skeleton of the ISD process knowledge. The framework is not assumed to form an exhaustive list of the constituents of all ISD process knowledge, because process knowledge may also include other constituents that currently are not part of our framework. Such additional constituents are facts (e.g., about the applicability of techniques to specific problems), case histories (e.g., successful cases of an application of specific methodologies and approaches), metaphors (e.g., architecture as a metaphor of ISD), and theories (serving as warrants to back up the process knowledge). Our hypothesis is, however, that techniques, methodologies, approaches, and paradigms that are covered by our framework do form a significant component of ISD process knowledge, and as the references to tool building above illustrate, the framework could also integrate other constituents of process knowledge.

While our framework only deals with a part of systems development process knowledge, this is nevertheless an essential step toward clarifying the body of ISD knowledge. Insofar as the framework captures the deep structure of approaches, it contributes to a deeper and richer understanding of ISD. Finally, through facilitating the documenting of the essential features with their associated tools, it contributes to building representations of the objects of work in ISD. Insofar as systems developers have relied on symbolic representations in analysis and design, they naturally work on representations deflecting physical implementation until the end of IS development. Our framework further reinforces the symbolic nature of IS work by providing higher levels of abstractions in representing and thinking about methodologies and approaches.

Practical and Educational Implications

In this paper it was suggested that the formal structures for guiding information systems development work are better conceived as general ISDAs that organize generic principles into convenient schemata. If this is true, it suggests that practicing systems analysts should pay attention to the generic features, which make up the essences of whole classes of ISDMs rather than individual instantiations. This principle can be illustrated by metaphorically applying Brooks’ [12] distinction between essences and accidents of software products to ISDAs. One can note that a complete ISDM necessarily includes several accidental features (e.g., related documentation notations). These, while important for practice, do not define its essence. The general ISDAs defining the essences of their member ISDMs may be conceived as templates that help generate more detailed ISDMs “on the fly,” possibly combining existing “ISDM fragments” [35]. We wish to point out, however, that when combining “ISDM frag-
ments” in methodology engineering, it useful to understand their underlying perspectives and assumptions.

A further implication of the focus on methodological “essences” follows for the education of future practitioners. These essences can be interpreted to correspond to “embedded perspectives” as used by [78]:

A method[ology] seldom contains explicit statements about its perspective. The perspective is instead embedded implicitly in its guidelines, techniques and tools. . . For this reason a major objective in any education in systems development should be to teach students to analyze every method[ology] they are exposed to with the purpose of identifying its embedded perspective.” ([78], as quoted in [114]).

What we argue here is that the application of our framework can help one to recognize the “embedded perspectives” of ISDMs or ISDAs and that the recognition of the embedded perspective is not just important for students, but also for practicing analysts. Apart from the associated ethical issues, analysts need to understand to what extent they are reinforcing or attempting to reform existing organizational practices and policies with their preferred methods and tools. Without understanding the perspectives underlying professional practices, an analyst succumbs to a “professional blind spot.” Our framework provides a vehicle for documenting the perspective underlying any professional method. Through this, in principle at least, it can be expected to contribute to continuing improvements of practice by providing a broader theoretical understanding of systems development approaches.

**NOTES**

1. The terms “method” and “methodology” need some clarification, as their use differs in Europe and North America. In Europe “method” is increasingly used to refer to the systematic procedure of conducting systems development, whereas “methodology” refers to the study of methods (cf. research methods and methodology). In North America, “methodology” is used as Europeans use the term “method.” To avoid the confusion this paper uses the acronym “ISDM.” Where we do use one term, we shall adopt the North American “methodology.”

2. To illustrate the difference, if the Unified Modeling Language had been developed as an object-oriented approach rather than as a specific object-oriented methodology, the major focus would have been on the intersection of the three major methodologies underlying it [10, 54, 94] rather than on their union.

3. The distinction between goal, guiding principles, fundamental concepts, and principles of the ISD process has similarities with the framework proposed by Song and Osterweil [99], in particular with the notion of “concept” in their “type hierarchy” (see the section on “Analysis and Comparison of ISD Methodologies” for a brief review of their framework). They define a “concept” as an idea that influences the design of a software development methodology and distinguish concepts such as “problem,” “principle,” “guideline,” “criterion,” and “measure.” To illustrate the above concepts, they provide examples such as to produce a changeable program (problem), achieve high cohesiveness (principle), find a noun to identify objects (guidelines), coupling, and cohesiveness (measure). These examples suggest that the concept “problem” in [97] is close to “goal” in our framework. Their “principles” are similar to our “guiding principles and beliefs.” Their “guidelines” are structured according to the actions in the software process and in that sense resemble our “principles of the ISD process.” They seem to be at a much more detailed level, however, as illustrated by the example above. Our “fundamental
concepts" essentially corresponds to their concept of "artifact," but also includes concepts related to their "measures." "Fundamental concepts" also cover major "design issues" in the framework of Song and Osterweil (e.g., essential model versus implementation model in the case of SA/SD).

4. We are aware that not all OO methodologies completely share all these features (e.g., encapsulation). Some of these differences will be analyzed in the section entitled "OMT as an OO Approach." A thorough analysis of the OO approaches is beyond the scope of this paper, however.

5. The term "paradigm" has been a controversial concept (cf. [65]) ever since Thomas Kuhn [61] introduced it in his influential book on scientific revolutions. Kuhn used it to describe the historical development of the natural sciences, in particular physics and astronomy. In the social sciences the term "paradigm" is usually used to describe the basic assumptions underlying coexistent theories rather than the evolution of the social sciences [14].

6. The concept of inheritance is a much debated concept in object-orientation (cf. [37]). Note that inheritance in our context does not concern only attributes (ontology, epistemology, methodology, and ethics in the case of paradigms; goals, guiding principles, fundamental concepts, and principles of the ISD process in the case of approaches), but also their values (i.e., the specific positions a paradigm or approach takes with regard to the attributes).

7. The quotation marks in "ISD approach" are there because in most of the publications describing the approaches, the concept of "approach" is visible only in an embryonic form.

8. These two situations are merged, however, in the actual procedure for maintaining the classification structure (see Appendix 2).

9. The procedure works for classifying an approach as well by simply treating the approach as a methodology. Consider an ISD approach "A" without methodology instances. Apply the A–A procedure by treating "A" as a methodology M. After inserting M into the classification framework, the class with M as an instance can be named "A," and then M can be removed from the classification structure.

10. The interpretations of the eleven ISDAs are not unambiguous. A more detailed analysis, distinguishing subapproaches within the main approaches listed in Tables 1 and 2 might have allowed for a more unambiguous interpretation. However, such an analysis is beyond the scope of this paper. (But see, for example, [45] and [90] for reviews of different semantic information/data models underlying the IM approach, and [28] and [79] for reviews of the STD approach). We do note however, that our A–A procedure for inserting new ISDAs and ISDMs in the classification structure (see the section on "The Procedure for Maintaining the Classification Structure") does allow us to distinguish different subapproaches.

11. The position of Multiview in the framework of Figure 1 is analyzed in more detail in the section on "Multiview and Multiple Inheritance."

12. That is, classes the features of which are close to those underlying M. These features include both inherited and immediate features of the class in question.

13. C* with M as its instance shares the goals, guiding principles, fundamental concepts, and principles of the ISD process underlying M.

REFERENCES


84. OPEN. www.csse.swn.edu.au/cotar/OPEN.


Appendix 1: Description of the Eleven Information Systems Development Approaches

IN ADDITION TO THE MAIN REFERENCES, the justifications we offer aim at explaining some of the non-obvious interpretations that were made—interpretations that may differ from the prevailing beliefs about the eleven ISDAs in question.10 (For more information, see [40, 47, 51].)

Six Functionalist ISDAs

The goals and guiding principles of the Structured (SA/SD) approach are taken directly from [115]. The fundamental concepts reflect the data flow tradition of Structured Analysis (e.g., [22]) in contrast to SADT [91], which does not have data stores and terminators as explicit concepts. In contrast to the many accounts of Structured Analysis (e.g., [6]), Table 1 does not consider the top-down decomposition process, advocated by DeMarco [22] for example, as essential to SA/SD as an approach, because later versions (e.g., [115]), have rejected it. But Table 1 does identify top-down partitionable models as an essential aspect of the SA/SD approach (cf. guiding principles). The summary of the Information Modeling (IM) approach in Table 1 is a synthesis of a large number of publications (e.g., [71]), reflecting the fact that the IM approach has numerous methodology instances. These, for example, dominated the
ISDMs compared in the CRIS effort [80, 81, 82]. Table 1 identifies an attribute as one of the fundamental concepts of the IM approach, following the ER tradition [18] in this respect, even though the so-called binary models (e.g., NIAM [46]) do not use it, adopting instead attributes as entities. The analysis of the Decision Support Systems (DSS) approach has been condensed from [56] and [100]. The summary of the Sociotechnical (STD) approach is mainly based on [74] and [88, 89]. In view of Pava’s criticism of the concept of “autonomous work group,” the concept has been dropped from the level of the approach, even though it has had a significant role in the sociotechnical design tradition (e.g., [38]). The analysis of the Infological (INF) approach is based on [68]. The long list of goals of the Object-Oriented (OO) approach has been borrowed from [31] as quoted in [37]. The guiding principles, fundamental concepts, and principles of the ISD process are relatively widely shared in the OO community, even though [94] and [97] do not share the idea of encapsulation at the level of OO analysis (see the section entitled “OMT as an OO Approach”).

Table 1 shows clear differences between the six functionalist ISDAs in terms of their goals, guiding principles, fundamental concepts, and principles of the ISD process. In the case of goals, the STD approach clearly differs from the others in its emphasis on job satisfaction objectives. The six ISDAs also have more or less distinctive guiding principles, such as the top-down partitionable transformation (process) models in the SA/SD approach, data as a stable basis for applications in the IM approach, IS evolution in the DSS approach, minimal critical specification in the STD approach, division of system development into a number of development areas in the INF approach, and seamless analysis, design, and implementation in the OO approach.

Five Nonfunctionalist ISDAs

The characterization of the Interactionist approach [57, 58, 59, 60]) is based on the web and institutional views rather than the discrete entity and tool views, following Kling and Scacchi’s [60] preference in this respect. Kling and Scacchi provide a rich set of ideas to characterize the web and institutional views. Table 2 is able to summarize only some of the authors’ major points. It must be noted that the Interactionist approach is very embryonic as an ISD approach, because it has not been elaborated into a concrete methodology for ISD. Nevertheless, we propose to treat it as an approach, because we argue that an ISDA may also be based on purely descriptive research (nomothetic or idiographic), provided that the concepts, understanding, and insights gleaned from such descriptive research contribute a useful framework for ISD. Because descriptive theories can have normative implications, it is possible to construct both ISDMs and ISDAs from them. In particular, at the level of approaches, we encourage their derivation from descriptive theories. This is because the general features of ISDAs link more directly to the theoretical constructs of descriptive theories, whereas the ISDMs contain more practical details, such as notations, tools, etc. There is concrete literature evidence for the usefulness of the Interactionist approach for ISD, because Kling and Scacchi [60, p. 71] suggest that the web model descrip-
tions could yield useful normative guidelines for ISD. This does not necessarily require any extensive development of new artifacts, but the question is largely about perception and interpretation (descriptive versus normative).

The Speech Act (SA)-based approach (e.g., [2, 3, 110, 111]) is an example of a more recent ISDA with a number of methodology instances. Our interpretation of the SA-based approach is not entirely based on the abstraction of common features of existing ISDMs, but also on the theoretical background of the SA-based approach. To illustrate this, Table 2 identifies discourse/conversation analysis and propositional content analysis (information modeling in [66] as a fundamental aspect of the SA-based approach. This is based on our interpretation of Speech Act theory (see also [67]). Winograd et al. [27, 110, 111] do not address the subject of propositional content analysis. Consequently, we interpret the SA-based methodology of Winograd et al. as partial in this respect. It is beyond the scope of this paper to go into details of these specific ISDMs. We only wish to point out with this example that each methodology instance of the SA-based approach may use slightly different activities, techniques, and tools for discourse and propositional content analysis.

Soft Systems Methodology (SSM) approach is an example of an ISDA with a few methodology instances, for example, SSM according to Checkland [16] and Checkland and Scholes [17], Wilson’s [109] version, Multiview, [5] and FAOR [96]. At the same time SSM exemplifies the case of developing both as an ISDA and as an ISDM in parallel. We interpret the extension of SSM to include the stream of cultural analysis [17] in addition to the stream of logic-based analysis as an evolution of SSM as an approach. At the same time Checkland and Scholes cautiously distance themselves from the seven-stage model of the stream of logic-based analysis [16]. This can be interpreted as evolution of SSM as a methodology from Checkland’s original version. Even though techniques like rich pictures and CATWOE are intimately associated with SSM, they are techniques that do not define SSM as an approach.

The Trade Unionist approach [8, 9, 23, 33, 62] has evolved through a number of generations. Even though these have implied certain changes in emphasis and focus, we interpret them as a continued expansion and enrichment of the approach. The development of the trade unionist approach has mainly taken place at the level of an ISDA. As a methodology the trade unionist approach is still somewhat embryonic, although some specific techniques (such the union-led shadow project organization, mock-ups, and prototypes) have been proposed.

The Professional Work Practice (PWP) approach [1, 73] has also mainly been developed as an ISDA. It includes a limited repertoire of specific techniques (such as diagnostic mapping techniques, metaphorical design, future workshops, and use of diaries). As an ISD approach, it has been particularly explicit in stating a number of principles for the ISD process. It lists twelve performance principles, ten management principles, and two principles of the relationship between management and performance. These principles revolve around the dualities listed under the “Concepts” in Table 2. The basic message of these concept pairs is that the opposites of each duality are interacting and therefore should be addressed concurrently.
Appendix 2: The Assimilation–Accommodation Procedure

The procedure starts with abstracting the characterizing features of the ISDM to be inserted. After that it checks whether there are any candidate classes for the methodology to be inserted. If not, the methodology represents a new ISD approach and is inserted accordingly (step 2). If there are candidate classes, the procedure includes three “passes.” Pass 1 (steps 3 and 4) checks whether a new methodology is an instance of any of the candidate classes. If it is, the methodology is simply inserted as an instance of the class in question. Because the features of existing classes differ from each other, there is at most one such class (ISDA). If a methodology is not directly an instance of any of the candidate classes, there is a possibility that it is a version of some existing class or that it is an integrated methodology that inherits features from several ISDAs.

Pass 2 (steps 5–11) checks whether a methodology can be considered an instance of some class when an existing candidate class is generalized by deleting or modifying some of its features. If there is such a candidate (steps 6 and 9), the classification structure is modified and the methodology is inserted as an instance of the generalized class (steps 7 and 10). Steps 8 and 11 modify the class structure in the case where the deleted or modified features were inherited in the class structure.

Pass 3 (steps 12–21) deal with the situation of multiple inheritance. In this case the methodology must be abstracted into a new approach (step 12). Step 14 checks whether the methodology (or the new class formed from it) shares all the features of some existing class. (Note that during this third pass the features of the existing class do not cover all features of the methodology in question.) In such a case, the new class (corresponding to the methodology) is inserted as a subclass of the existing class in question. Otherwise, the possibility that a methodology (or the new class formed from it) is a version of some existing class, when generalized appropriately, is analyzed in steps 16 and 19. If possible, steps 17 and 19 form the generalized class with its subclasses, and steps 18 and 21 modify the class structure if the deleted or modified features were inherited in the class structure. The third pass ends when all the features of the methodology (or the new class formed from it) are inherited in the resultant class structure (steps 15, 18, and 21) or the set of candidate classes have been analyzed (step 13).

The A–A Procedure (M)

1. Insert M as an instance of the class “ISD methodologies.” Analyze the features \( F = \{F\} \), that is, goals, guiding principles, fundamental concepts, and principles of the ISD process underlying M.

2. Consider whether there are candidate classes \( C = \{C\} \) for M.\(^{12}\) If “no,” form a class \( C' \) with M as its instance.\(^{13}\) Insert \( C' \) as an instance of the class “ISD approaches” and go to step 23.
**M as an Instance of an Existing Class C**

3. Select a candidate class \(C \in C\). If all candidate classes are analyzed, go to step 5.

4. Analyze whether the features of \(C\) and those underlying \(M\) are equivalent, that is, \(F(C) = F(M)\). If “yes,” insert \(M\) as an instance of \(C\); insert \(C\) as an instance of ISD approaches (if not already done) and go to step 24. If “no,” go to step 3.

**M as an Instance of a Modified Class \(C'\)**

5. Select a candidate class \(C \in C\). If all candidate classes are analyzed, go to step 12.

6. Consider whether \(C\) can be generalized to a class \(C'\) so that \(C'\) includes a strict subset of immediate and inherited features of \(C\) (i.e., \(F' = F(C') \subset F(C)\)), and that the features \(C'\) are equivalent to features underlying \(M\), that is, \(F(C') = F(M)\). If “no,” go to step 9.

7. Form a class \(C'\) with a subclass \(C\). Associate the subset \(F'\) of (immediate and inherited) features of \(C\) \((F' \subset F(C))\) to be generalized with \(C'\). The features of \(C\) not to be generalized remain in \(C\) (i.e., \(F(C) = F(C) - F'\)). Insert \(C'\) as a subclass of all the classes of which \(C\) was a subclass. Delete from \(C'\) those features that it inherits from its superclasses. Insert \(M\) as an instance of \(C'\).

8. If \(C'\) inherits features not-to-be-generalized (i.e., \(F \in F(C)\)), search the class \(C^*\) from which the feature was inherited. Do \(CLEAN(C^*, F)\). Repeat step 8 until no \(F \in F(C)\) is inherited by \(C'\). If any of the not-to-be-generalized features are inserted in \(C'\), delete them. Insert \(C'\) as an instance of ISD approaches. Go to step 23.

9. Consider whether a subset \(F^* \subset F(C)\) of the immediate and inherited features of \(C\) can be modified to features \(F' = \{F'\}\) so that \(C\) and \(C''\), in which \(C\) includes to-be-modified features \(F^*\) and \(C''\) includes the modified features \(F'\), are subclasses of \(C'\), and the features of \(C''\) and \(M\) are equivalent (i.e., \(F(C'') = F(M)\)). If “no,” go to step 5.

10. Associate the not-to-be-modified features of \(C\) with \(C'\) (i.e., \(F(C') = F(C) - F^*\)). The to-be-modified features remain in \(C\) (i.e., \(F(C) = F^*\)). Associate the modified features with class \(C''\) (i.e., \(F(C'') = F'\)). Insert \(C\) and \(C''\) as subclasses of \(C'\). Insert \(C'\) as a subclass of all the classes of which \(C\) was a subclass. Delete from \(C'\) those features that it inherits from its superclasses. Insert \(M\) as an instance of \(C''\).

11. If any of the to-be-modified features (i.e., \(F \in F(C) - F^*\)) is inherited by \(C'\), search the class \(C^*\) from which the feature was inherited. Do \(CLEAN(C^*, F)\). Repeat step 11 until no modified features are inherited by \(C'\). If any of the to-be-modified features are inserted in \(C'\), delete them. Insert \(C'\) as an instance of ISD approaches. Go to step 22.

**M and Multiple Inheritance**

12. Form a class \(C''\) with \(M\) as its instance. Associate all the features underlying \(M\) with \(C''\), that is, \(F(C'') = F(M)\). Insert \(C''\) as an instance of the class “ISD approaches.”

13. Select a candidate class \(C \in C\). If all candidate classes are analyzed, go to step 22.
14. Analyze whether C shares a subset of features of C", that is, \(F(C) \subset F(C')\). If “yes,” insert C" as a subclass of C. If “no,” go to step 16.

15. If all the features of C" are inherited in the class structure, go to step 22. Otherwise go to step 13.

16. Consider whether C can be generalized to a class C' so that C' includes a strict subset of immediate and inherited features of C (i.e., \(F(C') \subset F(C)\)), and that C' shares a subset with features of C", that is, \(F(C') \subset F(C'')\). If “no,” go to step 19.

17. Form a class C' with a subclass C. Associate the subset \(F'\) of (immediate and inherited) features of C \((F' \subset F(C))\) to be generalized with C'. The features of C not-to-be-generalized remain in C (i.e., \(F(C) = F(C') - F'\)). Insert C' as a subclass of all the classes of which C was a subclass. Delete from C' those features that it inherits from its superclasses. Insert C" as a subclass of C'.

18. If C' inherits features not-to-be-generalized (i.e., \(F \in F(C)\)), search the class C* from which the feature was inherited. Do \(\text{CLEAN}(C*, F)\). Repeat step 18 until no \(F \in F(C)\) is inherited by C'. If any of the not-to-be-generalized features are inserted in C', delete them. If all the features of C" are inherited in the class structure, go to step 22. Otherwise go to step 13.

19. Consider whether a subset \(F* \subset F(C)\) of the immediate and inherited features of C can be modified to features \(F' = \{F'\}\) so that C and C"", in which C includes to-be-modified features \(F*\) and C"" includes the modified features \(F'\), are subclasses of C', and C"" shares a subset of features of C"" (i.e., \(F(C'') \subset F(C'')\)). If “no,” go to step 13.

20. Associate the not-to-be-modified features of C with C' (i.e., \(F(C') = F(C) - F*\)). The to-be-modified features remain in C (i.e., \(F(C) = F*\)). Associate the modified features with class C"" (i.e., \(F(C'') = F'\)). Insert C and C"" as subclasses of C'. Insert C' as a subclass of all the classes of which C was a subclass. Delete from C' those features that it inherits from its superclasses. Insert C"" as a subclass of C"".

21. If any of the to-be-modified features (i.e., \(F \in F(C) = F*\)) are inherited by C', search the class C* from which the feature was inherited. Do \(\text{CLEAN}(C*, F)\). Repeat step 21 until no modified features are inherited by C'. If any of the to-be-modified features are inserted in C', delete them. If all the features of C"" are inherited in the class structure, go to step 22. Otherwise go to step 13.

22. Consider which of the new classes C' and C"" are meaningful as potential ISD approaches. Insert them as instances of the class “ISD approaches” (if not already done).

23. Consider the paradigmatic influences of the new “ISD approaches.”

24. The classification has been performed successfully.

\textbf{Procedure CLEAN(C,F)}

1. Delete the feature F from C and insert it to all subclasses of C.

2. If C becomes empty (i.e., does not include any immediate features), insert all its subclasses as subclasses of all its superclasses and delete C.

3. If C had instance methodologies, insert them using the A–A procedure.