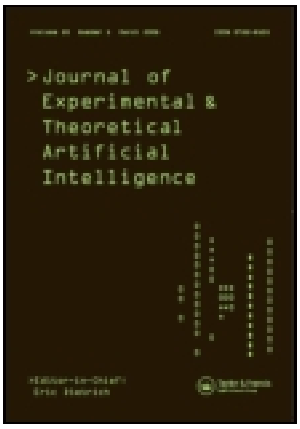
Philosophical reflections on diagram models and diagrammatic representation

Article *in* Journal of Experimental & Theoretical Artificial Intelligence · September 2012

DOI: 10.1080/0952813X.2012.693665

Koray Karaca
University of Twente
26 PUBLICATIONS **205** CITATIONS



Journal of Experimental & Theoretical Artificial Intelligence
Publication details, including instructions for authors and subscription information:
http://www.tandfonline.com/loi/teta20

Philosophical reflections on diagram models and diagrammatic representation

Koray Karaca [a]
[a] Interdisciplinary Centre for Science and Technology Studies, University of Wuppertal, Gauss str 20, 42119 Wuppertal, Germany

Version of record first published: 04 Sep 2012

To cite this article: Koray Karaca (2012): Philosophical reflections on diagram models and diagrammatic representation, Journal of Experimental & Theoretical Artificial Intelligence, 24:3, 365-384

To link to this article: http://dx.doi.org/10.1080/0952813X.2012.693665

Journal of Experimental & Theoretical Artificial Intelligence
Vol. 24, No. 3, September 2012, 365–384

Philosophical reflections on diagram models and diagrammatic representation

Koray Karaca*

Interdisciplinary Centre for Science and Technology Studies, University of Wuppertal, Gauss str 20, 42119 Wuppertal, Germany

(*Received 26 March 2012; final version received 29 April 2012*)

Diagram models are widely used in system and software engineering to design and analyse information systems. In this article, I examine diagram models within Morgan and Morrison's 'models as mediators' account. I argue that even though diagram models and theoretical models differ in terms of representations they deliver, namely diagrammatic and sentential representations, respectively, as theoretical models mediate between the high-level theory and the real world, diagram models mediate between the abstract system and its concrete realisation in the world. Diagram models play this mediating role through the diagrammatic representation of the dynamic and static features of the system under consideration. Moreover, I argue why diagrammatic representation proves more efficient than sentential representation as a tool of design and analysis of information systems.

Keywords: system and software engineering; diagram models; diagrammatic representation; sentential representation; models as mediators; locality feature of diagrams

1. Introduction

The use of visual, non-propositional, devices – such as diagrams, graphs, pictures and tables – in science is widespread. For example, science journals, books, textbooks as well as research presentations usually contain various kinds of visual representations that supplement the content of the text in various ways. Despite their widespread use, however, the study of visual modes of representation had been largely neglected in philosophical studies of scientific representation up until the 1980s. This had been largely due to the prevailing 'language-based conception' of science that conceived of scientific discourse as a solely and inherently linguistic process encapsulated in the statements of laws of nature in particular and in scientific theories, hypotheses and models at large.[1] However, over the past three decades or so, scholars of different research traditions have offered a number of diverse studies dealing with the use and role of visual displays in different branches of science.[2] In these studies, it was pointed out that non-sentential, visual modes of scientific representation are as essential as sentential (propositional) modes to the production, confirmation and dissemination of scientific knowledge.

The main task of this article is to philosophically examine 'diagram models' that have not yet received the attention of philosophers of science despite their widespread use in

*Email: karacak@gmail.com; karaca@uni-wuppertal.de

ISSN 0952–813X print/ISSN 1362–3079 online
© 2012 Taylor & Francis
http://dx.doi.org/10.1080/0952813X.2012.693665
http://www.tandfonline.com

system and software engineering (SSE), which is basically concerned with the analysis and design of information systems. This neglect might be traced to the fact that, like other relevant debates in philosophy of science, the model debate has also largely centred on the study of sentential modes of scientific representation.[3] Having introduced their necessary technical features, I shall seek to characterise the essential features of diagram models within Morgan and Morrison's (M&M 1999) account of models, namely 'models as mediators'. Moreover, revisiting the relevant literature in cognitive science and philosophy, I shall seek to argue why diagrammatic representation proves more efficient than sentential representation as a tool of design and analysis in the context of SSE.

2. Diagrammatic modelling in SSE

At present, there exist two different but complementary methodologies about diagrammatic modelling in SSE. One is called *structured system analysis and design* (SSAD) and uses the 'process oriented' approach to diagrammatic modelling. The other is called *object-oriented analysis and design* (OOAD) and uses the 'object oriented' approach. In the next two sections, I shall, respectively, introduce SSAD and OOAD and examine various types of diagram models that are associated with them. The subject matter under consideration is so extensive that it is not necessary to consider all types of diagram models available in SSE. For that reason, my analysis will be selective but geared to capture the essential features of diagrammatic modelling.

2.1. *Process-oriented diagrammatic modelling*

SSAD was originally developed in the late 1970s by software engineers such as DeMarco (1978), Gane and Sarson (1979) and Yourdon and Constantine (1979). It essentially consists of a group of methods used in SSE to graphically depict the movement of data and the associated processes throughout an 'information system', which includes a wide range of applications such as a patient care system or a weather forecast system. *Nasa System Engineering Handbook* defines an information system to be 'a set of interrelated components which interact with one another in an organised fashion toward a common purpose. The components of a system may be quite diverse, consisting of persons, organisations, procedures, software, equipment, and/or facilities' (Aster and Shishko 1995, p. 3). SSAD essentially aims at a graphical modelling of an information system through the identification of its constituent processes and entities that produce, transform, manipulate and store data that flow within this system.[4] What is meant here by 'process' essentially concerns actions performed on data so that they are produced, transformed, stored and distributed. The major tool used in SSAD is what is called a 'data-flow diagram' (DFD) that enables to graphically model how data flow through an information system, the relationships among data flows as well as how data come to be stored at specific locations in a system. DFDs also show the processes that change or transform data. Since DFDs primarily concern the movement of data among processes, these diagrams are also called 'process models' (also called 'function' or 'task' models). In this sense, the type of modelling DFDs illustrate is often referred to as 'process modelling' in the literature of SSE (Hoffer, George, and Valacich 2008, pp. 206–207).

A DFD consists of several graphical units; namely 'process', 'data-store', 'source/sink' and 'data-flow'. Each unit is represented on a DFD by a specific symbol as shown

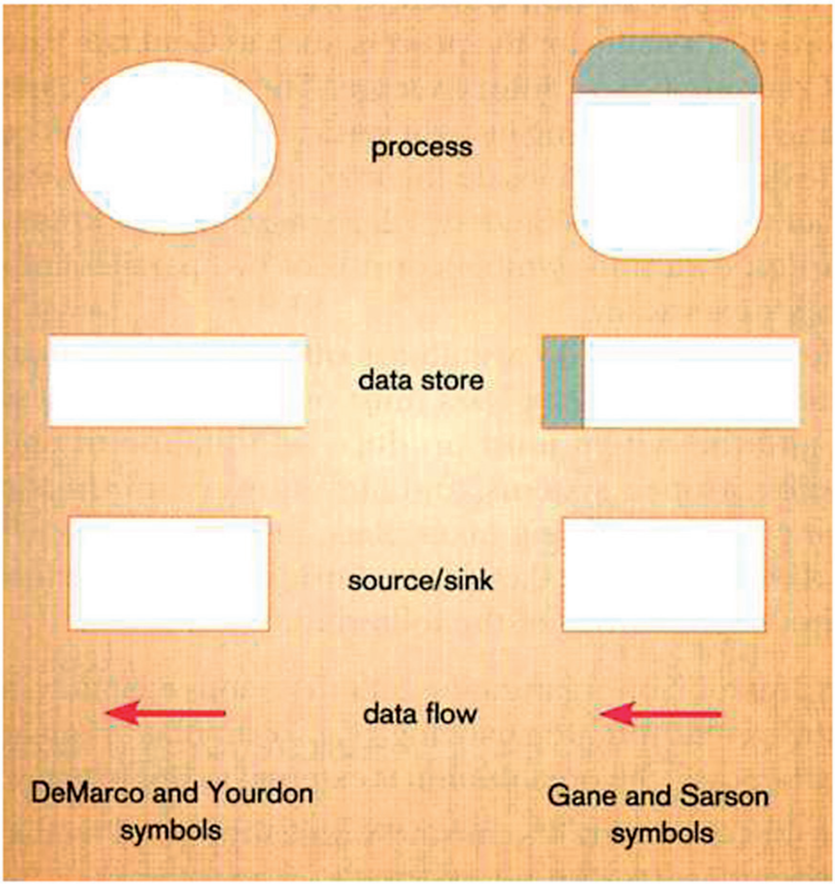


Figure 1. DFD symbols in SSAD.
Source: Hoffer et al. 2008, p. 209. Reprinted by permission of Pearson Education, Inc., Upper Saddle River, NJ.

in Figure 1. As is also noted in this figure, there are two main symbol conventions currently used in SSAD as to how to build a DFD; one is proposed by DeMarco (1978) and by Yourdon and Constantine (1979) and the other is by Gane and Sarson (1979). ‘Data store’ on a DFD denotes where data are at rest in a system, ‘data flow’ denotes data in motion from one place to another. The term ‘source’ or ‘sink’ – also referred to as ‘external entities’ – is used on a DFD to denote the origin and/or destination of data in a system (Hoffer et al 2008, pp. 209–210).

In order to illustrate how the above terminology is used to build DFDs as well as to introduce other rules associated with them, I shall make use of a simple but illustrative example.[5] Let us imagine a restaurant that uses an information system that takes customer orders, sends the orders to the kitchen, monitors goods sold and inventory and generates reports for management. This information system can be represented by the DFD shown in Figure 2. This gives the *highest level* view of the system and is called a ‘context diagram’. The latter is defined to be a graphical depiction of the boundaries of a system, the external entities that interact with the system and the major information flows between the entities and the system. The context diagram shown in Figure 2 contains only one process, namely

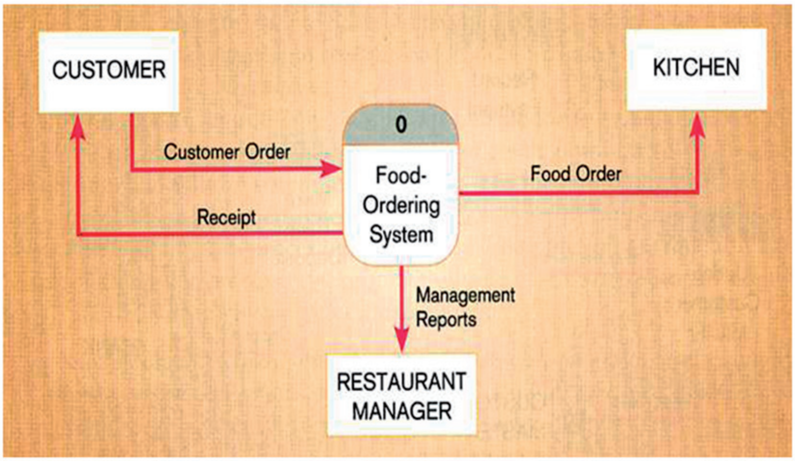


Figure 2. Context diagram of a restaurant.
Source: Hoffer et al. 2008, p. 212. Reprinted by permission of Pearson Education, Inc., Upper Saddle River, NJ.

'food ordering', no data stores, four data flows and three sources. It is important to note that in SSAD, by convention, all context diagrams must contain *only* one process. Another important convention used in SSAD is that internal data stores, namely data stores associated with the system, unlike the external ones, do not appear on a context diagram. The context diagram shown in Figure 2 has three sources; namely 'customer', 'kitchen' and 'restaurant manager', which altogether represent the external entities that interact with the food-ordering system under consideration. Note that in the same context diagram four different types of data are shown; namely: 'customer order data', 'receipt data', 'food order data' and 'restaurant management data'. A *uni-directional* arrow represents the flow of each data type between the system and the external entities.

The DFD shown in Figure 3 is called a 'level-0 diagram' in SSAD. It contains the *primary individual processes* in the food-ordering system at the highest possible level. Each process has a number that ends in '.0', corresponding to the level number of the DFD. The primary individual processes represented on the level-0 DFD shown in Figure 3 include (Hoffer et al. 2008, p. 212):

- capturing data from different sources (e.g. Process 1.0),
- maintaining data stores (e.g. Processes 2.0 and 3.0),
- producing and distributing data to different sources (e.g. Process 4.0) and
- high-level descriptions of data transformation operations (e.g. Process 1.0).

It is to be noted that the level-0 diagram in Figure 3 is obtained from the context diagram in Figure 2 by following the rule of 'functional decomposition'. This rule can be understood as the decomposition of a larger process into its constituting sub-processes (creating sub-systems of a larger system), which may in turn be broken down into smaller processes (sub-systems). Functional decomposition continues until one reaches the point at which no sub-process can be broken down any further (Hoffer et al. 2008, p. 216). 'In general, a level-*n* diagram is a DFD that is generated from *n* nested decompositions from a level-0 diagram' (Hoffer et al. 2008, p. 217). For example, the diagram shown in

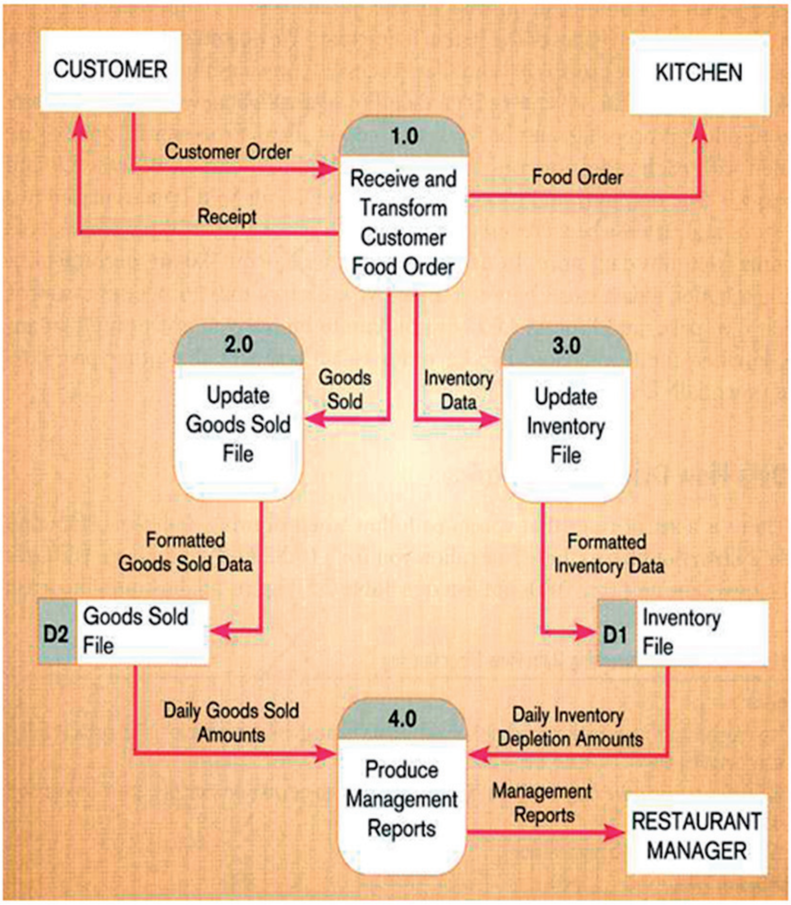


Figure 3. Level-0 DFD of a restaurant.
Source: Hoffer et al. 2008, p. 213. Reprinted by permission of Pearson Education, Inc., Upper Saddle River, NJ.

Figure 4 is a 'level-1' DFD that was obtained by decomposing the 'process 1.0' of the level-0 DFD shown in Figure 3.[6]

2.2. *Object-oriented diagrammatic modelling*

Since DFDs mainly focus on the identification of processes and data flows among these processes, they are not designed to represent how different sources or sinks interact in *real time* with each other. In SSE, this is provided by what are called 'sequence diagrams' and 'communication diagrams' that are based on OOAD. Unlike SSAD, OOAD offers a graphical representation in terms of the 'classes' and 'objects' found in the system under consideration.[7] In OOAD, the term 'object' is basically defined to be a tangible entity that has a well-defined behaviour, in that it performs a specific task whenever it is called upon by a message to do so. The term 'object class' (or shortly 'class') denotes a group of objects that have the same attributes, relationships and operations (Booch et al. 2007, p. 20 and

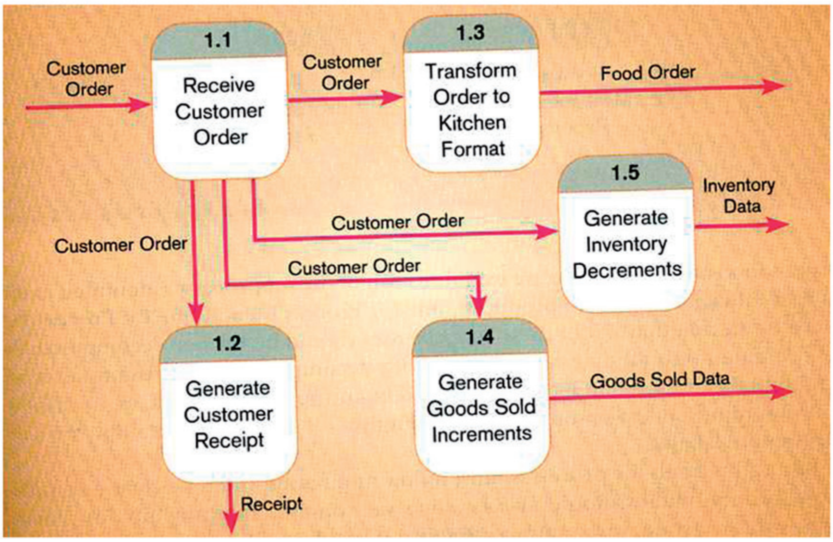


Figure 4. Level-1 DFD of a restaurant.
Source: Hoffer et al. 2008, p. 217. Reprinted by permission of Pearson Education, Inc., Upper Saddle River, NJ.

pp. 92–94; Hoffer et al. 2008, pp. 321–322). Below, I shall examine in turn ‘sequence’, ‘communication’ and ‘class’ diagrams.

2.2.1. *Sequence diagrams*

A sequence diagram involves a graphical depiction of the ways in which a particular set of objects interact within an information system during a certain period of time in order to achieve a common purpose. Like DFDs, sequence diagrams are built according to certain rules and conventions (Booch et al. 2007, sec. 5.8; Hoffer et al. 2008, pp. 268–275). As shown in Figure 5, a typical sequence diagram has two axes. Time increases down the vertical axis. The objects under consideration are shown by rectangular boxes along the horizontal axis at the top of the diagram. What is called the ‘lifeline’ of an object is shown by either a dashed or a solid line emanating from that object and lying along the vertical axis. Each *thin* rectangular box emanating from an object and superimposed on the lifeline of the same object represents the time period during which an object performs a task. In OOAD, objects communicate with each other by sending messages. On the sequence diagram shown in Figure 5, each message is shown by a solid arrow drawn horizontally and pointing from the object sending the message towards the object receiving it.[8] Return messages, unless they help understanding the communication between objects, are not shown on sequence diagrams, as they unnecessarily complicate diagrams. Message scripts are generally put just above the message arrows on sequence diagrams. Commented but short scripts of the scenario for which the sequence diagram is designed may also be written to the left of a sequence diagram as shown in Figure 5, which describes a climate planning scenario. The sequence diagram shown in Figure 5 contains four different objects, namely ‘Gardening Plan’, ‘Environmental Controller’, ‘Heater’ and ‘Cooler’. The return messages are implicit. When the diagram is read from left to right and from top to

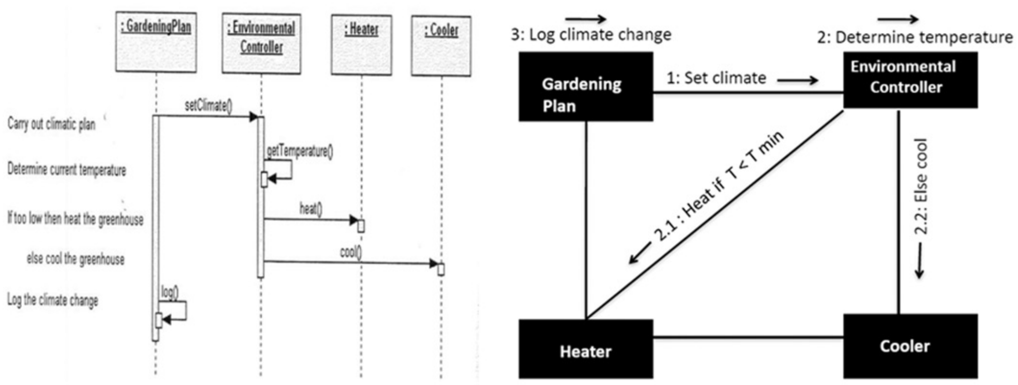


Figure 5. Sequence diagram (on the left part) and communication diagram (on the right part) for a climate planning scenario.
Source for the diagram on the left part: Booch et al. 2007, p. 210. Used with permission.

down together with the message scripts written to the left, one obtains the sequence of interactions, or scenario, for which the diagram is drawn.

2.2.2. *Communication diagrams*

The information contained in a sequence diagram can be transferred without any loss to what is called a 'communication' (also called 'collaboration') diagram in OOAD (Booch et al. 2007, sec. 5.14). However, while sequence diagrams are primarily designed to visualise the *order* in which the messages are sent among objects in a system, communication diagrams are primarily designed to visualise the ways in which objects collaborate with each other to achieve a common purpose. Figure 5 shows the communication diagram of the scenario described in the previous sub-section. Since communication diagrams do not focus on the order of communication among objects, they do not contain a time axis. Rather, the messages among objects are shown by number tags and by text-labelled arrows. The increasing order of numbers indicates the time order in which the messages in the system are exchanged among objects. If a message prompted other messages in the system, the latter are nested inside of the former. Each box denotes a particular object and the name of the object appears in this box.

2.2.3. *Class diagrams*

Four kinds of relationships among classes are defined in OOAD; namely association, generalisation, aggregation and composition (Booch et al. 2007, chap. 3 and sec. 5.7; Hoffer et al. 2008, pp. 321–335). Of these four kinds of class relationships, association is the most general one and indicates that two independent object classes associate with each other in some manner. *Multiplicity* indicates the degree of association; i.e. the number of objects that participate in a given association relationship between any two classes. Three kinds of multiplicity can be defined, namely 'one-to-one', 'one-to-many' and 'many-to-many'. For example, the multiplicity of the association relationship between the class *Country* and the class *Capital-City* is one-to-one. Aggregation relationship between any two classes denotes a 'part of' relationship between the objects of those classes. Given any

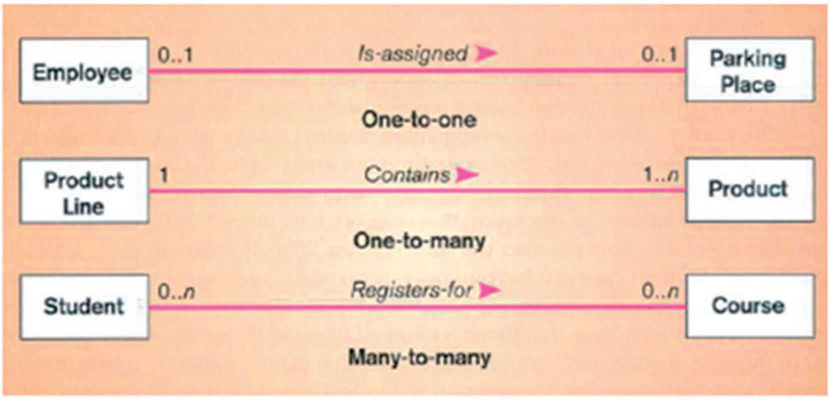


Figure 6. Examples of association relationships of different multiplicities.
Source: Hoffer et al. 2008, p. 324. Reprinted by permission of Pearson Education, Inc., Upper Saddle River, NJ.

two classes A and B, if each object of B is part of another object of A, then A and B are in an aggregation relationship and they are called the whole (also called 'aggregate') and the part, respectively. For example, there exists an aggregation relationship between the classes *Wheel* and *Automobile*, the former being the part and the latter being the whole; because a wheel is part of an automobile. Composition is a stronger form of aggregation relationship in that each instance of a class is only a part of an instance of another class. That is to say, the multiplicity on the aggregate end is 1. For example, a room is only a part of a building; so the class *Room* is the part and the class *Building* is the aggregate. Generalisation expresses an 'is a' relationship between a *subclass* and its *superclass*. For example, an apple is a fruit; so, there exists a generalisation relationship between the classes *Apple* and *Fruit* in such a way that the former being the subclass and the latter the superclass.

Class diagrams are designed to graphically depict various class relationships as stated in the previous paragraph. On a class diagram, a class icon is a rectangle box that consists of three compartments. The one at the top contains the name of the class, the middle one contains the list of its attributes and the bottom one contains the list of its operations. An attribute of a class is a property shared by all its member objects, and an operation is a function exhibited by those members. It is typical that in class diagrams, the bottom two compartments are omitted; if necessary, only attributes and operations that are relevant to the diagram are shown. A class relationship is navigable in both directions, i.e. *bidirectional*, unless it is restricted by a *navigability* arrowhead that indicates a *unidirectional* navigability. Also, class diagrams may include textual labels that denote contents of relationships that exist between classes.

The first diagram in Figure 6 shows a 'one-to-one' association relationship between the classes *Employee* and *Parking Place*. The multiplicity of this association relationship can be 0 or 1, meaning that each employee can be assigned at most to one parking place. An arrowhead and the textual label 'is-assigned' indicate the direction and content of the association relationship, respectively. The other class diagrams in Figure 6 represent, respectively, class association relationships of multiplicity 'one-to-many' and 'many-to-many', where *n* indicates that an unlimited number of objects can participate in the given

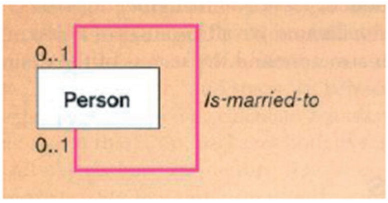


Figure 7. Example of self-association relationship.
Source: Hoffer et al. 2008, p. 324. Reprinted by permission of Pearson Education, Inc., Upper Saddle River, NJ.

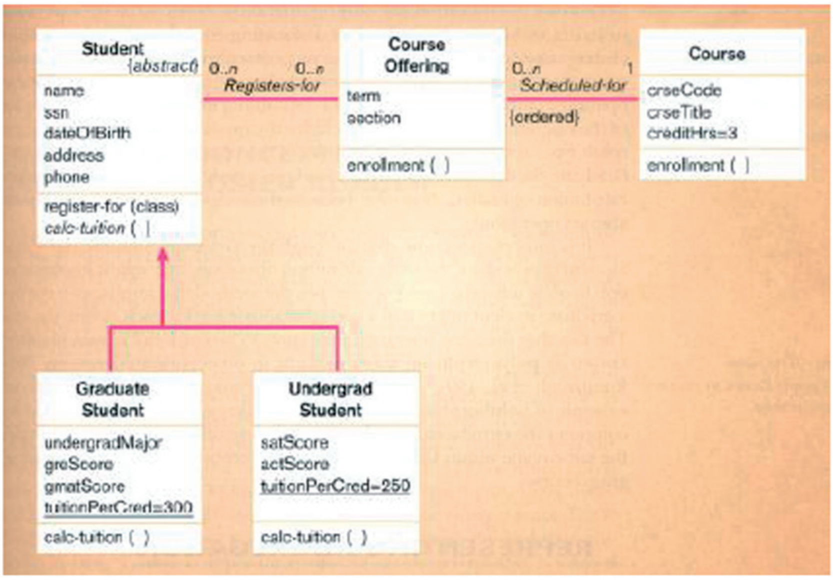


Figure 8. Examples of association and generalisation relationships.
Source: Hoffer et al. 2008, p. 329. Reprinted by permission of Pearson Education, Inc., Upper Saddle River, NJ.

association relationship. Instead of *n*, sometimes the symbol '*' is also used to indicate multiplicity. A class may have an association with itself as shown in Figure 7, where the class *Person* is in an association relationship with itself, the relationship being that each person is either unmarried or married to another person. This is a one-to-one association relationship with multiplicity 0 or 1.

Figure 8 involves a class diagram containing both association and generalisation relationships. A 'many-to-many' association relationship exists between the classes *Student* and *Course Offering*. In addition, a generalisation relationship exists between the classes *Student* and *Graduate Student* and *Undergrad Student*. On class diagrams, the direction of a generalisation relationship is shown by an arrow. In this example, the arrow indicates that the generalisation relationship is towards the class *Student*; meaning that the objects of the classes *Graduate Student* and *Undergrad Student* are students. Note that on this diagram the attributes and operations of each class are also listed.

The class diagram in Figure 9 illustrates aggregation and composition relationships that are shown by hollow and solid diamond symbols, respectively. On this class diagram,

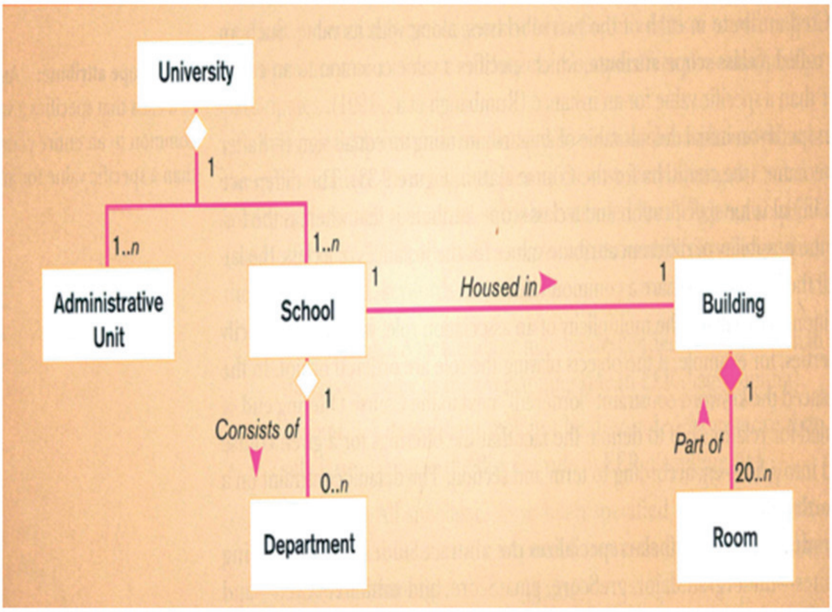


Figure 9. Examples of aggregation and composition relationships.
Source: Hoffer et al. 2008, p. 330. Reprinted by permission of Pearson Education, Inc., Upper Saddle River, NJ.

there exists an aggregation relationship between the class *University* and the classes *Administrative Unit* and *School*. The multiplicity of this relationship is 'one-to-many'. Similarly, an aggregation relationship exists between the classes *School* and *Department*. However, the relationship between the classes *Building* and *Room* is of composition type, and its multiplicity at the aggregate end is 1.

3. A philosophical perspective on diagram models

In this section, I shall examine the essential features of diagram models by drawing on M&M's 'models as mediators' account that has recently received widespread attention in the model debate in philosophy of science. According to this account, models are *autonomous* agents, in that '(1) [they] *function* in a way that is partially independent of theory and (2) in many cases they are *constructed* with a minimal reliance on high level theory' (M&M 1999, p. 43). M&M argue that by virtue of this partial independence models can mediate between the theory and the real world. In their view, via this mediating role, models function as tools or instruments that enable us to learn more about theories and the real world. An important consequence of M&M's account is that the features underlying the construction and the manipulation of models can be categorically investigated with respect to how models 'are constructed, how they function, what they represent and how we learn from them' (M&M 1999, p. 10). These four categories defined in M&M's account are originally intended to characterise the essential features of theoretical models. Since theoretical models are by their very nature expressed in sentential language, the representations they deliver with regard to natural phenomena are *sentential*.

However, in what follows, I shall argue that the aforementioned categories are also useful to identify and characterise the essential features of diagram models.

We have previously seen that in the context of SSE there exist two main approaches to diagrammatic modelling. The construction of DFDs is based on the process-oriented modelling and those of communication/sequence and class diagrams are based on the object-oriented modelling. The process-oriented modelling emphasises the representation of processes and of the transformations of data sets in information systems. On the other hand, the object-oriented modelling emphasises the representation of communication among objects and of relationships between object classes. In the first place, this indicates that the constructional features of DFDs are distinct from those of sequence/communication and of class diagrams. Note also that this difference in terms of constructional features results in a difference in terms of representational features, in that while DFDs represent processes and associated data flows, sequence/communication and class diagrams represent, respectively, communications among objects and class relationships among object classes. Note also that data flows among processes and communications among objects are 'dynamic' (or behavioural) features of a system; whereas class relationships – i.e. association, generalisation, aggregation and composition – are 'static' features. The above discussion suggests that what we learn from diagram models discussed in Section 2 of this article are dynamic and static features of information systems; namely what processes exist in the system, what data flows exist among those processes, how constituting objects communicate with each other and what kinds of class relationships exist among them.

M&M (1999, p. 11) identify 'representation as the main mechanism that enables us to learn from models'. The discussion in the preceding paragraph suggests that M&M's observation also holds true for diagram models, in the sense that they are informative about the dynamic and static features of information systems, which are conveyed through diagrammatic representations. Another point suggested by the preceding discussion concerns the division of 'representational labour' among diagram models. That is to say, each diagram model is aimed to represent a different feature (dynamic or static) of the system of interest. It is worth noting that this is analogous to the division of representational labour among scientific models, which represent different aspects of natural phenomena in different ways.[9]

I have so far dealt with the question of what we learn from diagram models. However, what is also *epistemologically* of interest is how this learning takes place in practice (M&M 1999, pp. 11–12). In SSE, diagram models are built either to design a new system or to analyse an existing system to optimise it. During the stage of design, through the use of diagram models, the model builder develops the dynamic and static features of a system to be implemented in the real world. During the stage of analysis, one analyses through diagram models an existing system to learn its static and dynamic features that altogether constitute its knowledge. That is, the knowledge acquired through the analysis of diagram models during the analysis stage essentially consists of the static and dynamic features of the system of interest, which the model builder has envisaged during the design stage. Therefore, diagram models serve a dual function in SSE; through diagrammatic representations they deliver, they 'are both a means to and a source of knowledge' (M&M 1999, p. 35). In the context of design, diagrammatic representation constitutes the medium through which the static and dynamic features of the system to be implemented are developed. On the other hand, in the context of analysis, diagrammatic representation constitutes the medium through which the static and dynamic features of an existing

Table 1. The features of diagram models with respect to construction, function, representation and learning.

Model type	Construction	Function	Representation/learning
DFDs	Process-oriented modelling	Instruments of system design and analysis	Processes and associated data flows: dynamic features
Sequence/ communication diagrams	Object-oriented modelling	Instruments of system design and analysis	Communications among constituting objects: dynamics features
Class diagrams	Object-oriented modelling	Instruments of system design and analysis	Class relationships among constituting objects (association, generalisation, aggregation and composition): static features

system are learnt. This means that while diagram models are used as tools of design in the development of a new system, they are used as tools of analysis when an existing system is analysed for its static and dynamic features. Therefore, as M&M (1999, p. 11) remarked for the case of theoretical models, diagram models function as tools and serve different functions in different contexts. Table 1 summarises the features of diagram models in terms of the categories defined in M&M's account.

An important difference between theoretical models and diagram models is that, *generally*, both the construction and the interpretation of the latter are *essentially* independent of scientific theories. In this sense, the mediating role M&M think of with regard to theoretical models does not apply *directly* to diagram models. However, as I shall argue, diagram models play a different kind of mediating role in the context of SSE. The above discussion has shown that in this context the model builder envisages through diagram models a system that may later be implemented and realised in the real world. That is, during the design phase, the system whose dynamic and static features are represented by a set of diagram models is just a *fiction*, i.e. an imaginative creation that is not yet actualised. However, after the system has been actualised, the same set of diagram models utilised in the design phase represents the static and dynamic features of a *real* system and thus can be used for analysis purposes. In this sense, diagram models mediate between the abstract system in the designer's mind and the concrete realisation of this abstract system in the real world. It is exactly by virtue of this mediating role that diagram models are able to play the aforementioned dual function between the context of design and the context of analysis, and thus they can function as instruments of both analysis and design in SSE.

4. Representation matters: the differences between sentential and diagrammatic representations

The issue of how to characterise the differences between diagrammatic and sentential representations has been long debated in the literature of cognitive science.[10] So far in this debate, the cases studied to illustrate the claimed differences between these two modes of knowledge representation have concerned relatively simple examples of diagrams, such as those used in science textbooks. Section 2 of this article has demonstrated that diagram models are relatively more complex, refined and standardised diagrams than those considered in the aforementioned debate. Also, their range of applications is much greater; making those models an important and novel case study to understand the differences between diagrammatic and sentential representations, as well as to understand why the former proves more efficient as a tool of both design and analysis in the context of SSE. In what follows, I shall address this issue by revisiting the relevant literature in cognitive science and philosophy.

In a paper published in 1971, which aimed to illustrate how interactions between philosophy and artificial intelligence may be useful for both disciplines, Sloman (1971, sec. 4) made a distinction between which he called 'analogical representations', which are offered by pictures, maps, photographs, diagrams, etc., and 'sentential representations',[11] which consist of a description in some language, including human natural languages, computer programming languages and logic languages such as first-order predicate logic. According to Sloman's definition of this distinction, the structure of the analogical representation of a system gives information about the structure of the real target system, in such a way that the properties of and the relations between parts of the representing object (i.e. picture, map, diagram, etc.) represent properties and relations of parts in the real system. By contrast, the structure of propositional representations need not correspond to the structure of what they represent. For example, the 'phrase "the city 53 miles north of Brighton" contains the symbol "Brighton" as a part, but the thing denoted does not contain the town Brighton as a part. The thing denoted, London, has a complex structure of its own, which bears no relation whatsoever to the structure of the phrase' (Sloman 1975, p. 165).

In a now seminal paper entitled 'Why a Diagram is (Sometimes) Worth Ten Thousand Words', Larkin and Simon (1987, p. 68) contrasted diagrammatic and sentential representations, which they defined as follows:

- A data structure in which elements appear in a single sequence is what we will call a *sentential representation*.
- A data structure in which information is indexed by two-dimensional location is what we call a *diagrammatic representation*.

Echoing Sloman's distinction between analogical and sentential representations, Larkin and Simon (1987, p. 66) argued that the 'fundamental difference between our diagrammatic and sentential representations is that the diagrammatic representation preserves explicitly the information about the topological and geometric relations among the components of the problem, while the sentential representation does not'. Here, of course, as also remarked by Kulpa (1994, p. 77), one should understand Larkin and Simon as indicating a structural (i.e. in terms of spatial relations contained) *correspondence* between a diagram and what this diagram represents, rather than an *exact match* between these two. Larkin and Simon suggest that when diagrammatic and sentential representations are *informationally equivalent* (i.e. if all of the information in the one is also inferable from the other, and vice versa), the former can be *computationally* more efficient; i.e. it requires less complex reasoning and effort to access and use the necessary information. To support this claim, they first point out that diagrammatic representations, unlike sentential representations, explicitly preserve spatial relations associated with a problem situation. This substantially reduces the need to use and to match symbolic labels in the representation as well as in the search of information. Moreover, in Larkin and Simon's (1987, p. 98) account, diagrams have the ability to group together all *related* pieces of information that must be used to make a problem-solving inference, as opposed to sentential representation where pieces of information may be separated in a list of statements. This feature of diagrammatic representation, to which Koedinger and Anderson (1990, p. 517) referred to as 'locality feature', significantly reduces the search for information to make a problem-solving inference.

Koedinger (1992) offered an extended account of the advantages of diagrammatic representation over sentential representation. First, he suggested that diagrams have computational *emergent properties* due to geometric relations they support. This means that the process of constructing a diagram can reveal certain properties of the problem situation that are not yet part of its original understanding. According to Koedinger, the emergent properties of diagrams can substantially reduce the problem space when conjunctive information, i.e. more than one piece of information, is represented. Second, Koedinger suggested that whole-part relations in diagrams can specify certain structural constraints that can be used as a guide to efficient knowledge organisation. In Koedinger's (1992, p. 156) view, a 'problem solver that is either given or can learn such a knowledge organisation is likely to be more efficient than a problem solver whose knowledge is organised via sententially-based abstractions'.

In a similar vein to the accounts offered by Koedinger and by Larkin and Simon, Shin suggested that in terms of the representation of *conjunctive information*, there is a fundamental difference between diagrammatic and sentential representations. Shin (1994, chap. 6) pointed out that the way a diagram represents conjunctive information heavily relies on our *perceptual* abilities; as a result, fewer conventions are involved in diagrammatic representations than in sentential representations. As an example, Shin considered the following sententially represented conjunctive information about a problem situation: the object A is to the left of the object B, and the object C is to the right of the object B. Here, the information that A is to the left of C becomes obvious and does not require additional perceptual inference when it is represented on a diagram. In this example, that A is to the left of C is an *emergent* property of a diagram that represents the above conjunctive information. However, in sentential representation, that A is to the left of C is not obvious and has to be inferred from the given conjunctive information.

So far in this section, we have seen that both the accounts offered by Sloman and by Larkin and Simon distinguish between diagrammatic and sentential representations in that while the former explicitly preserve spatial relations they represent, the latter do not. However, given that diagrams are designed to represent also *non-spatial* relations, e.g. temporal relations, the above distinction turns out to be *not* very successful. Nor is it helpful to understand why diagrammatic representation is often preferred over sentential representation in many contexts. It is to be noted that due to the very nature of diagrammatic representation representing relations on diagrams need to be *spatial*. Therefore, it is of interest to ask how non-spatial relations are represented on diagrams and how, and whether, this makes them computationally more efficient over sentential representations.

This question was *partly* addressed by Shin (1994, sec. 6.2) in the context of relatively simple examples of diagrams such as family trees, timetables, pie charts, etc. Shin suggested that diagrammatic representations transform non-spatial relations into spatial relations and that this process *typically* requires much fewer syntactic or symbolic devices as compared to sentential representations. In her view, the more a representation system employs syntactic or symbolic devices, the less efficient it is; as those devices and their associated conventions must be learnt beforehand in order to understand what sentential representations really represent.

5. The advantages of diagrammatic representation over sentential representation in the context of SSE

In this section, taking stock of the previous conclusions, I shall examine how diagram models represent spatial and non-spatial relations. To this end, I shall first consider DFDs wherein a data flow indicates the movement of data from one particular location in the system to another. So, a data flow denotes a *spatial* relation between processes as well as sources and data-stores in a system. We know from Section 2 that data flows are represented on DFDs by appropriately *text-labelled* solid arrows pointing from the object from which data originate towards the object to which data are transferred. Note that solid arrows are *symbolic* devices; whereas text labels indicating contents of data are *syntactic* devices. Therefore, data flows, which denote spatial relations in a system, are represented again as spatial relations on DFDs through the use of both symbolic and syntactic devices.

Next, I shall consider sequence/communication and class diagrams that, respectively, represent communications among objects and relationships among object classes. Remember that communication (or messaging) concerns the movement of a message from one particular object to the other in a system. In this sense, it denotes a spatial relation. In addition, the sequence of messaging is also represented in sequence and communication diagrams. Since this denotes the time order of messages between objects, one can say that sequence and communication diagrams also represent *temporal* relations. In Section 2, we have seen that in sequence and communication diagrams, directions of messages among objects in a system are shown using solid arrows that are labelled by texts indicating contents of messages exchanged. In the case of sequence diagrams, the direction of messaging is shown by solid arrows drawn horizontally and pointing from the lifeline of the object sending the message towards the lifeline of the object receiving the message. Since the flow of time on a sequence diagram is represented as the increase down the vertical axis, the spatial order of those solid arrows down the vertical axis defines the temporal order of messaging in the system. In the case of communication diagrams, directions of messages among objects are also denoted by text-labelled arrows; but with the difference that they point directly from the object sending the message towards the object receiving it. However, since time is not explicitly represented on communication diagrams, solid arrows representing directions of messages are numbered in the increasing order; so that the numerical order of (message) number tags denotes the temporal order of messages. Therefore, in both sequence and communication diagrams, directions of messages, which are spatial relations, are represented as (linear) spatial directions through the use of solid arrows between objects. On sequence diagrams, the temporal order relation of messages is transformed into a spatial relation using a special convention; namely, that the increase along the vertical axis represents the flow of time. By contrast, on communication diagrams, the temporal order relation of messages is represented without having it transformed into any spatial relation; but it is represented again using a special convention; namely that the increasing numerical order of number tags of arrows represents the order of messaging. Note that, on both sequence and communication diagrams, text-labels are syntactic devices, whereas solid arrows and number-tags are symbolic devices.

Finally, I shall consider class diagrams, which represent various class relationships such as association, generalisation, aggregation and composition. I shall start with association that denotes any kind of *dependency* relationship among otherwise unrelated object classes in a system. In this sense, association relationship does not denote a spatial relation between objects. Remember that a solid line connecting the two classes under consideration represents association on class diagrams. The multiplicity of association, i.e. whether it is 'one-to-one' or 'one-to-many' or 'many-to-many', is represented using numbers and the symbol '*' (or the letter *n*). Therefore, on class diagrams, an association relationship is transformed into a spatial relationship between two object classes through the use of various symbolic devices, namely a solid line, numbers, the symbol '*' and the letter *n*. Other class relationships represented on class diagrams, namely generalisation, aggregation and composition, denote *part–whole* relationships between objects in that all objects of one class are contained in another class in the sense defined in the class-relationship under consideration. It is important to note that these part–whole relations are not *necessarily* spatial. However, as in the representation of association relationship, the foregoing relationships are transformed into spatial relations through the use of symbolic devices. Solid lines denote again all those relationships. The direction of aggregation and composition relationships is denoted by hollow and solid diamond symbols, respectively. The direction of generalisation is denoted by a solid arrow. Also, numbers are used to denote the multiplicity of the class relationship under consideration. Therefore, on class diagrams, various class relationships are transformed into spatial relations through the use of various *symbolic* devices, namely solid lines and arrows, hollow and solid diamond symbols, numbers, the symbol '*' or the letter *n*, together with their associated conventions as described above. It is optional that on these diagrams solid lines are text-labelled in order to briefly describe the class relationship under consideration.

The above discussion shows that various types of relations involved in a system are representable by diagram models through the use of what I shall call 'representational devices' that are either symbolic or syntactic. The results are summarised in Tables 2 and 3. This conclusion partly supports Shin's remark that in order for non-spatial relations to be represented on diagrams, they have to be transformed into spatial relations through the use of suitable representational devices and their associated conventions. However, two remarks are in order here. First, the only non-spatial relation that is not transformed into a spatial relation is the temporal order relation that denotes the time order of messages on communication diagrams. Second, Shin does not make a clear distinction between syntactic and symbolic devices. But, the above analysis of diagram models calls for such a distinction.

The above conclusion indicates that spatial and non-spatial relations are inferable from diagram models only to those who know the meanings of the representational devices and of their corresponding conventions used on these diagrams. If this is granted, diagrammatic representations embodied by diagram models can be said to be computationally more efficient than sentential representations in representing various relationships involved in a system. I offer the following considerations for this claim. First, since both spatial and non-spatial relations are represented as spatial relations in diagram models, they are displayed *explicitly* and thus become more readily perceivable instead of being *implicit* in sentential representations. As has been pointed out above, the representations of non-spatial relations in diagram models require the use of a set of

Table 2. Represented relations and corresponding representing relations in diagram models.

Types of diagram models	Represented relations	Representing relations
DFDs	Spatial relations: data-flows among system units	Spatial relations: text-labelled solid arrows among diagram units
Sequence diagrams	Spatial relations: directions of messages among objects	Spatial relations: text-labelled solid arrows among diagram units denoting objects
	Temporal relation: time order of messages among objects	Spatial relation: vertical spatial order of text-labelled solid arrows
Communication diagrams	Spatial relations: directions of messages among objects	Spatial relations: text-labelled solid arrows among diagram units denoting objects
	Temporal relation: time order of messages among objects	Numerical relation: numerical order of message number tags
Class diagrams	Part-whole relationships among object-classes: generalisation, aggregation and composition	Spatial relations: solid lines (with solid arrows, hollow and solid diamonds, respectively) between diagram units denoting object-classes
	Dependency relationship among object-classes: association	Spatial relations: solid lines (with arrowheads) between diagram units denoting object-classes

Table 3. Syntactic and symbolic representational devices and associated conventions used for the representation of spatial and non-spatial relations in diagram models.

Types of diagram models	Syntactic representational devices and associated conventions	Symbolic representational devices and associated conventions
DFDs	Text-labels: contents of messages exchanged	Solid arrows: directions of data-flows
Sequence diagrams	Text-labels: contents of messages exchanged	Solid arrows: directions of messages
Communication diagrams	Text-labels: contents of messages exchanged	Solid arrows: directions of messages; message number tags: numerical order of message number tags denotes time order of messages
Class diagrams	Text-labels: descriptions of messages exchanged	Solid lines: class relationships; arrowheads: directions of associations; solid arrows: directions of generalisations; hollow diamonds: directions of aggregations; solid diamonds: directions of compositions; numbers and the symbol "*" (or the letter n): multiplicities of class relationships

representational devices as presented in Table 3. The size of the set comprising these representational elements of diagram models and their conventions is far smaller than the size of the set comprising syntactic elements (such as words, articles, verbs and propositions) and their associated syntactic rules and conventions used in sentential representations. For instance, the temporal order of any two messages can be represented sententially through the use of a sentence stating which one of the messages precedes the other. On the other hand, the same temporal relation can be expressed diagrammatically on diagram models through the use of two different numbers with increasing or decreasing order. It is clear that the latter representation is more direct to perceive. Because, sentential representation in this case requires the knowledge of both syntactic elements and their associated rules and conventions, whereas numerical order requires much less; namely, only the order relation between two numbers. It is important to note that, in the above case, as the number of messages increases, it becomes more obvious to see that the complexity of sentential representation outweighs that of diagrammatic representation.

Second, by virtue of locality feature, diagram models have the ability to bring together the same types of representing relations more easily (Larkin and Simon 1987, p. 98). For example, as shown in Table 2, while relations representing data flows lie adjacent to each other on DFDs, relations representing messages among objects lie adjacent to each other on sequence diagrams. It is to be noted that all types of relations that can be represented by diagram models can also be represented sententially. However, as the number of represented relations increases, the number of sentences representing those relations also increases. This results in a gradual separation of representing relations from each other in sentential representation and thus of related pieces of information that must be used together for design and analysis purposes. Therefore, it requires more effort and complex reasoning to access and use the same information from sentential representations than from diagrammatic representations.

Lastly, remember that according to my analysis in the previous section, the spatial and non-spatial relations associated with a system denote the dynamic and static features of this system. Thus, the discussion in this section suggests the conclusion that diagrammatic representation proves more advantageous to represent the dynamic and static features of a system.

6. Summary and concluding remarks

In this article, I have offered a philosophical perspective on the modelling features of diagram models as well as on their advantages over sentential representations. In philosophy of science, the term 'model' is *typically* used in relation to theoretical models. I have applied M&M's 'models as mediators' account to diagram models and showed that even though they differ from theoretical models in terms of type of representation they embody, their essential features can be analysed in terms of the same categories, namely construction, function, representation and learning. I have argued that diagram models play a mediating role between an abstract system and its concrete realisation in the real world through the diagrammatic representation of the dynamic and static features of the system under consideration. I have also pointed out that by virtue of this mediating role, diagram models can function as instruments of both analysis and design. This mediating role can be likened to the mediating role played by theoretical models, which, in M&M's account, mediate between the high-level theory and the real world.

Moreover, in this article, drawing on the literature in cognitive science and philosophy, I have argued that in terms of the ease of both acquiring and conveying information, diagrammatic representations delivered by diagram models are more efficient than sentential representations in representing the dynamic and static features of information systems. Of course, this conclusion should be understood with the *proviso* that those who are to utilise diagram models have the necessary expertise on the rules and conventions as well as on syntactic and symbolic devices employed in the construction and interpretation of diagram models.

Notes

1. This approach finds its roots in the writings of logical empiricist movement and also of Karl Popper. It is also prevalent in the writings of the philosophers of the post-positivist school, e.g. Quine, Hanson, Kuhn and Feyerabend.
2. See, e.g. the articles in the edited volumes by Lynch and Woolgar 1990 and Baigrie 1996.
3. However, there are notable exceptions to this general observation. For instance, Giere (1996) examined the use of diagrams that played a central role in the twentieth century revolution in geology and called them 'visual models', in the sense that they are *non-sentential* and provide *visual representation* of world phenomena. An earlier exception is Harré (1970, chap. 2), who used the term 'model' to connote *also* pictures. What he calls 'iconic models' are those that substantially resemble the subjects they represent. For example, the 1953 Watson-Crick model of DNA can be seen as an iconic model in Harré's sense.
4. For the presentation of SSAD, I shall rely on Hoffer et al. 2008, which is a prominent source in SSE.
5. This example is from Hoffer et al. 2008, chap. 7.
6. For various other data flow diagramming rules, see Hoffer et al. 2008, pp. 214–220.
7. For the presentation of OOAD, in addition to Hoffer et al. 2008, I shall also rely on Booch et al. 2007, which is also a prominent source in SSE.
8. For different types of messages, see Booch et al. 2007, sec. 5.8.
9. For a discussion of scientific models, see, e.g. Frigg and Hartmann 2006.
10. For a review on this subject, see, e.g. Cheng, Lowe, and Scaife 2001.
11. Sloman uses the term 'Fregean' representations for the second category.

Acknowledgements

This research has been carried out in the context of the project 'LHC-experiments between theory-ladenness and exploration: the problem of data selection', funded by the German Science Foundation (DFG), within the research collaboration 'The Epistemology of the Large Hadron Collider (LHC)' at the University of Wuppertal: http://www.lhc-epistemologie.uni-wuppertal.de. The present article has benefitted from discussions at the Epistemology of Modeling & Simulation conference, Pittsburgh, PA, 1–3 April 2011. Special thanks are due to Pearson Education, Inc., and Grady Booch and co-authors for granting permission to use material from their publications.

References

Aster, R., and Shishko, R. (1995), *NASA Systems Engineering Handbook*, Washington, DC, USA: National Aeronautics and Space Administration.

Baigrie, B.S., (ed.) (1996), *Picturing Knowledge: Historical and Philosophical Problems Concerning the Use of Art in Science*, Toronto: University of Toronto Press.

Booch, G., Maksimchuk, R.A., Engle, M.W., Young, B.J., Conallen, J., and Houston, K.A. (2007), *Object-Oriented Analysis and Design with Applications*, Upper Saddle River, NJ: Addison-Wesley.

Cheng, P.H., Lowe, R.K., and Scaife, M. (2001), 'Cognitive Science Approaches to Understanding Diagrammatic Representations', *Artificial Intelligence Review*, 15, 79–94.

DeMarco, T. (1978), *Structured Analysis and System Specification*, Upper Saddle River, NJ: Prentice-Hall.

Frigg, R., and Hartmann, S. (2006), 'Models in Science', in *The Stanford Encyclopedia of Philosophy* (Spring 2006 ed.), ed. E.N. Zalta, http://plato.stanford.edu/entries/models-science/

Gane, C., and Sarson, T. (1979), *Structured Systems Analysis: Tools and Techniques*, Englewood Cliffs, NJ: Prentice Hall.

Giere, R. (1996), 'Visual Models and Scientific Judgment', in *Picturing Knowledge: Historical and Philosophical Problems Concerning the Use of Art in Science*, ed. B.S. Baigrie, Toronto: University of Toronto Press, pp. 269–302.

Harré, R. (1970), *The Principles of Scientific Thinking*, Chicago: Chicago University Press.

Hoffer, J.A., George, J.F., and Valacich, J.S. (2008), *Modern Systems Analysis and Design* (5th ed.), Upper Saddle River, NJ: Pearson Education, Inc.

Koedinger, K.R. (1992), 'Emergent Properties and Structural Constraints: Advantages of Diagrammatic Representations for Reasoning and Learning', in *AAAI Technical Report on Reasoning with Diagrammatic Representations (SS-92-02)*, ed. N.H. Narayanan, Menlo Park, CA: AAAI, pp. 151–156.

Koedinger, K.R., and Anderson, J.R. (1990), 'Abstract Planning and Perceptual Chunks: Elements of Expertise in Geometry', *Cognitive Science*, 14, 511–550.

Kulpa, Z. (1994), 'Diagrammatic Representation and Reasoning', *Machine Graphics and Vision*, 3, 77–103.

Larkin, J.H., and Simon, H.A. (1987), 'Why a Diagram is (Sometimes) Worth Ten Thousand Words', *Cognitive Science*, 11, 65–100.

Lynch, M., and Woolgar, S., (eds.) (1990), *Representation in Scientific Practice*, Cambridge, MA: MIT Press.

Morgan, M., and Morrison, M. (1999), 'Models as Mediating Instruments', in *Models as Mediators: Perspectives on Natural and Social Science*, eds. M.S. Morgan and M. Morrison, Cambridge: Cambridge University Press, pp. 10–37.

Shin, S. (1994), *The Logical Status of Diagrams*, Cambridge: Cambridge University Press.

Sloman, A. (1971), 'Interactions between Philosophy and Artificial Intelligence: The Role of Intuition and Non-Logical Reasoning in Intelligence', *Artificial Intelligence*, 2, 209–225.

Sloman, A. (1975), 'Afterthoughts on Analogical Representation', in *Proceedings of Theoretical Issues in Natural Language Processing* (TINLAP-1), eds. R. Schank and B. Nash-Webber, Cambridge, MA: MIT Press, pp. 164–168; *Readings in Knowledge Representation*, eds. R.J. Brachman and H.J. Levesque, Los Altos, CA: Morgan Kaufmann Publishers, pp. 431–440 (Reprinted in 1985).

Yourdon, E., and Constantine, L.L. (1979), *Structured Design: Fundamentals of a Discipline of Computer Program and Systems Design*, Englewood Cliffs, NJ: Prentice-Hall.