Software as an Information Artifact

Conference Paper · September 2014

DOI: 10.13140/2.1.3844.8640

4 authors, including:

Xiaowei Wang
Università degli Studi di Trento
4 PUBLICATIONS **15** CITATIONS

SEE PROFILE

Giancarlo Guizzardi
Universidade Federal do Espírito Santo
234 PUBLICATIONS **4,090** CITATIONS

SEE PROFILE

John Mylopoulos
University of Toronto
744 PUBLICATIONS **23,329** CITATIONS

SEE PROFILE

Software as an Information Artifact

Xiaowei Wang[1], Nicola Guarino[2], Giancarlo Guizzardi[3], John Mylopoulos[1]

[1] Department of Information Engineering and Computer Science, University of Trento, Italy
{xwang, jm}@disi.unitn.it
[2] ISTC-CNR, Trento, Italy
guarino@loa.istc.cnr.it
[3] Ontology and Conceptual Modeling Research Group (NEMO),
Federal University of Espírito Santo (UFES), Brazil
gguizzardi@inf.ufes.br

Abstract. Although software plays an essential role in the modern society, few studies have been done on the true nature of software. For many, software is just code, something intangible best defined in contrast with hardware, but it is not particularly illuminating. Through the practical scenarios, it is common to recognize a software as the same one underlying many changes in the code, such as bug fixing. This process reveals that software is something different than just a piece of code, and we justify its intrinsic nature as an information artifact. Building on Jackson's and Zave's seminal work on foundations of requirements engineering, we propose in this paper several kinds of software artifacts, and the huge impacts to the human society of these software artifacts make them different from other kinds of information artifacts.

Keywords: software, information, artifact, information artifact, software requirements.

1 Introduction

The term 'information' is usually used to indicate something contains knowledge. People deal with huge amount of information every day. We read news to know what are happening even on the opposite side of the earth; we check the flight and train time tables to arrange our trips; we watch movie or read novels in our leisure time for fun. Our life depends on the knowledge which is encapsulated in the information. As in most of the cases we create the information intentionally, we can recognize it as artifact. Intuitively, we can name it as information artifact (IA), which is constituted by information. Further discussions on artifact, constitution relation could be found in the rest of the paper, as well as the information.

Usually, software is understood as a set of instructions which is intangible, as the opposite of hardware, used to control computers[1].As this set of instructions contains

[1] Although software might be interpreted as a more general concept, we restrict our discussions to the computer software in this paper.

the processing knowledge of the computer, and usually developed deliberately, it is reasonable to recognize software as an IA. Similar as other kinds of IAs, software plays an essential role in the human society. It has become an indispensable element of our culture, as it continues its relentless invasion of every facet of personal, business and social life.

Despite the importance of information and software, after decades of study on the topics, it seems there is still ambiguity about the true nature of information, software and the relation between them. To tackle this problem, we try to discuss the ontological nature of information artifact in this paper, and especially provide an ontology of software, which might become a foundation for relating research in the software engineering (SE) and requirements engineering (RE), such as software evolution, software configuration management, and etc.

To illustrate out idea, the rest of the paper is arranged as follows: section 2, the concepts of information, artifact, and information artifact are discussed; section 3, details of recognizing software as an information artifact is illustrated; section 4, an ontology of software artifacts is proposed (only a concise introduction is provided, as the details of the ontology is published in a paralleling FOIS 2014 paper); section 5, a brief conclusion is presented.

2 What is An Information Artifact

Since the famous semiotic triangle was proposed by Ogden et al. [1], it has been more than 10 years researchers trying to figure out the true nature of information. Smith [2] and Ferrario et al. [3] tried to explain what are 'mind' and 'concept' respectively. Following them, Maass [4], Gangemi [5] and Jureta et al. [6] provided several proposals about 'information object'. Among the researchers, Fortier and Kassel (F&K hereafter) proposed an ontology of Information and Discourse Acts (I&DA) [7], in which a systematic ontological analysis was provided about the information and relating concepts. In their paper, three main concepts were proposed: Information (called 'Content' originally), a non-physical object, the mind or knowledge itself; Expression (called 'ContentBearingObjects' originally), a syntactic representation of the information, which is also a non-physical object; Inscription, the realization of the syntactic representation which is the physical support of the information.

F&K's proposal is clear and intuitive enough to explain the information related concepts, and we take it as a baseline with some terminology modification. An example could make it clearer: I have an idea that I love my dog, and this thinking content/knowledge is the *information* in my mind; then I want to express this idea to someone else, then I figure out an sentence as 'I love my dog'. I haven't written it down anywhere, this syntactic sequence of symbols according to English is called *expression*; finally, I write down the sentence on a piece of paper, the structure of the ink on the paper forms the *inscription* of the *information*. Note the inscription depends on the physical *medium* but it is not the *medium*.

Till now, we explained the related works about information, but information does not equal to information artifact, we still need to figure out the artifact part of the

information artifact. Among several literature works about artifact [8], [9], [10], [11], we find the one proposed by Baker [9] works very well for us: firstly, Baker makes clear that artifacts are the results of intentional processes, which, in turn, are motivated by intentions (mental states) of (agentive) creators; moreover, she connects the identity of an artifact to its proper function, i.e., what the artifact is intended to perform; finally, Baker recognizes that the relation between an artifact and its proper function exists even if the artifact does not perform its proper function.

A simple scenario could be a paperweight is an artifact constituted by a pebble from a river [11]. When I walk along a river, I find a pebble which is very nice. I pick it up and decide to use it to make my papers stable on the table. Just at the same time the decision is made, a paperweight is merged as a new artifact collocating with the original natural pebble.

Recently, by considering both concepts of information and artifact, Smith et al. [12] provided a definition of IA based on the Information Artifact Ontology (IAO)[2] which was proposed for the Human Genome Project, "*an information artifact is an entity that has been created through some deliberate act or acts by one or more human beings, and which endures through time, potentially in multiple copies.*" For the case of simplification, we can translate this definition with our terminologies as "*an information artifact is an artifact which is constituted by an inscription of the information.*"

Smith's proposal is suitable to deal with the information which is inscribed with physical support, such as my passport, and this passport depends on the paper and ink but does not equal to them. Like Dublin Core [13], they try to provide terminologies to describe such kind of artifacts, such as "*IA #12345 is-about some given person, or uses- symbols-from some specified symbology, or links-to some second IA #56789* [12]." Although the '*information artifact*' proposed by Smith is essential is many cases, we treat it from a different perspective that an *information artifact* is an artifact which is constituted by a syntactic expression of the information.

We believe it is also important to independently consider the syntactic expression of information as constituent of IA. Furthermore, as stated in [14] by Kassel, we treat information artifact (e.g., especially software) as non-physical artifact instead of physical artifact. Similar view was hold by Faulkner and et al. [15] that software is non-material object and which is "*a 'syntactic' entity ... in consisting of a set of well-formed expressions* [16] *written in an appropriate language, and where 'well-formed' means that these expressions adhere to the syntactical and semantic rules of that language.*" The statements above lead us to our proposal that software is an information artifact, and this argument is explained in detail in the following section.

3 Software as an Information Artifact

Focusing on information aspects of software, several scholars (e.g. Eden and Turner [17], Oberle [18]) have addressed the complex relationships among i) software *code*,

[2] https://code.google.com/p/information-artifact-ontology/

consisting of an well-formed expression of a set of computer instructions; ii) a software *copy*, which is the inscription of this set of instructions through a hard medium; iii) a *medium*, the hardware medium itself; iv) a *process*, which is the result of executing the software copy.

A different approach to account for the artefactual nature of software is taken by Irmak [19], According to him, software is synonymous to program and can be understood in terms of the concepts of algorithm, code, copy and process, but none of these notions can be identified with software, mainly because, due to its artefactual nature, software has different identity criteria. We share many of Irmak's intuitions, as well as the methodology he adopts to motivate his conclusions, based on an analysis of the condition under which software maintains its identity despite change.

Consider another simple scenario, I want to develop a simple alarm system to notify me the time. Then, I write a piece of code *c1*, and I assume it will run well in the computer. However, after a test, it doesn't work and I find a bug, and by fixing the bug I have a different piece of code *c2*, and this time it works well. It is a common sense that, during the process, there is something (the program) keeps unchanging with different codes at different times, and it is the ownership of the bug leads to such statement. Simply the code is loyally performed by the computer, and the computer will accept every code with no different judgment; while, on the other side, it is the people who judge the artifact (the program) created according to their intentions. In other words, it is the program buggy, but not is the code. Hence, programs are necessarily artifacts and possess essential proper functions, but codes are not necessarily artifacts, and the relation between an artifact (the program) and its material substrata (the code) is taken to be one of *constitution*.

A program has the essential property of being intended to play an internal function inside a computer. For every program we assume the existence of a unique *specification* of such expected behavior, called *program specification*. In order for a program to exist, this specification must exist, at time *t* there is somebody who recognizes *a piece of code c* as intended to implement the program specification *s*, or there is an explicit description of this intention (e.g., via a documentation in the code or in an explicitly described program specification) which is recognizable by someone.

Till now we can conclude that a program is an information artifact, and a program specification along with the intentional creation act could be used as the identity criteria of it. Besides program, there are other kinds of software information artifacts, which could be systematically categorized into different abstract layers, and such discussion is provided in the following section.

4 Ontology of Software Artifacts

As we have seen, the identity criteria of programs are bound to the *internal* behavior of a computer. On the other hand, software is usually intended as an artifact whose ultimate purpose is constraining the behavior of an environment *external* to the computer. In the general case, the software's ultimate purpose is achieved by running a

code which produces certain effects inside a computer, which drives a physical machine, which in turn produces certain effects on its external environment.

This view has been described by Jackson et al. [20] which draw a clear distinction between the external environment, which is where the ultimate effects of software are expected, and the internal computer-driven system.

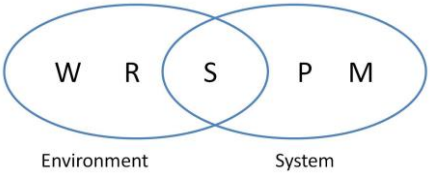

Figure 1. A model of software requirements engineering (from [20]).

This view is depicted in Figure 1. As [20] observes, W-R-S-P-M are descriptions/restrictions (we call them specifications to unify the terminologies) on the phenomena happening in different parts of the environment and system: a program specification is represented as P, platform specification/machine knowledge as M, interface specification as S, requirement specification as R and environment specification/domain knowledge as W. An interface specification is seen as a communication means between the environment (where both W and R hold) and the system (where M and P exist). As described by the authors: *"if S properly takes W into account in saying what is needed to obtain R, and P is an implementation of S for M, then P implements R as desired"*

In conclusion, while in the previous section we focused on the *immediate* function of programs as information artefacts, producing some effects inside a computer. Here this model allows us to understand how programs play their *ultimate* function, which is producing certain effects in the external environment. According to the specifications achieved in different parts of the environment of system, we can propose three main kinds of information artifacts: a *program* is constituted by some *code* intended to implement P under M; a *software application* is constituted by a *program* intended to implement S; a *software product* is constituted by a *software application* to implement R under W.

5 Conclusions

In this paper, we discussed the ontological nature of information artifact, and how software could be recognized as an information artifact. A brief introduction of an ontology of software is also presented, and a detail version could be found in a paralleling paper in FOIS 2014 titled '*Towards an Ontology of Computer Software: a Requirements Engineering Perspective*'. We wish this work could become foundations for the relating domains such software evolution, versioning control and etc.

Acknowledgements. Support for this work was provided by the ERC advanced grant 267856 for the project entitled "Lucretius: Foundations for Software Evolution" (http://www.lucretius.eu), as well as the "Science Without Borders" project on "Ontological Foundations of Service Systems" funded by the Brazilian government.

References

1. Ogden, C.K., Richards, I.A., Malinowski, B., Constable, J., Crookshank, F.G.: The Meaning of Meaning: A Study of the Influence of Language Upon Thought and of the Science of Symbolism. Routledge (2001).
2. Smith, B.: Beyond concepts: Ontology as reality representation. In: Varzi, A. and Vieu, L. (eds.) Proceedings of FOIS 2004. pp. 73–84. IOS Press, Turin (2004).
3. Ferrario, R., Oltramari, A.: Towards a Computational Ontology of Mind. Aerospace Conference, 2005 IEEE. pp. 1–9 (2005).
4. Maass, W., Goyal, S., Behrendt, W.: Knowledge Content Objects and a Knowledge Content Carrier Infrastructure for ambient knowledge and media aware content systems. Knowledge-based Media Anal. 449–456 (2004).
5. Gangemi, A., Borgo, S., Catenacci, C., Lehmann, J.: Task Taxonomies for Knowledge Content D07. (2004).
6. Jureta, I.J., Mylopoulos, J., Faulkner, S.: A core ontology for requirements. Appl. Ontol. 4, 169–244 (2009).
7. Fortier, J.-Y., Kassel, G.: Managing knowledge at the information level: an ontological approach. Proceedings of the ECAI'2004 Workshop on Knowledge Management and Organizational Memories. pp. 39–45 (2004).
8. HILPINEN, R.: On artifacts and works of art1. Theoria. 58, 58–82 (1992).
9. Baker, L.R.: The ontology of artifacts. Philos. Explor. 7, 99–111 (2004).
10. Borgo, S., Vieu, L.: Artefacts in formal ontology. Handb. Philos. Technol. Eng. Sci. 273–308 (2009).
11. Guarino, N.: Artefactual Systems , Missing Components , and Parts Replacement. Artefact Kinds: Ontology and the Human-Made World. pp. 1101–1117 (2013).
12. Smith, B., Malyuta, T., Rudnicki, R., Mandrick, W., Salmen, D., Morosoff, P., Duff, D.K., Schoening, J., Parent, K.: IAO-Intel: An Ontology of Information Artifacts in the Intelligence Domain. In: Laskey, K.B., Emmons, I., and da Costa, P.C.G. (eds.) STIDS. pp. 33–40. CEUR-WS.org (2013).
13. Dublin Core User Guide, http://wiki.dublincore.org/index.php/User_Guide.
14. Kassel, G.: A formal ontology of artefacts. Appl. Ontol. 5, 223–246 (2010).
15. Faulkner, P., Runde, J.: The social, the material, and the ontology of non-material technological objects. European Group for Organizational Studies (EGOS) Colloquium, Gothenburg. (2011).
16. Lando, P., Lapujade, A., Kassel, G., Fürst, F.: An Ontological Investigation in the Field of Computer Programs. In: Filipe, J., Shishkov, B., Helfert, M., and Maciaszek, L. (eds.) Software and Data Technologies SE - 28. pp. 371–383. Springer Berlin Heidelberg (2009).
17. Eden, A.H., Turner, R.: Problems in the ontology of computer programs. Appl. Ontol. 2, 13–36 (2007).
18. Oberle, D.: Semantic Management of Middleware. Springer (2006).
19. Irmak, N.: Software is an Abstract Artifact. Grazer Philos. Stud. 86, 55–72 (2013).
20. Gunter, C.A., Jackson, M., Zave, P.: A reference model for requirements and specifications. Software, IEEE. 17, 37–43 (2000).