An Issue About the Revision of Digital Artifacts

Duygu Aktas

Social Ontology 2020

Topic: Digital objects such as computer programs, web-based programs, social media accounts, etc. appear to change enormously. One position is that such changes are only result of some linguistic conventions rather than true changes. Our daily behaviors about digital objects suggest otherwise. We use several software artifacts which undergo changes through time, and we still believe that we use the same product. In this presentation, I argue that digital objects survive through changes.

1 What is a digital artifact?

1.1 Digital objects are eternal abstract objects.

According this view, digital objects are identical either with their algorithm or their code.

An algorithm

…is an eternal object similar to mathematical entities such as numbers and sets.

…implements a certain function that is intended to create desired change for effective data usage in a computer.

…describes which steps will be taken and how these steps will be sequenced in order to reach a desired end.

Code: A code is a written version of these instructions step by step in a computer language.

1.2 Rigid dependence:

Digital artifacts are not identical with neither their algorithm nor their code instead they are rigidly dependent on their algorithm. According to this view, software is an historical kind of artifact which is produced by its programmers with certain intentions (Irmak, 2012, p.68).

The problem is that either position does not explain the change in digital artifacts. In both cases, whenever there is a new algorithm, *necessarily* there is a new digital object.

1.3 A layered ontology of digital artifacts:

During the programming process, a programmer produces various different artifacts such as a code base, a program, a software system, etc. Some of these belong to different kinds, and some constitute others (Wang et al., 2014).

1.3.1 Problems with the layered ontology of digital artifacts:

a. The most fundamental problem is that how the constitution relation holds between these different kind of entities is not clear from their arguments. They adopt Baker's notion of constitution. Baker's notion of constitution requires spatial coincidence, and software artifacts do not have spatial properties.

b. Even if we were to guarantee that constitution relation holds in their account as they argued for, their view still gives rise to an odd metaphysical conclusion (I discussed this part on my thesis, and skipped on this video).

c. There are various Facebook, Instagram, Microsoft Words products, etc.

In this account, the software product is the closest object to "software" in our daily usage. The essential property of a software product is its high-level requirements which means that intentions of a producer carried by the high-level requirements of the product. These features and functions include which kind of machine system the software product operates, and which languages the product supports.

Hence, software products like Instagram, Facebook, that operate in MacOs and Microsoft are different programs. Similarly, country-oriented versions of these products are different programs, even if they have the exact same high-level functions. Furthermore, the current version of Instagram that has the story function, and the initial Instagram that does not have this function are two different products (Wang et al., 2014, p329).

2 How to explain change?

In order to defend that digital artifact objects survive thorough change, I apply "*origin-as-act*" theory to digital objects.

- "Origin-as-act" is necessary and sufficient to determine the identity of artifacts (Evnine, 2016).
- The identity of a given artifact is essentially determined by acts through which it is made, instead of the matter which the artifact is originally made out of (Evnine, 2016, p. 243).

- The identity of these actions is fundamentally determined by the intentions that lead those activities. Moreover, modal properties of artifacts are fundamentally determined by the creative intentions which bring certain objects into existence. (Evnine, 2016, p.245).
- The relation between an object and the creative activities of making them is constitution, and constitution is a "being made out of" relation (Evnine, 2009).
- In Evnine's account abstract objects such as musical works, fictional characters and languages are also artifacts which are produced by their producers, and they are constituted by some entities which are intentionally selected by their producers (Evnine, 2009, pp. 213-216).

3 Origin-as-act view's application to digital artifacts:

Digital artifacts are abstract objects which are mainly (if not only) made out of their algorithm.

4 An issue about revision:

i) On origin-as-act view, the making out of processes include revision processes of artifacts.

ii) Evnine argues that making out of activities should be performed whomever they are actually performed by (Evnine, 2016, p. 246).

Given that i) and ii), prima facie seems that revisions of digital artifacts can be exclusively made by their producers.

On the other hand, in the case of daily objects, Evnine argues that a restorer maintains a given object in a way that mimics the original activities of the producer (Evnine, 2016, p.76). Moreover, Evnine also argues that an artifact can be produced by individuals who share a nested intention in order to perform an overarching action together (Evnine, 2016, p. 236).

I argue that group production and revisions made by those other than the original producer of a given artifact might also be considered as multiple individuals acting upon nested intentions to perform an overarching action. The original intentions which actually brought into existence the given artifact can be understood similarly in the case of determining their modal properties: by

investigating their history of production. In the case of revision, the intentions involved in the revising process should be involved in the overarching action which produces and revises a given product.

Question: How and to what extent the intentions of the restorer and the original maker should act in the same way?

a) Extreme intentionalism: The original producer of a given product is always right about the individuation of her product. The maker's intentions determines how many products she produced.
b) Moderate intentionalism: The original producer can be wrong about the nature of her own product, and about her analysis of her own work.
According to extreme intentionalism:
i. If the producer of Instagram intends to produce different products, when he extends the product into different machine systems, then IOS version and Android version of the Instagram are two different objects.
ii. If the producer intends to revise Instagram while he is adopting the exact same changes, then IOS and Android versions of the product are the same product.

However, the history of production is the same in both cases. The same (or fundamentally very similar) making out of activities which brought into existence the product Instagram could have occurred if Instagram would have been produced to exclusively work on a different machine system than it has been actually produced for. The making out of activities by which Instagram was produced could have had a different constituent- a different machine system requirement- in its high-level requirements.

Even though the programmer introduces a new machine system to his product with an intention to produce a new product, he fails in his actions. The reason for this is that the producer does not engage in distinct and different acts of making while he is adopting new machine systems

or functions to his product. Instead, he makes an addition to his previous activities, and engages in revising activities of a pre-existent product. Therefore, I think moderate intentionalism should be adopted.

5 Distributed shared actions:

Question: Which intentions are involved in the overarching action of producing a product in cases of distributed shared actions?

Assume that a company wants to develop an operating system. Operating systems are software systems which coordinate and run other softwares on computers. A company manager allocates the work into several teams of programmers. The design team dedicates which features and functions will be included in the product. Another team is responsible for programming the interface of the product. A third team defines various features and functions of the product. Furthermore, individuals who program the product do not know about the end product. Only the design team is aware that the end product will be a certain kind of software system. However, individual programmers carry out the work load. They choose algorithms which fits certain required purposes, they write codes, they define requirements, etc. The end product involves the labor of various independent programmers. The problem is that which specific intentions actually bring into existence the end product is uncertain.

I think, the design team is the producer of the program. Even though the design team does not program any piece of the product, they intend to make an artifact. They choose or indicate ready-made objects that are programmed by other teams' programmers, and bring into existence a program. Moreover, it seems to me that if there were no intentions to create an operating system during the work distribution, and the some random combination of programmers, accidentally, turn out to be work as an operating system, then there would not be an operating system. Makers who work independently on a project who are unaware of the end product, do not produce the end

product. They produce pieces of constituent matter of the end product which eventually constitute the end product.

However, if the programmers were aware of the overarching program and worked to complete their tasks as an attempt to finish the project, then their actions would have been a part of the overarching action of producing the operating system.

To sum up, I think application of Evnine's constitution notion provides a plausible account of the nature of digital artifacts in accordance with our linguistic and non-linguistic behaviors concerning these objects.

REFERENCES:

Baker, L., R. (2004). The ontology of artifacts. *Philosophical Explorations, (7)*2, 99-111.

Baker, L., R. (2007). *The metaphysics of everyday life.* Cambridge: Cambridge University Press.

Evnine, S., J. (2009). Constitution and qua objects in the ontology of music. *The British Journal of Aesthetics, (49)*3, 203-217.

Evnine, S., J. (2016). *Making objects and events a hylomorphic theory of artifacts, actions, and o organisms*. New York: Oxford University Press.

Irmak, N. (2012). Software is an abstract artifact. *Grazer Philosophische Studien, (86)*1, 56-72.

Renear, A., H., Dubin, D., Wickett, M., K. (2008). When digital objects change-exactly what changes?. *Proceedings of the Association for Information Science and Technology, (45)*1, 1-3. (2011).

Wang X., Guarino, N., Guizzardi, G., & Mylopoulos, J. (2014). Software as a social artifact: A management and evolution perspective. Conceptual Modeling. ER 2014. Lecture Notes in Computer Science *(8824)*. Cham: Springer International Publishing.