AN ONTOLOGY OF IS ARTEFACTS – SOME QUESTIONS

Hans Weigand[1], Paul Johannesson[2], Birger Andersson[2]

[1] Tilburg University, P.O.Box 90153,
5000 LE Tilburg, The Netherlands
{H.Weigand}@uvt.nl
[2] Stockholm University
Department of Computer and Systems Sciences, Sweden
{pajo,ba}@dsv.su.se

Abstract

From a design science perspective, information systems and their components are viewed as artefacts. Building on our ontology of artefacts (in progress), we want to address the question: what are the major kinds of artefacts in the IS field and how are they ontologically related? In this short paper we want to raise two questions: (1) how can an artefact ontology support business ontology? (2) how can an artefact ontology analyze the concepts of algorithm and some other objects in the IT domain?

Keywords: design science, artefact, algorithm

1 Introduction

Design Science Research [Hevner, 2004] is growing in popularity in the Information Systems field. Whereas empirical sciences acquire knowledge about the natural world (physics) or human behavior (social sciences), Design Science Research is interested in IT artefacts, such as algorithms and modeling languages, and the effectiveness of their use. As such, it has similarities with traditional engineering sciences and medicine, although the physical aspect of DSR artefacts in Information Systems is minimal. Building artefacts and evaluating artefacts in context is often seen as the core of DSR. However, there has been quite some unclarity about what is an artefact. Is artefact the same as "artificial object" [Simon,1996]? Although by now a lot has been written about design science, in the form of research articles and textbooks, a systematic investigation of the design artefact concept is still missing. Building on our ontology of artefacts (in progress), in this short paper we want to raise two questions: (1) how can an artefact ontology support business ontology? (2) how can an artefact ontology analyze the concepts of algorithm and some other objects in the IT domain?

2 Ontology of artefacts

Our ontology of artefacts (in progress) builds on and extends the work of Borgo [2009;2014] as well as Lawson [2008] and Kroes [2006] whose work is not addressing design science but the traditional technical sciences. We list here the summary of definitions only

An **artificial object** is a physical object intentionally or unintentionally created (or adapted) by a human. A **natural** object is a physical object that is not an artificial object.

An **instrument** is a role of an object in an event or activity. The instrument is *used* in the event.

A **technical object** is a physical object (*artificial* or *natural*) individual that conforms to an *artefact*. The *artefact* has a *make plan*, which allows reproducing more technical objects conforming to the artefact, as well as a *use plan* that specifies in which events and under which use conditions the technical object can be used successfully (so technical objects are in the domain of the role *instrument*). A technical object has *attributed capacities* that are *realized* in the use.

An **artefact** like ”the Diesel engine” is a type (1stOT in MLT - with ”artefact” a 2ndOT category) that has a design specification consisting of a *make plan*, a *use plan* and a *capacity specification*. A distinction is made between **technical artefacts** when there are *technical objects* conforming to the artefact, and **non-technical artefacts**, when this is not the case. A **product** is a type with a make plan but not necessarily a use plan and instrument role. So artefacts are products, but not vice versa.

> Examples of artificial objects are not only hammers, cars and software, but also a footprint in the sand. However, the latter is not a technical object, as it is not intentionally made, and has no use plan. A pebble at the beach is a natural object. A pebble taken by a human to serve as doorstop is an artificial object (formed from the natural object pebble) when the stone is adapted (moved, cleaned) for a purpose (intention). It also becomes an instrument then. When in some culture, pebbles are recognized as decent doorstops, then ”pebble” is a variant of the doorstop artefact and a pebble individual adapted and used in this way can conform to this artefact. A pebble has physical capacities like a certain weight and friction on which the attributed capacities – it is movable by a human, it does not move when a door hits it without extrta human pressure. For a computer, the attributed capacity is that it can execute a binary. This attributed capacity also builds on the physical capacities, but in a complicated way.
>
> A bird nest is not an artificial object as there is no human creator. A painting by Van Gogh is an artificial object, but not a technical object, nor a product. Cake is a product and artefact.
>
> An example of an instrument that is not a technical object is a branch that a walker picks up to be used as stick. Seedless banana individuals [Sperber 2007] are not technical objects, although they are the result of a long breeding process. However, the seedless banana type is a (non-technical) artefact to which seedless banana individuals conform. The latter are also instances of the kind banana. A synthetic diamond individual is a technical object that conforms to the technical artefact “synthetic diamond”. It is also an instance of the kind diamond, just like natural diamonds.

We posit a ”form” event behind any intentionally created artificial object. The form event has a physicial input and output, and an intentional input and output. The former are responsible for the material aspect of the object, the latter for the form aspect. For the technical object in particular, the intentional input corresponds to (is captured by) the make plan (as part of the artefact design), whereas the intentional output corresponds to the attributed capacity.

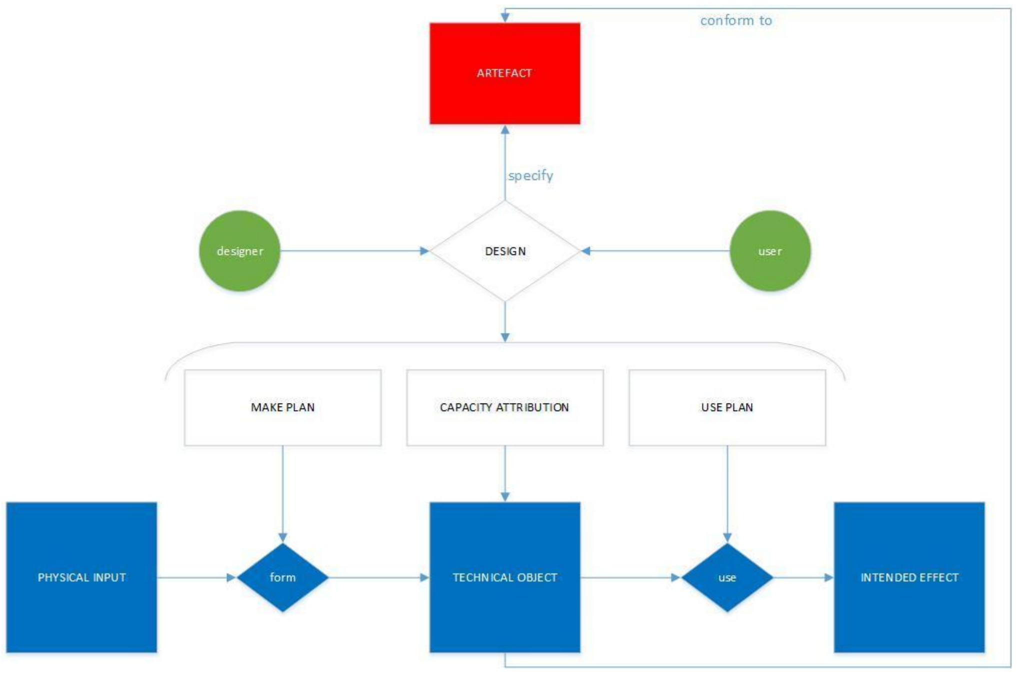

Fig. 1 Artefact ontology (informal overview)

A **text object** such as the blue book on my desk is a *technical object* where the attributed capacity (the written content) can be recaptured in the use (the reading). The effect of using the text object (when the use plan is human reading) is a change in the mental state of the human agent corresponding to the attributed capacity (the text meaning). A text object has a physical basis (it is a technical object), a meaning (the attributed capacity) and an intended use. A text artefact is an artefact that has text objects conforming to it. In principle, each (content-wise) different text is a different artefact, but they are all subtypes of the kind ”text”, or more specific texts such as ”program”. A text artefact is differentiated from other texts by its specific design specification.

Although a human language is not an artefact, text is an artefact. More in general, we can talk about writing technology that started with clay tablets and papyrus and nowadays is evolving into digital technology. Basically, writing is a form event by which a technical (physical) object is created (the individual piece of paper, blank before the writing, and filled with characters after the writing). What counts for the writing is the meaning that is attached to the characters - the intentional output of the writing event. This meaning is recaptured by the reader in the reading process. We can use the distinction between data and information (datalogical vs infological) to distinguish the technical object (data object) and the attributed content (info object). The reading recaptures the meaning (so the reader attributes the same capacity to the text as the author) drawing on the shared language and shared education in reading/writing (use condition). According to [Sinha 2015], it is not specific to texts that the attributed capacity is understood by the user – it applies to tools in general – but of course this ”meaning” is much more detailed and specific for a text than it is for, say, a bicycle.

Examples of text objects are Mary’s book, and Paul’s computer chess game. Text objects may also be called symbolic objects, as long as it is kept in mind that these objects always have a physical representation.

A (domain) **Design Science (DSR) artefact** is an artefact with a well-described design specification (formal rigor) that is not only directly or indirectly used in a (domain - e.g.information system) practice but also in the research practice (the scientific community) for building up knowledge. Note that not every artefact is a DSR artefact. In the same way, not all design is design research (cf. [McPhee, 1997]).

When a certain rigor is required for the design, the design specification must be explicit in the form of a text object. This turns the artefact (as having a 1-1 relationship to its design specification) into a text object itself. Often, this is a composite object, consisting of several text objects. For instance, in the case of system design, one text object could be a data model. This data model (text object) itself instantiates a data model artefact. At the data model artefact level, there is not only an attributed capacity (the data model content or meaning) but also a make plan (including e.g. layout, and the use of a modeling language) and use plan. So at the data model artefact level, the focus (and analysis) is not on the database to be built, but on the data model. We posit that text artefacts in the IS domain include, for instance, algorithms, programming languages, the neural network and the Tropos requirements engineering method. What exactly is the status of these objects and how they relate to IT needs to be clarified further.

3. Artefact ontology as part of business ontology

We suggest that an ontology of artefacts can contribute to business ontologies. Economic entities are characterized by exchange and production processes.The REA ontology talks about resources that can be understood as rights on some underlying object (POA, COFRIS). In most if not all cases, the underlying object is an artefact, and so has artefact properties such as an intended use and an artefact life cycle. These properties are important, for instance, for understanding the post-delivery exchange phase and environmental accounting. This holds at least for material objects. Given the artefact properties, two kinds of after sales relationships between provider and customer can be identified: (a) the provider keeps a certain level of responsibility for the object after the exchange, including the responsibility to support the customer in the proper use; (b) for continuous evolution of the artefact design, use experience needs to be collected and evaluated.

For immaterial exchange objects (in particular, services) the artefact status is debatable: they are designed, but service instantiations are typically activities, not physical objects. One way of looking at services is to see them as the interface to valuable enterprise resources (e.g. a taxi drive is an interface to the the taxi and the driver). Then the service design is part of the design specification of these underlying resource objects.

4. Algorithms and some other IS artefacts

March & Smith [1996] mention constructs, models, methods and instantiations as IS design reserach artefacts, where ”methods” includes algorithms. According to [Offermann et al, 2010] an algorithm, like a method, describes a sequence of activities that are executed by a computer. They define, it is as “an executable description of a system behavior. Examples given are “sorting algorithm”, “data mining algorithm”, and “protocol”. From the ontological perspective, the sorting algorithm, or one of its variants, e.g. quicksort, is an artefact that is specified by pseudo-code and is instantiated in all its executable representations (text objects conforming to the algorithm artefact). In the pre-computer age, these executable representations had to be executed (mentally or on paper) by humans. The executable representation is used in a sorting process, taking a list of numbers (elements of an ordered set) in order to generate a sorted list. This sorted list is created and is itself an instance of the "sorted list" type, another artefact. The sorted list artefact specifies an attributed capacity (that the numbers in the list are ordered) and a use plan, e.g. the use in a search process. The use plan of the algorithm coincides with the make plan of the sorted list (that make plan and use plan of two artefacts coincide is quite common in production processes).

In the computer age, a software program implements the algorithm. Strictly speaking, the computer does not execute the algorithm, but the software code, where a distinction must be made between source code and object code. In the case of compilation, the program (JAVA source code) is itself also an artefact (type level). The program is described by JAVA code and is instantiated in all source code representations, e.g. John's copy. When the source file is compiled, John's copy is a text technical input object (it is somewhere in the memory, has a size in number of bytes, etc.) but in the compilation process, it is read and used as intentional input. The meaning of the text (source code) is translated into the capacities of the object code. Subsequently, the object code can be the make plan for producing some desired data object, as described above, but now by using the computer rather than in a manual way.

Although an algorithm specification can be instantiated in a software program, the reverse is not necessarily true. For a program to be regarded as an algorithm instance, there must be a well-defined use goal - like the property "being sorted". As described in the above, the use plan of the algorithm is itself a make plan. This is not true for every program - for instance, for a software game. The game is an artefact, but it is not the instance of an algorithm. The game program is the instance of a game design, and this game design is an artefact. In the classification of [Offermann et al, 2010], this artefact is a "system specification". The distance between the game design and the game software can be big or small. If the specification is already described by source code, or pseudo-code, the distance is small and the interesting artefact (from a DSR perspective) is the game design. If the game design only specifies some constraint (e.g. "a program that plays chess"), then the distance with the code is big, and the program itself is an interesting DSR artefact.

Where algorithms are type-level artefacts on the behavioral dimension of systems, other "system specification" dimensions include the data structure dimension, the software structure dimension and the interaction structure dimension. Well-known examples include the relational data model, B-trees (data structure), the service-oriented architecture (software structure) or the TCP/IP protocol (interaction structure). Supporting all system specification dimensions are formal language artefacts, like computer languages. Offermann [2010] also mention the "pattern" as artefact. The pattern is interesting as it can be seen as design knowledge (technological rule) about some artefact (and so not the design science artefact itself), or can be seen as a component design object, just like a generic software library, and then it is indeed (on the type level) a genuine DSR object. A method is also mentioned as a design artefact by Offermann (and found in many papers). The method is not executable by a machine. If it is well-described, then it takes the form of a text object, and so as a technical object. To count as a DSR object, it must be formalized (well-described at type level, in terms of its composition, specification and use plan) and of course used in a practical context. The "use" may be wider than what Offermann indicates ("activities to create or interact with a system") as long as it falls within the scope of IS research.

4. Conclusion

Questions to the VMBO community:

(a) How relevant is the artefact status of resources (products) for business ontology?
(b) Is an algorithm an artefact, and what is the relationship algorithm/program?

References

Borgo, S, Vieu, L. (2009) Artefacts in formal ontology. Philosophy of technology and engineering sciences, 273-307

Borgo, S., M. Franssen, P. Garbacz, Y. Kitamura, R. Mizoguchi and P.E. Vermaas (2014) Technical Artifacts: An Integrated Perspective, Applied Ontology 9, 217-235

Hevner, A., March, S., Park, J. and Ram, S. (2004) Design Science in Information Systems Research'. *MIS Quarterly*, Vol 28, No 1, pp.75-105.

Kroes, P. A. and Meijers, A. W. M. (2006). The dual nature of technical artifacts, Studies in History and Philosophy of Science, 37(1): 1–4

Lawson, C. (2008) An Ontology of Technology: Artefacts, Relations and Functions. Techne 12(1):48-64.

March, ST, Smith GF (1995) Design and natural science research on information technology. Decision Support Systems 15:251-266.

McPhee, K. (1997) Design Theory and Software Design. Technical report TR 96-26, Univ of Alberta.

Offermann, P. Blom, S., Schönherr, M, Bub U (2010) Artefact types in information system design science – a literature review. In: Winter R et al (eds) Global perspectives on design science research, Springer, pp.77-92.

Simon, H. (1996) The Sciences of the Artificial (3rd edition), MIT Press.

Sinha, Ch. (2015). Language and other artifacts. Frontiers in Psychology.

Sperber, D. (2007), "Seedless Grapes: Nature and Culture", in Margolis, Eric and Stephen Laurence (eds.), 2007, Creations of the Mind: Theories of Artifacts and Their Representation, Oxford: Oxford University Press.pp. 124–137