Available online at www.sciencedirect.com
ScienceDirect
Computer Standards & Interfaces 31 (2009) 98–109
COMPUTER STANDARDS & INTERFACES
www.elsevier.com/locate/csi

Mechanisms for communication between business and IT experts

Haim Kilov [a,*], Ira Sack [b]

[a] Independent Consultant, and Stevens Institute of Technology, United States
[b] Stevens Institute of Technology, United States

Received 24 February 2007; accepted 18 November 2007
Available online 28 November 2007

Abstract

The paper shows how a system of important concepts and approaches proposed by system thinkers (such as philosophers, mathematicians, engineers, and computing scientists) and described in international (ISO) standards has been used to understand and specify various kinds of business and IT systems, and to base IT work on a solid foundation that has been used for communicating with non-IT experts, thus establishing successful and meaningful interactions between business and IT experts and organizations. These common elegant concepts — such as abstraction, system, structure, relationship, composition, pattern, name in context, etc. — come from exact philosophy and mathematics. They have been stable for centuries, and have been successfully used in theory, in industrial practice (including international standards), and in teaching of business and IT modeling. The essential stable *semantics* of these fundamental concepts and of systems specified using them ought to be clearly separated from the accidental (often IT-industry-imposed excessively complex and rapidly changing) details. The paper includes two case studies of applying the approach – with demonstrable success – in a large financial institution and in a leading publishing company.

Keywords: System of concepts; Semantics; Business-IT communication; RM-ODP; General Relationship Model (GRM)

1. Introduction

The proverbial communication gap between business and IT experts has been a sad reality for quite a while. This has led to substantial problems in information system design and development including significant monetary losses together with loss of customers' trust and patience [15]. As noted in *Computerworld* (October 11, 1999), "85% of IT departments in the US fail to meet their organizations' strategic business needs". More recently, almost the same percentage (84%) was cited in *Computerworld* (May 9, 2005) as the percentage of top executive MBA candidates (at the Fisher College of Business at Ohio State University) with full-time jobs who "when asked to recall personal experiences related to IT, cited very negative situations". At the same time, not all IT projects fail: in some environments business and IT experts do communicate in a successful manner. Therefore, it would be instructive to determine – and make explicit – some properties of successful business-IT communication.

The essential role of communication in human society was emphasized, for example, by Ludwig von Mises in his classical book "Human Action" [42]: "It is always the individual who thinks. Society does not think any more than it eats or drinks. […] There is joint action, but no joint thinking. There is only tradition which preserves thoughts and communicates them to others as a stimulus to their thinking. […] The foremost vehicle of tradition is the word. Thinking is linked up with language and vice versa. Concepts are embodied in terms. Language is a tool of thinking as it is a tool of social action". Certainly, people who want to communicate successfully should use not only (and not even mainly!) the same terms – see the section about names below – but rather the same system of concepts. (As an example, the same important concepts in [21] are expressed both in English and in German.) On a smaller scale, the ability of an organization to enable and sustain an environment in which effective communication takes place substantially contributes to the success of that organization [40, p. 329]. Therefore it is clear that "understanding of each other's domains by the business and IT functions" [40, p. 329] is of utmost

* Corresponding author.
E-mail addresses: haimk@acm.org, hkilov@stevens.edu (H. Kilov), isack@stevens.edu (I. Sack).

0920-5489/$ - see front matter © 2007 Elsevier B.V. All rights reserved.
doi:10.1016/j.csi.2007.11.001

importance. As observed in [52], *shared domain knowledge* was found to influence both short- and long-term alignment between business and IT objectives. Alignment is conceptualized as a state or an outcome [6,10] having both intellectual and social dimensions. The intellectual dimension is defined as "the state in which a high-quality set of interrelated IT and business plans exists", while the social dimension is defined as "the state in which business and IT executives […] understand and are committed to the business and IT mission, objectives, and plans" [51]. Similarly, "communications maturity ensur[es] ongoing knowledge sharing across organizations" [40, p. 71] which is necessary for proper interaction with customers [40, p. 362], partners, competitors, consultants, regulators, and so on.

Thanks to the division of labor, experts can concentrate – and achieve a lot – in their specific areas of expertise. Furthermore, they can, and should, use the achievements of other experts (in their own and other areas) as and when needed. In order to do that, it is necessary to use a common system of basic concepts for communication. This system of concepts does not belong to any specific area of expertise, but rather is used as a fundamental one in all of these areas. When we want to understand "what is there" in a specific area of expertise (that is, when we discover an ontology of that area) and when we communicate our understanding to others (that is, when we represent the ontology in some manner), we have to use such fundamental common concepts. And the importance of understanding and properly using these concepts increases in those areas of expertise that are less specialized, such as the area of information technology, especially, understanding and proper handling of complexity.

2. A system of common concepts

Where do these common concepts come from? As Mario Bunge noted, "all factual sciences, whether natural, social, or mixed, share a number of philosophical concepts … and a number of philosophical principles" [8]. More specifically, technologists "who work on general theories of systems, control theory, optimization theory, the design of algorithms or simulation are applied philosophers of sorts, since they use philosophical concepts, such as those of event and system, and philosophical principles, such as those of the existence and lawfulness of the external world" [7]. This is where business and IT experts can and should find a common ground for understanding and therefore communication.

In order to succeed, the fundamental concepts – rather than various rapidly changing IT-industry-imposed fashionable "new things" [15] – ought to be used in an explicit manner. These concepts are stable, and have been so for centuries. At the same time, proper exactification of some of these concepts became possible only more recently, thanks to the developments in exact philosophy, semiotics, mathematics, and computing science. In particular, it has been necessary for success to abstract away from the (extreme) complexities often imposed by software and service vendors, and to go "back to basics" [30].

We will see that the same fundamental system of concepts has been used both in exact philosophy and in important IT-based work, such as international standards (for example, RM-ODP — the Reference Model of Open Distributed Processing [26]). These concepts have been successfully used not only in theory, but also in industrial practice of business system modeling, design and development (see, for example, [33,16,39]), as well as in teaching (using information modeling and RM-ODP, with an appropriate very small subset of UML used for representation purposes) in a unit on modeling of a suitable IS course (such as data and knowledge management) for students of management, MBAs, IT, etc. [37].

Many philosophical foundations of these fundamental concepts have been exactified by Mario Bunge. Other thinkers, such as Wittgenstein [57] and F. A. Hayek [20,21], also contributed a lot. Some fundamentals go back to Aristotle: for example, there exists a very important difference between the Aristotelian and a prototypical (example-based) approach to modeling [41,36]. The former provides for the intension of the model; the latter – despite being buzzword-compliant in some popular IT-based methodologies – provides for its probably incomplete extension. When we use the former, we can always find out whether a fact corresponds to the model, while when we use the latter, we often cannot do that. In the same manner as testing of a program can show an error but cannot demonstrate that the program is correct (E. W. Dijkstra), examples are helpful to illustrate an ontology, but are inadequate when we want to create and communicate it. After all, testing of a program, or of an ontology (that is, using examples to check it), differs significantly from creating and understanding it.

It is very instructive to notice that essential concepts and structuring rules defined in the RM-ODP standard are based on the same fundamentals from exact philosophy, as well as on mathematics — "the art and science of effective reasoning" (E. W. Dijkstra). This international standard was created to support the definitions of "the basic concepts to be used in the specifications of the various components which make up the open distributed system" [26]. While such a standard may seem to be very specific and useful only in computer-based IT environments, open distributed systems exist not only – and not even mainly – in computer-based environments: such systems have existed and have been described in all kinds of human endeavor, for example, in a market economy [2,42], or in reasoning about purposeful behavior [22]. As an example, it was very easy and enlightening to show how the concepts and structuring rules defined in RM-ODP perfectly apply to a precise specification of a specific open distributed system – a banking clearing house – based on a century-old text [14] and still – without changes but perhaps with some refinements – applicable now [29].

Similarly, we observe that crucial business concepts have been known and explicitly used for a long time; they were clearly expressed, for example, in works by Adam Smith. And we also observe, with great pleasure that Smith's – more specialized – *Wealth of Nations* was based on philosophical foundations laid in his *Theory of Moral Sentiments*. This paper shows how the system of important concepts and approaches proposed by these thinkers has been used to understand and specify essential fragments of the ontologies of various business

and IT systems, and thus to establish successful communication between business and IT experts.

3. What's in a name: A (relatively) familiar example

The abysmal notion of "meaningful names" has existed in IT for quite a while [28]. This notion is not IT-specific: as observed by F. A. Hayek in his thought-provoking and eloquent Chapter "Our poisoned language" ([23], Chapter Seven), "while we learn much of what we know through language, the meanings of individual words lead us astray: we continue to use terms bearing archaic connotations as we try to express our new and better understanding of the phenomena to which they refer". The inadequacy of "meaningful names" in IT was recognized by Grace Hopper as early as in 1957: "[w]hile the computation of the square root of a floating decimal number remained the same in Pittsburgh, Los Angeles, and New York, the computation of gross-to-net pay obviously did not remain the same even in two installations in the same city" [25]. As a modern-day example, consider relying on "data names" – instead of an explicit information model – in XML within the context of Semantic Web (for a criticism and a semantics-based approach, see [39]). The need for semantic information integration based on meaning rather than on "data" and tools (none of which integrates data beyond the syntactic level) has been emphasized for a long time, and is becoming acknowledged in popular industrial publications [50] where, for example, we encounter references to "a hundred different meanings" of such terms as "on-time percentage" or "customer profitability". Let us try to exactify the concept of a name.

We start with observing that synonyms and homonyms exist: the same thing (or action, or process, or relationship) may have different names, and the same name may denote different things (or actions, etc.). Although many IT-based approaches promote the apparent need to determine and impose the "only correct name" of a thing, other names of that same thing, including nicknames and abbreviations, have been (and will be) used in business anyway, leading to various and often serious problems. It is counterproductive, for example, to try to determine which of the 50 or more somewhat different types of things, all named "patient" by various groups of HMO stakeholders *in different contexts*, is a real patient, and which of them, therefore, are "not real patients". To distinguish between these different types of things with the same name, we explicitly use contexts, such as "this is what emergency calls a patient", or "this is what insurance company XXX calls a patient". (If from an insurance company's viewpoint a patient is the person who pays insurance premiums then the statistical datum stating that 40% of male patients have been pregnant at some point in time could be understood…)

We continue with observing that in business, the context of a name may not always be made explicit, and may change even within the same narrative – even the same sentence – presented by a single person. Questions about semantics, determined to a large extent by the context, can always be (informally) asked and answered, although in many business situations this is not done because it is wrongly assumed that "everyone knows what XXX is". Since interactions with computer-based systems require a substantially more disciplined approach to using names, a strict discipline may often be imposed – often implicitly – on using "the same meaningful name" to denote apparently "the same thing". Such approaches often lead to failures.

There is no need to invent new and better approaches in order to solve the "name problem". Philosophers have noted that names by themselves do not convey any meaning: as stated by Wittgenstein, "[o]nly the proposition has sense; only in the context of a proposition has a name meaning" [57]. Similarly, semiosis was defined by Charles Saunders Peirce as "an action, or influence, which is, or involves, a cooperation of three subjects, such as a sign, its object, and its interpretant, this tri-relative influence not being in any way resolvable between pairs… If this triple relation is not of a degenerate species, the sign is related to its object only in consequence of a mental association, and depends upon habit" [49]. The concept of an interpretant is very close to the concept of a context within which a name is being considered [31]. Clearly, the context itself – the set of relevant relationships defining the structure of the relevant domain (that, is, a fragment of the ontology) – has to be defined in an explicit manner. Correspondingly, David Hilbert noted that the content of elementary geometry does not suffer any changes if we replace the words "point", "line", and "plane" by the terms "chair", "table", and "bar" [58].

This philosophy-based approach provided a solid foundation of the *system* of corresponding RM-ODP definitions [26]: a name is "a term which, in a given naming context, refers to an entity", and an identifier is "an unambiguous name, in a given naming context". Clearly, the same entity may have several names and several identifiers in the same naming context. These definitions are based on semantics and apply to any system, independently of whether it does or does not use computer-based information technology. And there has been no need to use IT-specific (or any buzzword-compliant and thus "rapidly changing") concepts for these definitions. They have been distilled, taking into account the appropriate philosophical foundations, from a huge body of industrial data and business modeling experience, including a lot of work in proper handling of data input described, for example, in [17,28,36]. It is a pity that the same old mistakes in name treatment and data input handling have still been encountered in many, and varied (failed) present-day IT-related activities (some of them may be described in articles with titles referring to "data quality" or "semantic integration"). To avoid such mistakes, it is essential to base the IT work on a solid foundation that, as we have seen, can also be used for communicating with non-IT experts.

A moderately detailed model of "name in context" based on the definitions discussed above was presented, for example, in [29]. This model – as any other business model, for that matter – substantially uses a very small number of generic relationships, such as composition and subtyping. The *semantics* of these relationships was precisely defined, for example, in RM-ODP using philosophical foundations such as those described by Mario Bunge [7,9] or by F. A. Hayek [21]. Thus, business and information modeling does not start with a blank sheet of

paper, but rather reuses essential business patterns such as generic relationships. Note that various business patterns based on composition were precisely described and used by such system thinkers as Adam Smith ([55], Book One, Chapter VI — the example of a price of commodity as a composition[1] of rent, labor and profit), F. A. Hayek [20,21], Ludwig von Mises, and others.

4. On systems and relationships

All too often, existing systems or those that are supposed to be built, are represented by "experts" using various kinds of box-and-line diagrams the semantics of which is either unclear or too vague to be of any use. Although some readers of such diagrams may have a warm and fuzzy feeling about them, especially if some names used in the diagrams happen to be familiar, different readers will usually have quite different understanding of the meaning of these diagrams. Often, the narratives describing the diagrams are either unclear or concentrate only on examples (compare with the contrast between the Aristotelian and prototypical approaches to modeling referred to above). Specifying an existing system or designing a new one on such a basis leads to serious problems and often to failures.

Again, as in the name example above, let us try to exactify the concept of a system together with some related concepts. To quote Mario Bunge, "[e]very system can be analyzed into its *composition* (or set of parts), *environment* (or set of objects other than the components and related to these), *structure* (or set of relations, in particular connections and actions, among the components and these and environmental items) and *mechanism* (or set of processes peculiar to it, or that make it tick)" [8]. Moreover, analysis means "breaking down a whole into its components and their mutual relations" [9]. These approaches were not invented by Bunge — they have been known and used for centuries. For example, Walter Bagehot in *Lombard Street* observed in 1873: "[t]he objects which you see in Lombard Street, and in that money world which is grouped about it, are the Bank of England, the Private Banks, the Joint Stock Banks, and the bill brokers. But before describing each of these separately we must look at what all have in common, and at the relation of each to the others" [2]. Thus, the concepts of a relation and of composition are of essence in analysis (and system design) work. Observe also that in order to understand individuals we ought to consider relationships between them: "individuals are merely the *foci* in the network of relationships" [21]. Clearly, we need to concentrate on relationship *semantics*, and in doing so we emphasize, in particular, that a line between two boxes is not a relationship since its semantics has not been specified (also, most relationships are not binary).

From the considerations above it follows that the same kind of approach may and should be used to understand business and IT systems, and that the same concepts and constructs may be used to analyze[2] and design such systems. Furthermore, the structure of composite processes may be understood in the same manner as the structure of composite ("whole") things, so that the same kinds of relationships may be used in analyzing (and designing) both things and processes. In this manner, it becomes possible to define and use a small number of concepts and constructs for understanding and describing any kind of system. When the semantics of these concepts is defined in an explicit manner clear to both business and IT experts (so that it can be explained on a proverbial back of an envelope), these experts can easily communicate: they will use the same system of concepts for creating their explicit ontologies, in the same manner as different people use the same well-defined and easily explainable system of concepts to create and use various kinds of roadmaps, from those in nicely bound atlases to those scribbled on a napkin. And in the same manner as the semantics of every relationship between things in a roadmap is conceptually clear – or can be easily explained – to any user of a roadmap, *the semantics of every relationship* between items in an ontology *should be conceptually clear* or be easily explainable to every user of the ontology. Fortunately, this is not too difficult since the number of basic relationships is quite small, and their semantics has been clearly defined.

As noted above, a system of such concepts was specified, for example, in international standards such as RM-ODP and the General Relationship Model (GRM [27]). These approaches to creation and use of ontologies and to system analysis (and design) in general have been applied both in academia (teaching and research) and in industry — in various business and IT areas including finance, insurance, telecommunications, document management, organizational modeling, managerial decision making, business process change, metadata management, business and IT system architecture, and many others [16,36,34,29,31,33,39,44,47]. Furthermore, these approaches have substantially influenced various OMG (Object Management Group) standards, such as the Relationship Profile of the UML profile for Enterprise Distributed Object Computing, the Model-Driven Architecture (MDA), etc.

4.1. Composition

One of the most important concepts in system thinking is that of a composition relationship. It is defined in RM-ODP as "[a] combination of two or more [items] yielding a new [item], at a different level of abstraction. The characteristics of the new [item] are determined by the [items] being combined and by the way they are combined" [26]. Similar definitions have been provided by philosophers, notably, by Bunge and Hayek. It follows that the concept of *emergent* properties of the composite is the essential one to understand and use composition. Whereas

[1] See the precise definition of the composition relationship below. While Adam Smith did not use this definition explicitly, his treatment of this example clearly demonstrates his excellent understanding of composition semantics.

[2] This applies to the analysis both of business systems and of IT systems. The latter often have to be explicitly analyzed because "it does what it does" is not an adequate characterization of an IT system by its vendor (or by anyone else).

in some cases the values of these properties may be easily determined by a computer-based system (e.g., the total number of pages in a paper document composed of sections), in other – more interesting – cases the values of the emergent properties have to be determined by humans (e.g., the abstract of such a document).

We may want to distinguish between various degrees of novelty of emergent properties. In some cases, we follow Bunge's definition of emergence and look at radical novelty: "[a] property of a system is emergent if it is not possessed by any component of the system. Examples: [...] being alive (an emergent property of cells), perceiving [...] and social structure (a property of all social systems)" [9]. In certain other cases, the novelty is not radical at all (the emergent properties may, for example, result from the relationships of the composite to its environment), while in many cases the degree of novelty of emergent properties is between the two extremes: consider the various degrees of social significance in various groups of individuals as described in [45]. This degree of novelty need not be fixed: for example, we can differentiate a group of people who met for the first time for some purpose from that group that functions as a team with social and other bonds. When the latter is disbanded (for example, by management), the bonds still remain, and therefore the team as a composite (perhaps of a different kind) still exists.

A composition relationship is in most cases not binary: the invariant that determines the emergent properties of the composite refers to all its components rather than only to one of them. Moreover, in understanding and using compositions we refer to the emergent properties of the composite, and therefore "definitions" of compositions that do not refer to property determination miss the essential semantics of the defined concept. These inadequate definitions lead to failures but still have been widely used in popular IT-based (more specifically, tool-based) modeling approaches because handling various kinds of emergent properties – and of property determination in general – is not easy and more often than not cannot be automated. In this context, we observe that the definitions of the essential generic relationships – composition, subtyping and reference [31] – are all based on property determination.

A property of any system (including a software system) is emergent if it is not possessed by any of the components of that system. Any complex system can be described in this manner. The observation about *abstraction layers* in software was made as early as in the 1960s, by Dijkstra, Hoare, and others. With respect to information modeling, the same kind of observation about any system was made, for example, in [36, pp. 30–31]. Similarly, as noted above, composition has been defined as a form of abstraction in RM-ODP. Examples of emergent properties of software systems may include reliability, execution speed, complexity, maintainability, and what a system of intelligent agents "knows". Even more specific examples include: a data warehouse as a composition of databases (together with its metadata and with ways of handling data quality issues); a database as a composition of data model, DBMS, schema, files, and code (including that for handling data quality issues!)[3]; the Google® search engine as a composition of "whatever is inside"; etc. Thus, competitive advantage may be considered as an important example of an emergent property of the Google® search engine while most properties of its components (with the exception of the user interface simplicity) are not visible to its users.

In understanding systems we often distinguish between different kinds of composition. For example, the distinction between a traditional and a modern corporation as described in [13] may be exactified as the distinction between a hierarchical and a non-hierarchical composition of parts of that corporation [31]. In a similar manner, we may distinguish between a traditional and modern industry: the former is composed of industry-specific technologies, and this composition is hierarchical because the technologies pertain only to "their" specific industry, while the latter is composed of various technologies, and this composition is non-hierarchical because many technologies are not specific to that industry and thus are reused as components by various industries [31]. Clearly, such a rough draft of the ontology of an industry or of a corporation – at a very high abstraction level – may be scribbled on the back of an envelope and successfully used by (high-level) stakeholders for demonstrably reasonable decision making. Note that the semantics of all elements, and especially of *all relationships* between elements, in these back-of-an-envelope presentations is well-defined and *can be easily explained* to existing and new stakeholders.

4.2. Business patterns

Composition is not the only important concept in systems thinking.

Business and IT modeling activities should never start with a blank sheet of paper. By using business patterns (also known as templates — "specifications of the common features of a collection of [items] in sufficient detail that an [item] can be instantiated using it" [26]) and discovering where such patterns can be instantiated in various business and IT system contexts, analysts or designers make their work substantially easier and at the same time more challenging: they can handle more complex systems without the need to reinvent basic constructs over and over again. In other words, the essence of analysis and design work can be described as pattern matching in context.

Again, the concept of a (business) pattern is not new at all. It was presented, in a clear and explicit manner, by Adam Smith in his *Theory of Moral Sentiments* (1759): "When a number of drawings are made after one pattern, though they may all miss it in some respects, yet they will all resemble it more than they resemble one another; the general character of the pattern will run through them all; the most singular and odd will be those which are most wide of it; and though very few will copy it exactly, yet the most accurate delineations will bear a greater resemblance to the most careless, than the careless ones will bear to one another".

[3] Compare the structure of this composition with the structure of the previous one.

Business patterns used in ontologies – and in analysis and design in general – may be classified into fundamental (such as "invariant"), generic (such as "composition" and "subtyping"), business-generic (such as "contract") and business-specific (such as "financial derivative", or "foreign exchange option"), and if business-specific (or even business-generic) patterns are not available or not known to the analyst, it is always possible to use the less specialized generic or fundamental ones [29,31]. In applying patterns, in creating new reusable patterns (and in modeling in general!), it is essential to use abstraction ("suppression of irrelevant detail" to enhance understanding [26]) and exactification which "consists of replacing vagueness with precision [and] is attained by using, wherever necessary, the exact and rich languages of logic and mathematics instead of ordinary language, which is necessarily fuzzy and poor because it must serve to communicate people of widely different backgrounds and interests" [8]. The same approach of using abstraction and precision – together with a small number of well-defined concepts – has been successfully advocated and used by such founding fathers of computing science as E. W. Dijkstra (for example, [12]) and C. A. R. Hoare (for example, [24]). And, as emphasized by von Mises, "[e]conomics, like logic and mathematics, is a display of abstract reasoning. Economics can never be experimental and empirical. The economist does not need an expensive apparatus for the conduct of his studies. What he needs is the power to think clearly and to discern in the wilderness of events what is essential from what is merely accidental" [42].

When we understand a business (be it a traditional business or that of some IT system) and communicate this understanding to others we want to concentrate on a stable foundation – the business domain – discovered by means of pattern matching in context. These patterns do not exist in isolation: to quote Lawvere, one of the founding fathers of modern category theory, "comparing reality with existing concepts does not alone suffice to produce the level of understanding required to change the world; a capacity for constructing flexible yet reliable *systems* of concepts is needed to guide the process". In the same manner as in science we deal with laws of nature, in business we deal with "business laws" — "patterns satisfied by facts" (Bunge). Furthermore, various actions including those to be accomplished by computer-based IT systems (and described in the requirements for such systems) substantially refer to the things and relationships of the appropriate domain and therefore should be specified using the business domain model. In other words, the (relatively stable) ontology of the domain comes first, before discovering and formulating any (relatively volatile) system requirements. And in the same manner as relationships are not the same as semantic-free links connecting relationship elements, ontologies are not the same as data dictionaries.

5. (Traditional) engineering processes and artefacts

The system of common concepts described above has been successfully used not only in such a "new" area of technology as information technology (including knowledge engineering), but also in "traditional" areas of technology: after all, different areas of technology have a lot in common [7]. Of course, "traditional" engineers and technologists have often used terminology different from that of this paper, but the semantics of the systems of concepts used by them was often very similar.

As properly stressed by Mario Bunge ([7] and elsewhere) and by many other authors, successful technology is science-based rather than empirical. In fact, the concept of science-based technology was discovered and formulated by the Greeks within approximately one century, about 2300 years ago [53]. And in more modern times, a typical research and development process in engineering that leads to creation of new or updated artefacts may be represented by the following technological method: "choice of field → formulation of a practical problem → acquisition of the requisite background knowledge → invention of technical rules → invention of artefact in outline → detailed blueprint or plan → test […] → test evaluation → eventual correction of design or plan" [7]. Clearly, instantiations of this process characterize both "traditional" engineering and software engineering. Furthermore, this process requires starting from the basics of the appropriate engineering (and business) domain, as described elsewhere in this paper. While this approach is natural for a "traditional" engineer, it has not been so for buzzword-compliant software engineers. As noted in the eloquent paper by Peter Amey [1], there is, for example, a very substantial conceptual difference between diagrams used in traditional engineering and those often used in software engineering: in traditional engineering, "the coloured picture is providing an abstract view of a rigorous mathematical model", while in software engineering, "the user of a contemporary CASE tool is looking at […] a vague illustration not underpinned by anything other than the informal design decisions of the tool vendors". This contrast has important social consequences: Amey observes that "as an aeronautical engineer I am amused by the idea of applying for a job at Boeing or Airbus quoting my "skills" as: screwdriver, metric open-ended spanners and medium-sized hammers!" — the approach all too often used in hiring software engineers. Of course, this harmful conceptual gap between different kinds of engineering does not have to exist.

Thus, an engineering process starts from an ontological model that may be refined or changed later. In engineering, as in other areas of human endeavor (including business and social ones), ontologically clearer conceptual models have been shown to facilitate better problem solving within real-world application domains. Some engineers do not use the term, "ontology", when they start their process, but the semantics of starting with an ontology is there: the background knowledge essential for success includes the ontology, the laws of nature, and the social and technological constraints of the chosen application area. Some authors have already been using the ontology-related terminology (and creating and reusing engineering ontologies!): for example, [5] not only describe an ontology providing the foundation of reusable engineering model components and show how a general and abstract common framework "can be used and reused as generic building blocks in ontology construction" leading to knowledge sharing across domains, but also, for example, properly

emphasize the need for viewpoints in dealing with complex systems — a concept standardized in RM-ODP.

More recently, we have seen excellent publications that demonstrate how to bridge the conceptual gap between traditional and software engineering. For example, Dines Bjørner [4] shows that effective ("pleasing, elegant, expressive, and revealing") specifications of all kinds – from traditional IT-independent ones such as railroads to software ones such as language compilation – use the same kind of structure and approach, starting with the basics of the appropriate domain. This kind of structure and approach is very similar to the one – based on mathematics and exact philosophy – shown in this paper. Indeed, both great philosophers [7] and great software engineers [48] stressed that software engineering, knowledge engineering and information technologies are subtypes of technology, and that, in particular, "the introduction of accredited professional programs in software engineering, programs that are modeled on programs in traditional engineering disciplines will help to increase both the quality and quantity of graduates who are well prepared, by their education, to develop trustworthy software products" [48].

Of course, the approach based on using ontologies in engineering contexts has its limitations. One of the major limitations is the difficulty of *directly* implementing operational models from given starting conceptual models: "refinement" requires invention and, in fact, an IT system specification is not a direct refinement of the appropriate business specification but a composition of this business specification and the chosen or imposed technological one (based on its own ontology) [31]. Also, ontological engineering usually does not consider every aspect of reality but uses abstraction to concentrate on its most relevant aspects.

Finally, both in traditional engineering and in information technology, artefacts (products and services) do not exist in isolation but rather exist as components (subsystems) of various systems including social and cultural ones. The external structure of any system – including a traditional engineering one – is a non-empty collection of relationships between components of the system and components of its environment. The external structure of an *open system* is not fixed and possibly not completely available. The outcome of an action in the context of an open system is always uncertain — an observation made by von Mises [42,43]. In complex social systems (such as the market) the behavior of acting individuals (based on value judgments) is unpredictable, and therefore it is possible to predict only patterns rather than the specific outcomes [20]. Thus, the simplistic reductionist approach often used in traditional engineering and IT, even if based on ontologies, may fail in complex systems within their social and cultural environment.

6. Case studies

6.1. A large financial institution: two teams of experts

In a large financial institution in Europe, two teams of experts could not find a reasonable way to communicate. One of these teams was "old-fashioned" and did not recognize the value of object-oriented approach(es) to system analysis and design. The other team wanted to be as "modern" as possible and therefore wanted to base its work on the most recent object-oriented constructs. Both teams had succeeded in completing various projects, but they were unable to work together.

One of the authors of this paper (together with colleagues) was invited to help (Some specifics of this work were described in more detail in [3].). An informal but rigorous assessment of the customer environment demonstrated that members of both teams often relied in their modeling work on implicit assumptions and on "box-and-line" diagrams, that is, on diagrams with "meaningful names" in boxes and with only binary relationships that were represented by "meaningfully-named" lines. Inevitable problems due to the absence of explicit semantics were resolved in informal and often not-too-rigorous ways, leading to miscommunications even within the teams and therefore to delays in completion of projects. Nevertheless, most if not all projects had been successful because the team members were specialists of very high caliber.

We wanted to help the excellent experts.

In order to do that, we had proposed to base all modeling work on a system of methodologically-, technologically- and tool-neutral concepts, so that these concepts could not be termed "object-oriented" or "non-object-oriented". We also recommended using only those concepts that had clearly and explicitly defined semantics. And we had shown that there was no need to invent such a system of concepts because *it had already existed* and standardized by ISO for quite a while and had been based on exact philosophy and mathematics. (It was the system of concepts described in this paper.) Moreover, we proposed to reformulate the (then) most difficult problems encountered by the teams using this system of concepts and predicted that by doing so clarity and understandability would be achieved both in formulating the problems and in finding appropriate solutions.

The financial institution experts were very happy with our proposals, especially with the fact that they would be able to use a (small!) system of concepts, especially business patterns, based on philosophy and mathematics rather than on a currently buzzword-compliant tool. These patterns constituted the solid foundation of a clear language used to communicate between members of different teams.

We started with a short presentation of the system of concepts and with small examples from the environment of the financial institution. Almost immediately, it became possible to handle one of the serious problems encountered by one of the teams: by formulating precise definitions of the constructs in their contexts, and specifically, by defining the kinds of generic relationships between these constructs and other, related, ones. Specifically, we found out and were able to express in a clear manner understandable to all stakeholders the semantic distinction between what was termed "business processes" and what was termed "business functions". This semantic distinction was formulated using two different subtypes of a composition relationship: one, an ordered composition-

assembly[4] (for business processes), and another, an unrestricted composition (for business functions). The explicit formulation opened the eyes of a lot of people who did not understand the semantics of this difference before.

This modeling activity was very illuminating: we used only (the invariants of) two types of composition and of subtyping in order to ask questions about the business patterns — relationships we looked at, and convince ourselves that we were on the right track. We were able to present the complete semantics of the model in one picture of moderate complexity due to the fact that each graphical representation element we used had precise semantics. Also, we demonstrated that the same system of concepts can (and should) be used to model relationships between things and between actions — no need to reinvent any different but tool-compliant constructs for the same concepts. Thus, magic still encountered in many discussions was replaced with clearly defined models.

Another serious problem dealt with a problematic specification (a fragment of an accounting business) perceived to be precise. In that old specification, all relationships were represented (thanks to the use of a specific tool) by named binary links, so that the semantics of the relationships was unclear and often missing. In the improved specification, it became clear what links "belonged together", i.e., constituted the same relationship, and also what kind of a generic relationship it was. Again, it was not difficult to formulate the improved specification. In particular, the invariant that distinguishes basic components of an account (postings) from derived components (position amounts and positions) became specified, and the distinction between basic and derived components became clear. The relationships were more semantically rich and were not artificially restricted by a methodology or tool. Thus, the "business rules" that often are "visible only in the code" could be explicitly demonstrated in the business specification.

The substantial improvement of the old specification happened due to the following reasons:

- generic relationships (such as different kinds of composition) were made clearly distinguishable from application-specific relationships obtained by instantiating the former;
- it became possible to show *explicitly* that a relationship associates more than two participants;
- it was emphasized that relationship semantics (such as that of composition, see above) includes more important and interesting aspects – property determination – than just cardinalities.

In this manner, the emphasis in a specification is not on links anymore, because links in specifications are at a very low abstraction level and correspond to *goto*'s in programming. Rather, the emphasis is on higher-level precisely specified meaningful constructs corresponding to (for example) *while* loops and procedures with parameters in programming. The team members who have been (or had been) programmers appreciated this comparison very much, while both the programmers and the business experts appreciated the clarity and simplicity of the semantics explicitly represented in the specifications.

All team members were happy with using the system of concepts based on fundamental ideas of exact philosophy rather than on volatile ("here today, gone tomorrow") and often undefined, but methodology- and tool-compliant, constructs. The "object-oriented" and "non-object-oriented" teams could, and did, communicate very well, and of course, the same applied to the business and IT stakeholders.

Finally, by providing the customer with a system of concepts including reusable business patterns, we succeeded in empowering the customer: "[t]he role of a trainer or consultant is to empower the customer, not to make himself indispensable" (Bertrand Meyer).

6.2. A leading publishing company: Using an ontological approach to model a publisher's business environment and strategy

One of the authors was hired to assist a team of a top management consulting and systems integrating firm that had been engaged by a paperbound publisher to transition the latter from a paperbound environment to an emerging electronic environment while maintaining its dominant market position. (Some specifics of this work were described in more detail in [54]). The consultants formed a joint development team (JDT) which consisted of several of their own senior level systems analysts working in conjunction with members of the publisher's upper level executives and subject matter experts (SMEs). The JDT was formed and charged with accomplishing two objectives in a relatively short time. First, it wanted to understand the publisher's business environment and develop a new brand identity based on a suite of new, profitable Internet product and service offerings. Second, it wanted to reengineer the publishing process and adapt emerging technologies in order to reduce costs and minimize workcycle time.

From the outset of the project, it was evident that the board suffered from a lack of "sensemaking" of critical aspects of the emerging electronic environment. Who were the new competitors? What was the nature of their electronic offerings? What new physical technologies were they using? What market and business data was available and how could they understand and exploit it? What were the publisher's opportunities, threats and risks? Although the consulting company routinely handled assignments such as this one, they felt hindered by the publisher's apparent inability to understand the "outside world" and respond to questions such as these. The consultants also wanted all members of the JDT to have a shared explicit understanding of the publisher's business environment before proceeding to identify, specify, and develop a competitive suite of integrated electronic products and services.

In fact, the publisher made available an abundance of its business data and information to the consultants. Yet, the latter felt that they could not successfully develop competitive web-

[4] A composition is an assembly if the existence of a composite implies the existence of its components.

based "solutions" through the use of the standard "business sanctioned" in-house methodology previously employed in their consulting practice. This methodology was based on traditional systems development lifecycle tools and a database of best practices and lessons learned. The consultants decided to seek an "out-of-the-box" approach to modeling the complex, unstructured "ocean" of data that the client company had made available.

In a world where speed frequently dictates "everything", the JDT decided that before proceeding further they had to model the publisher's current business environment. After consultation with the author, one systems analyst proposed that an ontological approach be taken to develop a business model and strategy that would facilitate the publisher's sensemaking and establish relevant and precise communications among all members of the JDT. The analyst asked and received permission to introduce to the JDT a short overview of an ontological approach as well as the rationale for adopting it. This presentation took only about thirty minutes.

For about half of a business day, the same system analyst led a brainstorming session in which the JDT concentrated its efforts to identify business relationships that existed in the current business environment. The next day an explicit and precise business-generic model referred to as the Value Net Information Model (VNIM) was proposed by the analyst and adopted by the JDT. This underlying idea of this model is that a competitive environment is founded on semantically well-defined relationships between business players. The JDT began its modeling by identifying customer, supplier, competitor and complementor as the most important subtypes of business players.

In the next few weeks, the JDT elaborated upon and refined the initial VNIM. They created an "as is" ontological model and then proceeded to create a "to be" ontological model that represented the publisher's desired future business environment. Once the "as is" and "to be" business models were completed and accepted by the client's business stakeholders, it was not too difficult or time-consuming to formulate a transitioning business strategy.

As a means to implement that strategy, necessary IT applications and infrastructure were identified and specified again using an ontological approach that used the *same system of concepts* that had been used for the specification of the business environment. Within a short timeframe, a suite of new Internet products and services were implemented and put into operation. The publisher was successfully transitioned to the e-environment and was able to maintain its dominant marketing position which it still currently enjoys.

The members of the JDT found that having explicit and precise models helped generate conversation and identify new business issues, opportunities and risks. The executives and SMEs were able to answer the questions they previously could not, as well as satisfactorily raise and resolve business and technology issues that occurred along the way. The resulting sensemaking led to greater entrepreneurial insight and the ability to capitalize on it. The members of the JDT appreciated that the ontological approach had broken down the usual cultural and communication barriers that had previously existed between business and IT. Indeed, they widely acknowledged that the ontological approach was key to their project success. The ontological models replaced *accident (situational data) by essence (contextualized knowledge)*, provided a basic foundation for both business and IT strategies- and the payoffs for the publisher were huge!

7. Conclusion: Communication, modeling, and decision making

Communication between experts in different domains is only possible on the foundation of a joint ontology (which is often assumed implicitly), and therefore such an ontology is essential for successful communication between (traditional) business and IT experts. To create this ontology, it is necessary to use *a common system of concepts and constructs* applicable to and extensible within any specific viewpoint. Exact philosophy defines such a system that was also standardized in RM-ODP and GRM. RM-ODP demonstrates how the same foundation [26] has been successfully used to define various specification viewpoints. Similarly, Gerald Weinberg in his very pragmatic text on general systems thinking [56] states that "the student trained in general systems thinking can move quickly into entirely new areas and begin speaking the language competently within a week or so". He further notes that mastery of a person's native (plus at least one non-native) language together with mastery of mathematics are essential for success in general systems thinking; in E. W. Dijkstra's independent opinion, these are the only two necessary prerequisites of a good programmer.

Since a computer-based system has to be exact, exactification is essential in the context of such systems in information management, business and IT modeling, decision making, etc. Moreover, independently of whether computer-based systems are or will be used, decision making in any system ought to be based on clear and explicit foundations: as E. W. Dijkstra stressed, it is pondering (or modeling) that reduces reasoning to a doable amount. Caveats certainly exist, especially in business systems, such as unpredictability of human actions [42] and of the environment (open systems) [24], but it may be possible to describe, reason about, and predict at least some essential characteristics ("patterns" [20]) of such systems. In fact, it may be undesirable or impossible to provide for a very *detailed* model of a business domain and especially processes, particularly when some actions are accomplished by humans rather than by computer-based systems (or when the unpredictable context, such as the market context, is of substantial importance). At the same time, precise and *abstract* models are possible and desirable: "precise" is not the same as "detailed", and in abstract models irrelevant details are suppressed to enhance human understanding.

Various business patterns – from fundamental to business-generic to business-specific – based on exact philosophical concepts have been used to provide clarity and understandability in business and IT modeling, and thus to communicate between business and IT experts. It was not necessary (nor perhaps desirable) to use the term, "philosophy", to establish

communication between, and with, quite a few of these experts, but the semantics of concepts could be explained and immediately used in a relatively straightforward manner.

Summing up, a business modeling process – essential for designing IT systems, for making business decisions (including those about automating some fragments of the business) demonstrably based on the essentials of that business, for teaching new employees, and for various other purposes – is clearly based on the synthesis of the concepts and constructs described above. Such a process certainly cannot (and should not) be automated. At the same time, it can be successfully accomplished and certainly can be taught. It explicitly uses a small number of concepts that have been well-known for centuries and were precisely defined in literature including international standards. It starts with creating an abstract model of the relevant fragments of the *business domain*. Such a model will be (relatively) stable and will provide a framework for more volatile models of business processes, of business requirements for IT systems, etc.

A business domain model is created by modelers jointly with the business stakeholders. The discovery and creation of such a model starts with its basics, that is, with the exactification of the main things and relationships of the domain that are most often implicitly assumed and used by the business stakeholders. Such an exactification is never accomplished using a blank sheet of paper: on the contrary, a business modeler (and often an experienced business stakeholder) uses business patterns to ask questions about and to exactify the essential aspects of the business. The fundamental and basic business patterns (the definitions of which was provided by system thinkers including philosophers), such as invariants and generic relationships, are always available for *recognition* and reuse in any business domain, even if nothing else is known *explicitly* about that domain. In most cases, business-generic patterns such as contracts (also already defined by system thinkers) are also available for recognition and reuse. In those cases when business-specific patterns are not available yet, it is possible to distill and formulate them based on the existing more generic patterns. All of these patterns can be successfully used for analyzing a business (or IT) domain, that is, to quote Bunge, for breaking down the relevant business fragment into its components and their mutual relations. Business process models can be created later, using the stable framework of the well-defined business domain and the same system of reusable business patterns (recall that *the same generic relationships* are valid for business domain and business process modeling). After that, all business decisions by stakeholders can be (and have been, see below) demonstrably based on such models — rather than on handwaving, political pressure, buzzword compliance, vendor pressure, and other harmful considerations.

7.1. Future work

While the concepts described in this paper have been around, and have been successfully used both in business and IT, unfortunately this usage has not (yet) become widely accepted, especially in many IT environments. The tendency to write programs before they are designed or even before they have requirements (K. Baclawski) and before the domain within which they (will) act has been understood, is still with us. On a more positive note, standardization developments (and, implicitly, their underlying philosophical framework) described in this paper, notably, RM-ODP and GRM, have been acknowledged and even popularized in such pragmatic documents as OMG's UML Profile for Relationships [47] and OMG's Model-Driven Architecture, especially its Computation-Independent Model (CIM) that "is sometimes called a domain model". Furthermore, "[i]t is assumed that the primary user of the CIM, the domain practitioner, is not knowledgeable about the models or artifacts used to realize the functionality for which the requirements are articulated in the CIM. The CIM plays an important role in bridging the gap between those that are experts about the domain and its requirements on the one hand, and those that are experts of the design and construction of the artifacts that together satisfy the domain requirements, on the other" [46].

A lot of research and practical work ought to be done to further the approach and activities described here for understanding, specifying and designing complex systems. At the same time, the appropriate philosophical foundation has been laid, and successfully, by such thinkers as Adam Smith, Friedrich August Hayek, Mario Bunge, and others, while the corresponding computing science and information technology foundation based on mathematics and on work of such thinkers as E. W. Dijkstra, C. A. R. Hoare, D. Bjørner, and others, has also been around for a while. These foundations are conceptually clear. Business patterns referred to in this paper have been described in literature [29,31] and successfully used in many industrial projects.

It would be theoretically very interesting and practically useful to provide more complete business domain models understandable – and usable! – to various business and IT stakeholders and *explicitly* based on work of such thinkers as Hayek, von Mises, and Bunge. Some preliminary work in these areas already exists [31,32,38]. In this manner it would become blindingly obvious that both academic and practical work on ontologies, as well as on business system modeling and design ought to be accomplished not from the tools (or languages) point of view, but rather ought to be based on a sound and explicit philosophical and mathematical foundation that has existed for a long time and that has become explicitly formulated more recently (for some explicitly formulated mathematical foundations, see, for example, papers by Joseph Goguen such as [18,19]). The underlying concepts and approaches have been successfully taught to, and very positively assessed by, students with backgrounds both in business and information technology [37]. More work – including that on curricula and textbooks – is certainly needed in this area as well.

When designing and developing a good ontological infrastructure, there is no need to start from a blank sheet of paper. Very interesting work in ontology development can and should be considered, and its appropriate fragments should be discovered, abstracted out, exactified (if needed) and reused. Such formal ontologies as OpenCyc provide an excellent example of an appropriate ontological infrastructure. Of course,

OpenCyc does not represent well the semantics of many elements (and especially relationships), for example, of Bunge's materialist ontology, but many of its definitions (such as those in its part-whole vocabulary [11] can be extended to do so.

In the future an increasing amount of IT development will be done by developers who are not in physical proximity with business analysts and users, and thus the communications gap between business and IT may become wider and thus more challenging to bridge. It is imperative to exactify business semantics as well as IT semantics using a common approach — especially in those business domains in which business applications are intended to be fully or partially automated. But this challenge is not completely new either: recall that one of the triggers for formulating the concepts required to describe behavioral semantics in such standards as RM-ODP and GRM [35] was the need – imposed by law – to separate between specification and implementation in telecommunications.

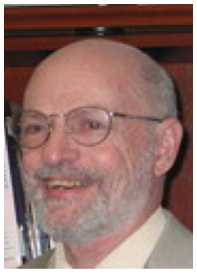
Acknowledgments

Many thanks go to our (virtual) teachers some of whom have been mentioned in the paper. Thanks also go to our colleagues and students, who provided excellent food for thought in their comments. Finally, a lot of thanks go to the organizers and participants of the ONTOSE'2005 workshop (especially, to Miguel-Angel Sicilia and Salvador Sánchez-Alonso) who by means of the online discussions substantially contributed to the clarification and improvement of the presentation of an extended abstract of this paper.

References

[1] P. Amey, Logic Versus Magic in Critical Systems, 2001, Praxis Critical Systems, 20 Manvers St., Bath, BA1 1PX, UK, praxis-his.com/pdfs/Logic_versus_magic.pdf.
[2] W. Bagehot, Lombard Street: A Description of the Money Market. Scribner, Armstrong & Co., New York, 1873.
[3] O. Bernet, H. Kilov, From box-and-line drawings to precise specifications: using RM-ODP and GRM to specify semantics, in: H. Kilov, K. Baclawski (Eds.), Practical Foundations of Business System Specifications, Kluwer Academic Publishers, 2003, pp. 99–109.
[4] D. Bjørner, Software engineering, vols. 1–3, Springer Verlag, 2006.
[5] P. Borst, H. Akkermans, J. Top, Engineering Ontologies, 1996, http://ksi.cpsc.ucalgary.ca/KAW/KAW96/borst/kaw96doc.html.
[6] M. Broadbent, P. Weill, Developing business and information strategy alignment: a study in the banking industry, in: J.I. DeGross, I. Benbasat, G. DeSanctis, C.M. Beath (Eds.), Proceedings of the Twelfth International Conference on Information Systems, 1991, pp. 293–306.
[7] M. Bunge, in: Martin Mahner (Ed.), Scientific Realism, Prometheus Books, 2001.
[8] M. Bunge, Philosophy in crisis, The Need for Reconstruction, Prometheus Books, Amherst, NY, 2001.
[9] M. Bunge, Philosophical Dictionary, Prometheus Books, Amherst, NY, 2003, Enlarged edition.
[10] Y.E. Chan, S.L. Huff, D.W. Barclay, D.G. Copeland, Business strategy orientation, information systems orientation, and strategic alignment, Information Systems Research vol. 8 (2) (1997) 125–150.
[11] CYC, 2002, http://www.cyc.com/cycdoc/vocab/part-vocab.html.
[12] E.W. Dijkstra, The teaching of programming, i.e. the teaching of thinking. In language hierarchies and interfaces, in: F.L. Bauer, K. Samelson (Eds.), Lecture Notes in Computer Science, 46, Springer Verlag, 1976, pp. 1–10.
[13] P. Drucker, The next society, The Economist, 361, 2001, p. 8246, (November 3rd–9th).
[14] C.F. Dunbar, in: O.M.W. Sprague (Ed.), Chapters on the Theory and History of Banking, Second edition, G.P. Putnams Sons, New York and London, 1901.
[15] B. Evans, Information Week, February 11, 2002.
[16] J.S. Garrison, Business specifications: using UML to specify the trading of foreign exchange options, in: K. Baclawski, H. Kilov (Eds.), Proceedings of the 10th OOPSLA Workshop on Behavioral Semantics (Back to Basics), Northeastern University, Boston, 2001, pp. 79–84.
[17] T. Gilb, G. Weinberg, Humanized Input, Winthrop Publishers, 1977.
[18] J. Goguen, Data, schema, and ontology integration, Proceedings, Workshop on Combination of Logics, Center for Logic and Computation, Instituto Superior Tecnico, Lisbon, Portugal, 2004, pp. 21–31.
[19] J. Goguen, What is a concept? in: Frithjof Dau, Marie-Laure Mugnier, Gerd Stumme (Eds.), Lecture Notes in Computer Science, Conceptual Structures: Common Semantics for Sharing Knowledge: 13th International Conference on Conceptual Structures, ICCS 2005, Kassel, Germany, July 17–22, 2005. Proceedings, vol. 3596, Springer Verlag, 2005.
[20] F.A. Hayek, The theory of complex phenomena, in: Mario Bunge (Ed.), The Critical Approach to Science and Technology (In Honor of Karl R. Popper), The Free Press of Glencoe, London, 1964, pp. 332–349.
[21] F.A. Hayek, The counter-revolution of science, Studies on the Abuse of Reason, The Free Press, Glencoe, Illinois, 1952.
[22] F.A. Hayek, The sensory order, Routledge and Kegan Paul Limited, London, 1952.
[23] F.A. Hayek, The Fatal Conceit (The Collected Works of F. A. Hayek, volume 1, The University of Chicago Press, Chicago, 1989.
[24] C.A.R. Hoare, in: P.J.L. Wallis (Ed.), Programming as an engineering profession, Software Engineering, State of the Art Report, vol. 11, No. 3, 1983, pp. 77–84.
[25] G. Hopper, Automatic Programming for Business Applications. In: Proceedings of the 4th Annual computer applications symposium, October 24–25, 1957, Armour Research Foundation, Chicago, 1957.
[26] ISO/IEC, Open Distributed Processing-Reference Model: Part 2: Foundations (ITU-T Recommendation X.902 | ISO/IEC 10746-2), 1995.
[27] ISO/IEC, Information Technology-Open Systems Interconnection — Management Information Systems–Structure of Management Information-Part 7: General Relationship Model, ISO/IEC 10165-7, 1995.
[28] W. Kent, Data and Reality, North-Holland, 1978 Also reprinted by 1stBooks, 2000.
[29] H. Kilov, Business Specifications, Prentice-Hall, 1999.
[30] H. Kilov, Back to basics, Requirements Engineering 6 (3) (2001) 200–203.
[31] H. Kilov, Business Models, Prentice-Hall, 2002.
[32] H. Kilov, Finding work: an IT expert as an entrepreneur, in: H. Kilov, K. Baclawski (Eds.), Proceedings of the OOPSLA2002 Workshop on Behavioral Semantics (Serving the Customer), Northeastern University, Boston, 2002.
[33] H. Kilov, K. Baclawski (Eds.), Practical Foundations of Business System Specifications, Kluwer Academic Publishers, 2003.
[34] H. Kilov, H. Mogill, I. Simmonds, in: H. Kilov, W. Harvey (Eds.), Invariants in the Trenches. Object-Oriented Behavioral Specifications, Kluwer Academic Publishers, 1996, pp. 77–100.
[35] H. Kilov, L. Redmann, Specifying joint behavior of objects: formalization and standardization, Software Engineering Standards Symposium, Brighton, UK, 30 Aug–3 Sep 1993, ISBN: 0-8186-4240-8, 1993, pp. 220–226.
[36] H. Kilov, J. Ross, Information Modeling, Prentice-Hall, 1994.
[37] H. Kilov, I. Sack, An innovative university course in data management for professionals, UKAIS Conference 2003, University of Warwick (UK), April 9–11, 2003, http://www.hear-see-do.com/ukais2003/auto_abstracts.asp.
[38] H. Kilov, I. Sack, Exploiting reusable abstractions in organizational inquiry: why reinvent square wheels? in: James Courtney, David B. Paradice, John D. Haynes (Eds.), Inquiring Organizations: Moving from Knowledge Management to Wisdom, Idea Group, 2005, pp. 337–359.
[39] T. Kudrass, Coping with semantics in XML document management, in: H. Kilov, K. Baclawski (Eds.), Proceedings of the Ninth OOPSLA

Workshop on Behavioral Semantics, Northeastern University, 2001, pp. 150–161.
[40] J. Luftman, (with Bullen, C., Liao, D., Nash, E., and Neumann, C.), Managing the information technology resource. Pearson Education International, Upper Saddle River, NJ, 2004.
[41] O.L. Madsen, B. Moller-Pedersen, K. Nygaard, Object-oriented programming in the BETA programming language, (Draft. August 11, 1992.)
[42] L. von Mises, Human Action: A Treatise on Economics, Yale University Press, New Haven, 1949.
[43] L. von Mises, The Ultimate Foundation of Economic Science: An essay on Method, D. van Nostrand Company, Princeton, New Jersey, 1962.
[44] J. Morabito, I. Sack, A. Bhate, Organization Modeling: Innovative Architectures for the 21st Century, Prentice Hall, 1999.
[45] W.F. Ogburn, M.F. Nimkoff, A Handbook of Sociology, Kegan Paul, Trench, Trubner & Co., Ltd., London, 1947.
[46] OMG, MDA Guide, 2003, Version 1.0.1, http://www.omg.org/cgi-bin/doc?omg/03-06-01.pdf.
[47] OMG, UML Profile for Relationships, 2004, http://www.omg.org/cgi-bin/doc?formal/2004-02-07.
[48] David L. Parnas, Software engineering programs are not computer science programs, Annals of Software Engineering 6 (1999) 19–37.
[49] C.S. Peirce, Collected papers of Charles Saunders Peirce, vols. 1–8, The Belknap Press of Harvard University Press, 1931, pp. 1931–1961.
[50] N. Raden, Start making sense: get from data to semantic integration, Intelligent Enterprise vol. 8 (10) (2005) 25–31.
[51] B.H. Reich, I. Benbasat, Measuring the linkage between business and information technology objectives, MIS Quarterly vol. 20 (1) (1996) 55–81.
[52] B.H. Reich, I. Benbasat, Factors that influence the social dimension of alignment between business and information technology objectives, MIS Quarterly vol. 24 (1) (2000) 81–113.
[53] L. Russo, The Forgotten Revolution: How Science was Born in 300 BC and why it had to be Reborn, Springer Verlag, 2004.
[54] I. Sack, A. Thalassinidis, Using information modeling to initiate business strategies — a case study for the e-publishing industry, in: H. Kilov, K. Baclawski (Eds.), Practical Foundations of Business System Specifications, Kluwer Academic Publishers, 2003, pp. 299–312.
[55] A. Smith, An Inquiry into the Nature and Causes of the Wealth of Nations, London, 1776, Printed for W. Strahan; and T. Cadell.
[56] G.M. Weinberg, Rethinking Systems Analysis and Design, Little, Brown and Company, Boston and Toronto, 1982.
[57] L. Wittgenstein, Tractatus Logico-Philosophicus, Kegan Paul, Trench, Trubner & Co. Ltd., London, 1933, 2nd corrected reprint.
[58] I.M. Yaglom, Mathematical Structures and Mathematical Modeling, Gordon and Breach Science Publishers, 1986.



Ira Sack is a full time Associate Professor of Management who has been teaching at the Leslie J. Howe School of Technology Management, Stevens Institute of Technology, in Hoboken, New Jersey (USA) for over 26 years. He has taught in executive information management programs at such firms as AT&T, Solomon Smith Barney, Paine Webber, Prudential, Johnson & Johnson, and Pearson Education, among others. Prior to his academic career, he was a full-time member of technical staff (MTS) at Bell Laboratories. His current research and publications are centered in organization modeling and organizational architecture, knowledge management, business strategy, ontologies, and diverse areas of information systems. He is a coauthor of the reference text, Organization Modeling: Innovative Architectures for the 21st Century, published by Prentice Hall. He served both as the principal investigator for two research grants and as a research consultant to the National Agency for Space Administration (NASA) and elsewhere.