Computational artifacts

interactive and collaborative computing as an integral feature of work practice

Schmidt, Kjeld; Bansler, Jørgen P.

Published in:
COOP 2016

DOI:
10.1007/978-3-319-33464-6_2

Publication date:
2016

Computational artifacts

Interactive and collaborative computing as an integral feature of work practice

Kjeld Schmidt* and Jørgen Bansler†

*Department of Organization, Copenhagen Business School, Denmark, and University of Siegen, Germany
†Department of Computer Science, University of Copenhagen, Denmark

Abstract. The key concern of CSCW research is that of understanding computing technologies in the social context of their use, that is, as integral features of our practices and our lives, and to think of their design and implementation under that perspective. However, the question of the nature of that which is actually integrated in our practices is often discussed in confusing ways, if at all. The article aims to try to clarify the issue and in doing so revisits and reconsiders the notion of 'computational artifact'.

Introduction

The key concern of CSCW research (and arguably of HCI too) is that of understanding computing technologies in the social context of their use, that is, as integral features of our practices and our lives, and to think of their design and implementation under that perspective. As Lucy Suchman expressed it in a programmatic article from 1993-94:

> 'Our efforts to develop a work-oriented design practice are based in the recognition that systems development is not the creation of discrete, intrinsically meaningful objects, but the cultural production of new forms of material practice. Our agenda […] is to bring developing objects out into the environments of their intended use, such that their appropriability into those environments becomes a central criterion of adequacy for their design. An implication of this agenda is that in place of the vision of a single technology that subsumes all others […], we assume the continued existence of hybrid systems composed of heterogeneous devices.' [39, p. 34]. (Cf. also [36, p. 99]).

In fact, there has, over many years, been a wide range of attempts to conceptualize, in Suchman's words, the 'appropriability' of such 'objects' into 'the environments of their intended use' so as to facilitate their 'artful integration'. Noteworthy examples of such attempts are notions such as 'artifacts in use' [1], 'appropriation' of interactive artifacts [6; 49], 'coordination mechanisms' [27], 'ordering systems' [28], 'socially embedded technologies' [12], 'artifact ecologies' [2], and 'practice-oriented' or 'practice-based' computing [18; 29; 30]. However, it seems fair to say that the large variety of proposed conceptions, and the obvious fact that nothing even remotely akin to consensus has emerged, indicates that the whole issue is still wide open.

Now, a strong contender in the competition to conceptualize the 'artful integration' of computing is the broad tradition that attempts to conceive of computing from an 'infrastructure' perspective [cf., e.g. 9; 13; 14-17; 21; 24; 34; 35]. But again the conceptions are hugely variegated if not contradictory. On one hand, Rob Kling and

Preprint
COOP 2016: 12th International Conference on the Design of Cooperative Systems,
23-27 May 2016, Trento, Italy

others introduced and have used the term ‘infrastructure’ to denote ‘all the resources and practices required to help people adequately carry out their work’ [15; 17]. On the other hand, Ole Hanseth and others use the term ‘information infrastructures’ to denote large-scale ‘computer networks with associated services’ [13, p. 409], not in the sense of ‘some kind of purified technology, but rather in a perspective where the technology cannot be separated from social and other non-technological elements, i.e. as an actor-network’ [14, p. 349]. And finally, Leigh Star and Karen Ruhleder have argued that the concept of ‘infrastructure’ should not simply be understood as a notion of large-scale technical structures but as a ‘fundamentally relational concept’: ’It becomes infrastructure in relation to organized practices’, that is, in as much as ‘it’ from the point of view of a particular ‘organized practice’ ‘sinks into the background’ and thus is treated as infrastructure [35]. In line with this rather boundless notion of ‘infrastructure’, ‘information infrastructures’, according to Star and Bowker, ‘provide the tools — words, categories, information processing procedures — with which we can generate and manipulate knowledge’ [34]. While the notion of ‘infrastructure’, like the range of other notions mentioned above, does serve to frame and focus on the issue of understanding computing technologies in the social context of their use, the disparities in its use are a source of immense confusion: In a given study, what is ‘the infrastructure’? A complex of interconnected technical artifacts or the social context in which it functions or even organizational arrangements that make it work? When talking about an ‘information infrastructure’, what is the ‘technology’ that ‘cannot be separated from social and other non-technological elements’? Is the ‘information infrastructure’ the underlying system of networked computers with associated software and protocols, or is it the applications or services running on that platform, or is it the network and services *in use*? Is it the artifact or the practice?

Moreover, what are the ‘heterogeneous devices’ that form the ‘hybrid systems’ of our work settings? Applications of computer technologies are surely used, often artfully, in conjunction with a host of other kinds of artifact, from pen and paper to tables and chairs, but does it make sense to create an abstract category of ‘heterogeneous devices’ and conceive of the motley of artifacts and materials of our lives as instances of such an all-encompassing category? While it certainly makes sense to point to the heterogeneity of our settings, distinctions are surely required.[1] In short, in order to overcome our bewilderment, not to mention to be able to compare findings from our studies, we need to be able to talk systematically about the entities that are to be ‘appropriated’ or ‘infrastructured’ or ‘artfully integrated’ into the manifold material settings of our practices.

Now, that is not as trivial a question as one might imagine. And this is probably one of the sources of our embarrassment. A simple question should illustrate our problem: What is it that is an application of the technical knowledge in computing? A ‘computer’, ‘a program’, a ‘system’? Do users actually ‘interact’ with a ‘computer’, and, if so, in which sense of ‘interaction’? With the computer ‘as a whole’, perhaps, but what would that mean? With the ‘application program’ and only now and then with ‘programs’ that are part of the operating system? With all these ‘programs’ as a whole or also with the ‘programs’ in conjunction with the CPU? With the ‘program’

[1] Similarly, what are the artifacts that make up ‘artifact ecologies? Are they devices, or apps, or services, or network protocols, or all of the above [2, p. 457]?

as conceived by the developers, with the ‘program’ as compiled and installed, or with the ‘program’ as instantiated as live circuits of running code? Or do users rather ‘interact’ with the objects of their work, whether digitally represented or not? Consider for example an ordinary digital calendar. It provides representations of categories such as ‘year’, ‘month’, ‘week’, ‘day’, ‘hour’, ‘day type’, ‘event’, and ‘event type’) as well as a set of operational primitives (e.g., ‘create event’, ‘set alarm’). It may also subscribe to data from services such as weather forecasts, national holidays, etc., just as it will exchange information such as event invitations, contacts, maps, etc. with other application programs. So far, it is straightforward to conceive of it as an application program with its specific data structure (in the form of one or several interconnected files). But when working with multiple devices (laptop, smartphone, tablet), what is displayed as one’s calendar may be distributed data structures that are continually and automatically synchronized via some cloud service. It is now no longer simply an identifiable application program with its files. One will nevertheless routinely use ‘it’ as a unitary calendar by virtue of the dependable and uniform replication of events in their relation to the generic data structure (a set of files, reciprocally synchronized). Now, a ‘shared’ calendar (shared by members of a project or a department) is not merely a stable set of files replicated across multiple devices. It is rather a composite of multiple individual calendars, partly intersecting, continually changing. And still, while distributed over myriad devices, it is routinely used as a unitary calendar. One is confident that what one sees in one’s calendar app is the ‘shared’ calendar. How do we systematically conceive of this?

These issues are more than technical: they are conceptual. This paper is an attempt to address these issues. We do not aim to develop a conceptual framework; we merely attempt to clarify the issue. In doing so, we are in the fortunate situation that the issue of what it is we ‘interact’ with when working with digital devices has been addressed before, 30 years ago in fact, under the label ‘computational artifact’ (or ‘interactive artifact’). This seems a good place to begin, not because what was achieved then solves our problems, which it does not quite do, but because the issue was raised and addressed squarely.

A logico-grammatical preamble

First, however, in order to steer clear of the metaphysics of computing, let us begin by making the conceptual observation that a bathtub is only a bathtub to a form of life for which taking a bath is an established practice. To an ant it is just a barren surface. Similarly, a technical artifact (a tool, a machine) is only a technical artifact by being a complement of a practice in which it is appropriated and routinely used. Or as Samuel Butler (1835-1902) wrote in his notebooks:

> ‘The very essence of a tool is the being an instrument for the achievement of a purpose. […] Therefore the word “tool”, implies also the existence of a living, intelligent being capable of desiring the end for which the tool is used, for this is involved in the idea of a desired end.’ [4, p. 19].[2]

[2] We are indebted to Bannon and Bødker for bringing Butler’s astute remarks to our attention [1].

An artifact such as a calculating machine may be able to perform ‘automatically’, i.e., proceed causally (for a period of time and under certain operational conditions) and without human intervention, but it does not make sense to say that it ‘calculates’ in and of itself. In the words of Ludwig Wittgenstein, in critical remarks aimed at Alan Turing:

> ‘Does a calculating machine *calculate*? Imagine that a calculating machine had come into existence by accident; now someone accidentally presses its knobs (or an animal walks over it) and it calculates the product 25 × 20.
>
> I want to say: it is essential to mathematics that its signs are also employed in *mufti*.
>
> It is the use outside mathematics, and so the *meaning* of the signs, that makes the sign-game into mathematics.’ [47, V §2].

The point is that *to calculate* is a normatively constituted activity; it is an activity governed by rules of what amounts to correct procedure and correct result: it is *a practice*. Wittgenstein again:

> ‘Turing’s “Machines”. These machines are *humans* who calculate.’ [48, §1096].

That is, it is *we* who, by manual control of tools and instruments or by the use of more or less automatic machines, do the work. Sure, the use of automatic machinery as part of our practices may have implications for these practices (educational, organizational, etc.), but they are nevertheless just that: technical complements of our practices. It only makes sense to talk about this *mechanical* (or *causal*) regularity from the point of view and in the context of the *normative* regularity of our practices in which these artifacts are integral technical complements [29, Chapter 13; 30; 32; 46]. In other words, it is *we* who engage in normatively constituted practices, *by* using rulers, compasses, and by using machines, computational artifacts included.

This should be clear enough. It is *we* who, in using computational (or interactive) artifacts, do the work. In that respect they are like any other kind of technical artifact: complements of our practices. This is not an issue here. The issue is rather that this class of artifact, computational artifacts, has some very special characteristics that are smothered when we, without further distinction, conflate them with other kinds of artifact or infrastructure.

The notion of ‘computational artifact’ in Suchman

The notion of ‘computational artifact’ was originally introduced by Lucy Suchman[3] in 1985 in a PhD-dissertation based on empirical studies at Xerox PARC and published

[3] The term ‘computational artifact’ is of course also used in methodological discourse in the completely different sense of contamination of data caused by the computational procedure. ‘Artifact’ in this derived sense generally means ‘something observed in a scientific investigation or experiment that is not naturally present but occurs as a result of the preparative or investigative procedure’. To make matters even more confusing, the exact same term is also being used in recent literature on the foundations of computer science [e.g., 45], without any reference to Suchman’s work whatsoever and in a very different sense, namely ‘the entities that computer scientists construct, the artifacts of computer science’ (*ibid*., §1). And computer science is defined as what? The science of computational artifacts?

as a technical report by Xerox PARC [37, pp. [iii], 1-12].[4] The notion of 'computational artifact' occupies an important place in the foundations of HCI, CSCW, and related fields of computing technology research, not because it is widely used, far from it, but because it figured prominently in the incontestably most influential attempt to formulate a conceptual foundation for this kind of research, namely, Lucy Suchman's *Plans and Situated Actions: The Problem of Human-Machine Communication*. She introduced and used it in her pathbreaking attempt to give a principled and clear articulation of what was to be addressed and explored in the research program of 'human-machine communication' or 'interaction': it was suggested as the term for *that* with which humans were supposedly 'interacting' or 'communicating'. By virtue of that role, and although the notion quickly all but dropped from circulation in HCI and neighboring fields, it was a cornerstone concept of her argument.

To understand Suchman's notion of 'computational artifact' it is important to keep in mind the context in which it was hatched. From 1979 to 1984, she was employed as a research intern at Xerox Palo Alto Research Center (PARC) in California, and upon having received her PhD degree in Social/Cultural Anthropology in 1984 she became a member of PARC's research staff, ultimately to be appointed Principal Scientist.[5] The point of this piece of biographical information is that she was working as a researcher at Xerox PARC when and where the ultimate conceptual development of the 'interactive computing' paradigm took place.

The 'interactive computing' paradigm was initially conceived in the early years of the Cold War in the course of the development of the Whirlwind computer (the key computer system for a new US air defense system, SAGE, which became operational in 1958 [23]). It received important further developments in the following years, with the early development of Computer-Aided Design at MIT [25; 42] and with the NLS ('oN-Line System') created in the 1960s at Douglas Engelbart's Augmentation Research Center at SRI International [11]. But the step that was decisive for establishing the paradigm was taken at Xerox PARC in the development in 1972-73 of an experimental computer 'workstation' with a 'graphical user interface', dubbed Xerox Alto [19; 43]. Based on the Alto concept, a more viable interactive computer named Xerox Star was developed (at PARC as well as the Xerox lab in El Segundo, California); it was introduced in 1981 [33]. That is, Suchman started working at PARC while the development of the Star was ongoing and finished her PhD dissertation at PARC only months after the ultimate exemplar of the interactive computing paradigm, the Apple Macintosh, was released up the road in Cupertino. Her dissertation was produced in the very epicenter — temporally, spatially, institutionally, culturally, and socially — of a technological upheaval of the first order.

The notion of 'computational artifact' was developed in order to be able to subject the 'interactive computing' paradigm that was then taking shape to critical examination.

[4] The dissertation was later published in a revised edition [38]. The two editions differ somewhat, most importantly in that the concluding chapter is greatly elaborated in the 1987 version. In this version the book has become a classic in CSCW and HCI. Suchman later republished the 1987 text with a lengthy introduction and under another title [41]. — The different editions of the book have together received almost 10.000 citations.

[5] http://www.lancaster.ac.uk/fass/sociology/profiles/lucy-suchman

Suchman was explicit about the context in which she developed the notion of 'computational artifact' and described it as follows:

'we now have a new technology which has brought with it the idea that rather than just using machines, we interact with them. In particular, the notion of "human-machine interaction" pervades technical and popular discussion of computers, their design and use. Amidst ongoing debate over specific problems in the design and use of interactive machines, however, no question is raised regarding the bases for the idea of human-machine interaction itself.' [37, p. 3; cf. 38, p. 1].

She did not cite examples of what she was referring to but did not need to either: she was referring to an idea that had become pervasive. The pioneers of the emerging technological paradigm had used different words to express this idea: 'human-computer team work' or 'conversation' [25], 'man-computer symbiosis' [20], 'human intellect' 'augmentation system' [10], 'man-machine communication system' [42], etc. The terminological variations notwithstanding, the core idea remained the same, namely that of a symmetrical relationship between human and machine.

Suchman's point of departure was that she found the very notion of 'interaction' between computing devices and humans problematic, i.e., in need of clarification. Thus, in examining the relationship between human and machine any preconceived notion of the specific nature of that relationship (such as 'interaction', 'communication', etc.) would anticipate what was to be investigated. In other words, she could not use a term like 'interactive artifact' in a critical investigation of the nature of that 'interaction'. The term 'computational artifact' was accordingly presented as a key term in formulating the research problem:

'The point of departure for this research is [...] the apparent challenge that computational artifacts pose to the longstanding distinction between the physical and the social; in the special sense of those things that one designs, builds, and uses, on the one hand, versus those things with which one communicates, on the other. While this distinction has been relatively non-problematic to date, now for the first time the term interaction — in a sense previously reserved for describing a uniquely interpersonal activity — seems appropriately to characterize what goes on between people and certain machines as well.' [37, p. 7; cf. 38, p. 6]. [6]

In short, Suchman introduced the notion of 'computational artifact' in the context of her effort to offer a critical corrective to 'the emergence of disciplines dedicated to making [computational] artifacts "intelligent"' and especially 'a practical effort' to subject the 'interaction between people and machines' to engineering [37, p. 9; cf. 38, p. 7].

Central to her concern with the notion of some symmetrical 'interaction' was the then much debated idea that computational artifacts might be able to 'explain themselves' and even give advice. She summarized the target idea as follows:

[6] Suchman's research problem was articulated in the same terms in 1999 in an article written in collaboration with Jeanette Blomberg, Julian Orr, and Randy Trigg: 'A central aim of Suchman's project was to suggest that the challenge of interactive interface design is actually a more subtle and interesting one than it was assumed to be by her colleagues in the field of human-computer interaction in the 1980s. Basically, their assumption was that *computational artifacts* just are interactive, in roughly the same that way persons are, albeit with some obvious limitations [...]. However ambitious, the problem in this view was a fairly straightforward task of encoding more and more of the cognitive abilities attributed to humans into machines in order to overcome the latter's existing limitations.' [40, p. 393. — Emphasis added].

> ‘Researchers interested in machine intelligence attempt to make [formal representations of plans] the basis for artifacts intended to embody intelligent behavior, including the ability to interact with their human users. The idea that computational artifacts might interact with their users is supported by their reactive, linguistic, and internally opaque properties. Those properties suggest the possibility that computers might explain themselves: thereby providing a solution to the problem of conveying the designer's purposes to the user, and a means of establishing the intelligence of the artifact itself.’ [37, p.[iii]].

To address this ‘problem of human-machine communication’ she conducted and reported on ‘a case study of people using a machine [a photo copier] designed on the planning model, and intended to be intelligent and interactive’ (*ibid.*). From this she concluded that the ‘access of user and machine to the situation of action’ is ‘asymmetrical’ in that ‘the ordinary collaborative resources of human interaction are unavailable’ to the machine [37, p. 124]. This insight was elaborated in the extended concluding chapter in the 1987 version of the text:

> ‘Today’s machines […] rely on a fixed array of sensory inputs, mapped to a predetermined set of internal states and responses. The result is an asymmetry that substantially limits the scope of interaction between people and machines.’ [38, p. 181]. ‘I have argued that there is a profound and persisting asymmetry in interaction between people and machines, due to a disparity in their relative access to the moment-by-moment contingencies that constitute the conditions of situated interaction. Because of the asymmetry of user and machine, interface design is less a project of simulating human communication than of engineering alternatives to interaction’s situated properties.’ [38, p. 185].

That is, the argument offered is that computational artifacts, in spite of ‘their reactive, linguistic, and internally opaque properties’, are cut off from ordinary human interaction because the latter is ‘situated’, i.e., ‘essentially *ad hoc*’.[7]

Suchman’s conception of ‘computational artifact’ has obvious merits.

First, by opting for the adjectival form ‘computational’ of ‘computer’ she shifted the emphasis from *thing* to *behavior*, from *object* to *functionality*. She thereby did not have to resort to common but notoriously confusing terms such as ‘the computer’, ‘the technology’, ‘the system’, ‘software’, ‘the program’, etc. For the problem with ‘interactive computing’ is the same irrespective of whether the device in question has the form of a permanent ‘hardware’ circuit, a temporary circuit in the shape of ‘firmware’, an ephemeral circuit in the shape of running ‘software’, or all of the above in conjunction, or whether the particular circuit is part of a CPU, the operating system, or some application program running under the control of the operating system, or whether the circuit resides as ‘embedded code’ in a photocopier or is a transient circuit distributed over multiple devices connected by some network.

Second, by using the term ‘artifact’ she was implicitly drawing on a century of anthropological work. For anthropology as well as archeology, ‘artifact’ is the standard term for objects designed, manufactured, and used by human cultures. It suggests that what we have here is comparable to stone axes, earthen pottery, and rattan baskets. The term ‘artifact’ was used in a similar deflationary way shortly after, by Pelle Ehn (‘computer artifacts’ [8]) and by Donald Norman and Edwin Hutchins (‘cognitive artifacts’ [22]).

[7] The proposition that situated action is ‘essentially *ad hoc*’ is deeply problematic (it leads to infinite regress) but that is not the issue here (for a critical discussion`, cf. [29, Chapter 12]).

In sum, the notion of 'computational artifacts' was introduced in order to address, in a principled way, the new phenomenon of humans working with machines that are highly reactive and malleable. It was introduced in the context of an attempt to dispel the mythology this new form of machinery had occasioned: the notion that we now are dealing with material objects imbued with human characteristics ('intelligence', 'language', etc.).

Suchman on the nature of 'computational artifacts'

According to Suchman's pioneering analysis, computational artifacts are characterized by having 'reactive, linguistic, and internally opaque properties' [37, p. [iii] and Chapter 2; 38, Chapter 2]. Let us briefly summarize what was meant by this proposition.

The 'reactivity' of computational artifacts: The characterization of computational artifacts as *reactive* seems obvious: Electronic computers are several magnitudes faster than any traditional machines and can react virtually instantaneously. However, as argued by Suchman, not all computing devices can be said to be *reactive*. As she emphatically points out, the reactivity of computational artifacts is predicated on 'the availability of interrupt facilities whereby the user can override and modify the operations in progress' [37, p. 10; 38, pp. 10 f.]. By contrast, computer devices used in batch processing modes (for the calculation of payrolls, tax returns, or master production schedules, or for the compilation of source code) are not reactive at all. They run their course. In stressing *reactivity*, that is, Suchman explicitly excluded the large body of computer programs used in mass data-processing from the category 'computational artifact'. The reason is obvious. It was the conceptual issues of the very concept of 'interactive computing' that were the target.

The 'linguistic properties' of computational artifacts: In elaborating what she meant by 'linguistic properties', Suchman stated that

> 'the means for controlling computing machines and the behavior that results are increasingly *linguistic*, rather than mechanistic. That is to say, machine operation becomes less a matter of pushing buttons or pulling levers with some physical result, and more a matter of specifying operations and assessing their effects through the use of a common language. With or without machine intelligence, this fact has contributed to the tendency of designers, in describing what goes on between people and machines, to employ terms borrowed from the description of human interaction — dialogue, conversation, and so forth: terms that carry a largely unarticulated collection of intuitions about properties common to human communication and the use of computer-based machines.' [37, p. 10 f.; cf. 38, p. 11].

It is a source of confusion that Suchman in the quoted passage distinguished 'linguistic' from 'mechanistic' means of control. Entering a 'command' ('`copy`', say) by typing it or by choosing a menu item or a button-like field on the display (whether or not labeled 'Copy') is not a linguistic act any more than pushing a tangible button on a dish washer or pulling a lever. They are simply different ways of activating a mechanism; in the case of computing machinery connecting an electric circuit, 'with some physical result'.

This indicates some deep-seated ambiguity, which is unsurprising given the historical context and the fact that this was a pioneering and hence also tentative effort. The 'tendency', to which Suchman referred, of designers 'to employ terms borrowed from the description of human interaction' that 'carry a largely unarticulated collection of intuitions about properties common to human communication and the use of computer-based machines' was simply not easily overcome.

Anyway, one can certainly say that computational artifacts have 'linguistic' (or, rather, *semiotic*) properties in the very limited sense that we can be use them to manipulate signs, where term 'manipulation' is shorthand for various transformative operations on the physical carriers of signs: Boolean operations and well as composite operations such as parsing, adding, subtracting, sorting, rearranging, moving, copying, comparing, etc. such (physical carriers of) signs and collections of signs.

The ability to manipulate (physical carriers of) signs is anyway not at all specific to or defining of computational artifacts. A mechanical typesetter, a traffic light, a time piece, or an electromechanical desktop calculator can also — in their own very limited way — be used to manipulate signs.

The 'internal opacity' of computational artifacts: Suchman's third defining characteristic, the computational artifact's being 'internally opaque', is *prima facie* mystifying. The entire argument is mystifying in that it, with references to Daniel Dennett [5] and Sherry Turkle [44], presumes that in order to understand a phenomenon, we have to reduce its behavior to internal mechanisms. But we do not need to understand the internal mechanism of an artifact in order to make rational use of it; nor do we in fact normally do that. One doesn't need, say, to understand the specifics of the lattice structure of steel alloys causing the operational properties of one's damascene kitchen knife: its hardness, its tensile strength, its elasticity. What one needs to understand is its 'functionality'. And in the case of machinery, what one needs to know is the dependable regularity of its behavior. That's all. The computational artifact is just a machine: a complex of elemental mechanisms carefully configured and built to perform with a high degree of predictability and dependability. And in incorporating it into our practice, we incorporate it *as such*: as a complex of causal mechanisms behaving in a highly regular manner which is useful in that practice. As long as the exhibited mechanical regularity can be integrated with the normative regularity of our practices, the internal mechanics is not of interest, except for very specific purposes, such as, for instance, maintenance, where we of course do take internal mechanisms into account.

Anyway, the point Suchman was trying to make by ascribing 'opacity' to computational artifacts was probably that we, in using a computational artifact, routinely treat it as a *functional unity* while disregarding its being an assembly of myriad computing devices:

> 'The overall behavior of the computer is not describable […] with reference to any of the simple local events that it comprises […]. To refer to the behavior of the machine, then, one must speak of "its" functionality. And once reified as an entity, the inclination to ascribe actions to the entity rather than to the parts is irresistible' [38, p. 15 f.].

Not only 'irresistible', we would like to add, but quite unproblematic. A computational artifact is indeed 'opaque', but the opacity is the result of deliberate design: mechanisms that are irrelevant to users are carefully hidden from view. That is, what Suchman (with Turkle) calls 'opacity' is what may just as well be called 'transparency'. Something is hidden for others things to be visible. In this case, the mechanism is hidden (opaque) to make visible what is relevant for the work. That is, their 'internal opacity' is what makes computational artifacts *transparent* in *users' terms*. What users need to 'see' are objects and operations pertinent to the practice in which the computational artifact is incorporated, that is, they need to be able to take for granted that the artifact behaves appropriately from the point of view of that practice.

In sum, then, on Suchman's conception, computational artifacts are designed to react in step with the activities in which they are used, to be incorporated in sophisticated semiotic practices, and to exhibit functional unity irrespective of its shifting internal constitution.

The notion of 'computational artifact' was introduced in order to address the nature of the 'interaction' in the then emerging 'interactive computing' technology and to do so in a way that did not subscribe, implicitly, to the then prevailing presumption of some kind of 'symmetry' between human and machine. By doing so, it oriented HCI and CSCW research towards the practices in which the artifacts are incorporated. This makes it an indispensable concept for technological research oriented towards developing applications for our work practice. The notion of 'computational artifact' has considerable merit.

Computational artifacts: A reconstruction

Let us first briefly reconstruct the conception of 'computational artifact', that is, try to clarify its rational core while avoiding the ambiguities and logical mistakes that were probably unavoidable birthmarks.

Just as we do not live in the blueprint of a house but in the building specified in the blueprint, when we talk about 'computational artifacts' (e.g., a 'shared calendar') we are not talking about 'computer programs', or the various stages of program specification as expressed in diagrams and code; we are rather talking about programs as running code (in conjunction with one or more CPUs and assorted other computing components), that is, we are talking about programs as *live electronic circuitry*, be it permanent or ephemeral. It is *that* with which we interact when using computing devices.

A key feature of computational artifacts is their capacity to react to and thus be integral part of unfolding events. Several architectural features are required to afford this.

(1) For the reactions of the device to be experienced as immediate and the device to be experienced as 'interacting', reactivity presumes a response-time that is less than (what in the context of a given practice is considered) state-changes in the environment, i.e., they must be able to execute in 'real time'. This is elementary.

(2) In the stored-program architecture originally devised by Alan Turing and John von Neumann, programs in the compiled form of binary code are treated as data and, when launched, reside in the computer's storage as — invisible and transient but no less physical — electronic circuits that can be activated virtually instantaneously. The advantage of this architecture was originally that the program only has to be launched once and that the computing device does not have to spend time to access external input devices to obtain the next command.

(3) The reactivity of computational artifacts is not afforded by the stored-program architecture in itself but presumes the very specific mode of operation classically referred to as 'manual intervention' [26] or, in Suchman's words, 'the availability of interrupt facilities whereby the user can override and modify the operations in progress' [37, p. 10; 38, p. 11]. Under this paradigm, the computing device does not

simply run through a prespecified set of 'commands' in a prespecified sequence; rather, a program launched into working storage and activated awaits the occurrence of certain external events to execute a 'command' (in source code expressed as, for example, '`on mouse-up, do X`').

(4) By virtue of their reactivity, as well as high-capacity gigahertz storage media, computational artifacts are used to manipulate signs (letters, numerals, geometric elements, patterns of imagery and sound) in step with external events. They are used as complements of semiotically constituted practices (calculating, computing, drafting, planning, writing, searching, etc.). It is this that occasions some to ascribe normative behavior ('rule-following') to automatic machinery, others to latch on to a mechanical model of the mind to account for rational conduct, and others again to cover-up their embarrassment by conceiving of 'interactive computing' (or 'human-machine systems') as a symmetrical relationship. The source of the befuddlement lies in the confusion of the 'grammatical' or conceptual distinction between normative and mechanical regularity. We should not allow ourselves to be misled by the apparent ability of computing devices to react usefully to written or spoken instructions or provide results in similar modalities.[8] We should remember that computing machines do not use *signs* any more than Tibetan prayer mills engage in *praying*. What makes computational artifacts different from classical-mechanical and electromechanical machinery in regard to their ability to manipulate (physical bearers of) signs is strictly their speed of operation together with the fact that they can manipulate vast amounts of data in real-time, because of related features such as high-speed inexpensive random-access storage technologies (RAM, flash storage, harddisks).

(5) A computational artifact is constituted by a structured set of 'object classes' and 'objects' (i.e., data structures with associated elementary operations). Accordingly, a computational artifact is inherently bound to *a practice* from which the categories represented by these objects and classes are derived and to which they are indigenous. It is within such practices, and only there, that the objects of the computational artifact have meaning.

An electronic calendar, for example, is bound to our institutionalized practices of time measurement for purposes of planning and organizing events such as harvesting, meetings, deadlines, travels, and is constituted by object classes such as 'time', 'event', and so forth, as well as the associated operations. The functional unity of the digital calendar is a manifestation of the dependable reproduction of this structure, irrespective of the distribution of objects and operations, and the circuitry in which they are realized, across computer entities and devices. Only the objects are made visible, the internal mechanics are not (they are transparent or opaque).

A CAD system similarly offers a family of elemental object classes (such as, in architecture, 'wall,' 'door', 'window', etc.) and complex object classes (such as 'floor plan', 'elevation', 'cross section', 'building'), and requisite operations on these types

[8] It is of course common for computer scientists and engineers to refer to the highly regular patterns of behavior of computational artifacts, that are the hallmark of their art, as 'rules' or 'procedures' (just as they speak of 'programming languages', etc.). This is an unavoidable feature of the natural attitude of these professions. Within the bounds of the practices of devising and building computational devices, it is as harmless as when we in ordinary language use expressions like 'sunrise' and 'sunset'. Damage only arises when this innocently convenient language is incorporated into systematic conceptualizations.

(such as 'locate', 'copy', 'move', 'group', etc.). Modern CAD systems furthermore offer the advanced functionality of abstraction called 'overlay drafting' in which the representation of a composition is divided conceptually into layers such that objects and object complexes can be separated according to, for example, the material involved (brick, concrete), the type of object (walls, doors, plumbing), or author, while the geometric projection is rigorously maintained across layers (cf. [7, §2; 28; 31].). This enables architects and engineers to work with discrete digital representations of different aspects of the overall composition in a manner such that the parts created or amended by each of them are immediately available to any other, in various combinations, by direct projection between or superimposition of their respective layers. That is, they interact by making state changes to a set of (digital) representations while at the same time ensuring overall consistency of their distributed activities by means of the functionality of an associated assembly of computational artifacts carefully designed to perform the projection calculus in real time.

Thus, although a computational artifact incorporates the behavior of multitude, physically distinct, elemental entities, perhaps distributed over a multiple devices, it exhibits *functional unity* over these myriad events without necessarily involving users in its intricate 'internal affairs', and it does so by virtue of its being bound to a practice. The functional unity exhibited by computational artifacts-in-use is the manifestation of just that — a design feat of the highest magnitude.

(6) When a program, under the stored-program paradigm, is compiled and stored in RAM and thus exists as live circuitry, the device can not only react but may — in response to (determinate) environmental conditions, especially to actions users might take — change behavior (within the determinate but vast, in practice perhaps incalculable, solution space constituted by the totality of possible combinations). It may also modify its own 'code' or regular behavior (or that of another running program) and do so in 'real time'.[9] Accordingly, the reactivity of computational artifacts obtained under the interactive computing paradigm is afforded by far more than simple operational speed; computational artifacts may change behavior quasi-dynamically as environmental conditions change. Reactivity thus also implies behavioral flexibility and variability.

(7) In an operating environment facilitated by the stored-program architecture, 'programs' no longer necessarily exist and act in splendid isolation. When active, a program may 'call' another program, even a program residing on a remote device, to execute a given 'command' (e.g., perform a specific process and return a data string) before it proceeds, or it may simply, under certain (determinate) conditions, hand over 'control' to another program (which operating systems do all the time). This is of course elementary. The point, however, is that in this environment, running programs may be made to combine and recombine to form large but often ephemeral machine systems. Multiple computational artifacts can be combined and recombined in real time, just as they can form more or less stable machine systems through remote resource calls or linking. Just think of a trivial browser operation involving preference files, java applets, browser extensions, queries to remote database system facilitated by CGIs, etc., and the invocation of other applications to handle special file types (such as PDF) — not to mention the OS level machinery making all this possible.

[9] This is what makes hacking possible.

Computational artifacts in practice

The point of the concept of 'computational artifact' is to enable us to conceive of its dynamically accomplished functionality as integral to a work practice.

As with any technical artifact, a hammer, a power loom, or a computational payroll calculation, computational artifacts are complements of our work practices; beyond that, they would be merely evanescent electromagnetic pulses. A computational artifact is bound to the practice of which it is a technical complement by being used — routinely, competently, repeatedly — to process signs (in the form of object classes and objects) inherent to that practice. But computational artifacts are also complements of our practices in a different and stronger sense. As technical accompaniments of *interactive* computing, and even more so as *collaborative* computing applications, their overall behavior is determined in the course of the activities in which they are incorporated. Now, a computational artifact is of course a material artifact, an assembly of live electronic circuitry. It is carefully designed to behave in a highly regular way. When not in use, i.e., in the absence of 'manual intervention', while in the 'wait loop', it is like an idling car engine, and in this state its behavior is well defined: it does nothing but 'wait'. However, *in use* its behavior is determined by the pattern of activity of which the interventions are part. For each intervening act, its behavior (its response to each 'command' or 'mouse-up' or 'swipe': the 'subroutines' activated, the data exchanged, the remote calls, etc.) is of course strictly mechanically determined. For each state in the process of its dynamic execution there is a finite set of possible moves, but once a move has been made (a key is pushed, a button tapped, a dial turned) the artifact undergoes an 'automatic' (highly regular causal) process. And so on. However, the overall pattern of behavior of the artifact-in-use is not one of mechanical regularity (in contrast to a computational payroll calculation); the overall behavior is a manifestation of the normative regularity of the practice in which it is used as a material artifact. When integrated into our practices, the causal regularities of the enacted computational artifacts — the shifting electronic circuitry — are interwoven into the patterns of the normatively regulated activities of these practices. It is this fine-grained interweaving of normatively regulated activities and mechanical regulation that is specific in computational artifacts.

The conception of computational artifacts outlined here may also help clarify the notion of 'infrastructure': (a) There is first 'infrastructure' in the sense of a technical complement of a given cooperative work practice or family of practices: an assembly of computational artifacts constituted by object classes conceptually indigenous to that practice or family of practices. Obvious examples are the 'information infrastructures' provided by what in CSCW is often termed 'coordination technologies': large-scale collaborative applications such as CAD systems, electronic patient records, document management systems, workflow management systems (and of course also group calendar systems). These are assemblies of computational artifacts that exhibit functional unity in use by virtue of the complex of domain-specific object classes (workflows, classification schemes, etc.) they all embody. (b) Then there is 'infrastructure' in the sense of (more or less stable) patterns of live circuitry, established dynamically, typically by cascades of automatic connections unleashed by manual intervention in the use of the practice-bound infrastructures, and facilitated by the 'information super-highway' or 'infrastructure' of the multitude computer devices (servers, laptops, smartphones), network facilities (routers, fiber-optic cables, electromagnetic fields), and the multitude network protocols and APIs by means of which the formation of the former are controlled and mediated. Infrastructures, in this second sense, are facilities that enable 'infrastructures' in the first sense to form; the infrastructure of the infrastructure, if you will. (c) And of course, there is also, to make the picture complete, 'infrastructure' in sense of the 'work to make the network work' [3]: the organized practices through which 'infrastructures' in the second sense are constructed and maintained.

In sum: In the design of computational artifacts for the purpose of serving in a regulating capacity in coordinative practices (a group calendar, a workflow system, a document management system), we construct a repertoire of potential electronic circuitry that can be activated and combined in multiple but determinate ways and that then, in step with the unfolding actions of the cooperating actors, are used in the coordination of interdependent activities. In other words, what is designed are mechanisms that function in a strictly causal manner but interactively: step by step, move by move. A major challenge in this, if not *the* challenge, is the following: (a) to identify, analytically, not only the governing schemes of the coordinative practice in question (categories, principles) but also the spectrum of variations, contingencies, routine troubles, etc. that practitioners have to deal with, as well as the sources of these variations, etc., the probabilities of their occurrence, and the ways in which they deal with these variations: contingency measures, improvisations, and workarounds; (b) to devise computational artifacts with a corresponding space of possible moves — including, of course, a repertoire of means of improvisation enabling workers to temporarily circumvent or respecify the behavior of computational artifacts; and (c) to devise means for workers to express the categories and principles of their coordinative practices in the behavior of computational artifacts, and so on. In short, the real challenge is that of developing machinery designed to be an integral part of our normatively constituted but contingent cooperative work practices: the challenge of meeting the requirements of *normatively* constituted practices by means of *mechanical* machinery while at the same time dealing with the contingencies of our work.

Acknowledgments

The research reported in this article has been supported by the Velux Foundation under the 'Computational Artifacts' project.

References

[1] Bannon, Liam J.; and Susanne Bødker: 'Beyond the interface: Encountering artifacts in use', in J. M. Carroll (ed.): *Designing Interaction: Psychology at the Human-Computer Interface*, Cambridge University Press, Cambridge, 1991, pp. 227-253.

[2] Bødker, Susanne; and Clemens N. Klokmose: 'Dynamics in artifact ecologies', *NordiCHI'12*, ACM Press, New York, 2012, pp. 448-457.

[3] Bowers, John M.; Graham Button; and Wes W. Sharrock: 'Workflow from within and without: Technology and cooperative work on the print industry shopfloor', *ECSCW'95*, Kluwer Academic Publishers, Dordrecht, 1995, pp. 51-66.

[4] Butler, Samuel: *The Note-Books of Samuel Butler*, A. C. Fifield, London, 1912.

[5] Dennett, Daniel C.: *Brainstorms: Philosophical Essays on Mind and Psychology*, MIT Press, Cambridge, Mass., 1978.

[6] Dourish, Paul: 'The appropriation of interactive technologies: Some lessons from placeless documents', *Computer Supported Cooperative Work (CSCW)*, vol. 12, no. 4, December 2003, pp. 465-490.

[7] Eastman, Charles M.: 'Why we are here and where we are going: The evolution of CAD', in: *ACADIA 1989*, pp. 9-26.

[8] Ehn, Pelle: *Work-Oriented Design of Computer Artifacts*, Arbetslivscentrum, Stockholm, 1988.

[9] Ellingsen, Gunnar; and Kristoffer Røed: 'The role of integration in health-based information infrastructures', *Computer Supported Cooperative Work (CSCW)*, vol. 19, no. 6, December 2010, pp. 557-584.

[10] Engelbart, Douglas C.: *Augmenting Human Intellect: A Conceptual Framework* Stanford Research Institute, Menlo Park, Calif., October 1962.

[11] Engelbart, Douglas C.; and William K. English: 'A research center for augmenting human intellect', in: *FJCC'68. Part I*, vol. 33, AFIPS Press, New York, 1968, pp. 395-410.

[12] EUSSET: *A Position Statement*, European Society for Socially Embedded Technologies, 2013. (Originally posted 2009). <http://eusset.eu/position-paper/>

[13] Hanseth, Ole; Erik Monteiro; and Morten Hatling: 'Developing information infrastructure: The tension between standardization and flexibility', *Science, Technology, & Human Values*, vol. 21, no. 4, October 1996, pp. 407-426.

[14] Hanseth, Ole; and Nina Lundberg: 'Designing work oriented infrastructures', *Computer Supported Cooperative Work (CSCW)*, vol. 10, no. 3-4, September 2001, pp. 347-372.

[15] Jewett, Tom; and Rob Kling: 'The dynamics of computerization in a social science research team: A case study of infrastructure, strategies, and skills', *Social Science Computer Review*, vol. 9, 1991, pp. 246-275.

[16] Karasti, Helena; and Anna-Liisa Syrjänen: 'Artful infrastructuring in two cases of community PD ', *PDC 2004*, vol. 1, ACM Press, New York, 2004, pp. 20-30.

[17] Kling, Rob: 'Defining the boundaries of computing across complex organizations', in R. J. Boland, Jr. and R. Hirschheim (eds.): *Critical Issues in Information Systems Research*, Wiley and Sons, New York, 1987, pp. 307-362.

[18] Kuutti, Kari; and Liam J. Bannon: 'The turn to practice in HCI: Towards a research agenda', *CHI'14*, ACM Press, New York, 2014, pp. 3543-3552.

[19] Lampson, Butler W.: *Why Alto*, XEROX Inter-Office Memorandum, Xerox PARC, Palo Alto, Calif., 19 December 1972.

[20] Licklider, Joseph Carl Robnett: 'Man-computer symbiosis', *IRE Transactions on Human Factors in Electronics*, vol. HFE-1, no. 1, March 1960, pp. 4–11.

[21] Monteiro, Eric, *et al*.: 'From artefacts to infrastructures', *Computer Supported Cooperative Work (CSCW)*, vol. 22, no. 4-6, August-December 2013, pp. 575-607.

[22] Norman, Donald A.; and Edwin L. Hutchins: *Computation via direct manipulation*, Institute for Cognitive Science, University of California, San Diego, La Jolla, California, 1 August 1988.

[23] O'Neill, Judy Elizabeth: *The Evolution of Interactive Computing through Time-sharing and Networking*, PhD dissertation, University of Minnesota, 1992.

[24] Pipek, Volkmar; and Volker Wulf: 'Infrastructuring: Toward an integrated perspective on the design and use of information technology', *Journal of the Association for Information Systems*, vol. 10, no. 5, May 2009, pp. 447-473.

[25] Ross, Douglas T.: 'Gestalt programming: A new concept in automatic programming', *AIEE-IRE'56 (Western)*, ACM Press, New York, 1956, pp. 5-10.

[26] Ross, Douglas T.; and John E. Ward: *Investigations in Computer-Aided Design for Numerically Controlled Production: Final Technical Report: 1 December 1959 – 3 May 1967*, Electronic Systems Laboratory, MIT, Cambridge, Mass., May 1968.

[27] Schmidt, Kjeld; and Carla Simone: 'Coordination mechanisms: Towards a conceptual foundation of CSCW systems design', *Computer Supported Cooperative Work (CSCW)*, vol. 5, no. 2-3, June 1996, pp. 155-200.

[28] Schmidt, Kjeld; and Ina Wagner: 'Ordering systems: Coordinative practices and artifacts in architectural design and planning', *Computer Supported Cooperative Work (CSCW)*, vol. 13, no. 5-6, December 2004, pp. 349-408.

[29] Schmidt, Kjeld: *Cooperative Work and Coordinative Practices: Contributions to the Conceptual Foundations of Computer-Supported Cooperative Work (CSCW)*, Springer, London, 2011.

[30] Schmidt, Kjeld: 'The concept of 'practice': What's the point?', *COOP 2014,* Springer, London, 2014, pp. 427-444.

[31] Schmidt, Kjeld: 'Of humble origins: The practice roots of interactive and collaborative computing', *ZfM Online*, May 2015. <http://www.zfmedienwissenschaft.de/online/humble-origins>

[32] Shanker, Stuart G.: *Wittgenstein's Remarks on the Foundation of AI*, Routledge, London, 1998.

[33] Smith, David Canfield*, et al.*: 'The Star user interface: An overview', *AFIPS'82,* AFIPS Press, Arlington, Virginia, 1982, pp. 515-528.

[34] Star, Susan Leigh; and Geoffrey C Bowker: 'Work and infrastructure', *Communications of the ACM*, vol. 38, no. 9, September 1995, p. 41.

[35] Star, Susan Leigh; and Karen Ruhleder: 'Steps towards an ecology of infrastructure: Complex problems in design and access for large-scale collaborative systems', *Information Systems Research*, vol. 7, no. 1, March 1996, pp. 111-134.

[36] Suchman, Lucy: 'Located accountabilities in technology production', *Scandinavian Journal of Information Systems*, vol. 14, no. 2, 1 January 2002, p. 7.

[37] Suchman, Lucy A.: *Plans and Situated Actions: The Problem of Human-Machine Communication*, Xerox Palo Alto Research Center, Palo Alto, Calif., February 1985.

[38] Suchman, Lucy A.: *Plans and Situated Actions: The Problem of Human-Machine Communication*, Cambridge University Press, Cambridge, 1987.

[39] Suchman, Lucy A.: 'Working relations of technology production and use', *Computer Supported Cooperative Work (CSCW): An International Journal*, vol. 2, no. 1-2, March 1993 [article dated 1994], pp. 21-39.

[40] Suchman, Lucy A., *et al.*: 'Reconstructing technologies as social practice', *American Behavioral Scientist*, vol. 43, November 1999, pp. 392-408.

[41] Suchman, Lucy A.: *Human-Machine Reconfigurations: Plans and Situated Actions, 2nd Edition*, Cambridge University Press, New York, 2007.

[42] Sutherland, Ivan Edward: *Sketchpad: A Man-Machine Graphical Communication System*, PhD diss., Lincoln Laboratory, MIT, Cambridge, Mass., 30 January 1963.

[43] Thacker, Charles P.*, et al.*: *Alto: A personal computer*, Xerox PARC, 7 August 1979.

[44] Turkle, Sherry: *The Second Self: Computers and the Human Spirit*, Simon & Schuster, New York, etc., 1984.

[45] Turner, Raymond: 'The philosophy of computer science', *The Stanford Encyclopedia of Philosophy*, 20 August 2014.

[46] Williams, Meredith: 'Blind obedience: Rules, community, and the individual', in M. Williams (ed.): *Wittgenstein's Philosophical Investigations: Critical Essays*, Rowman & Littlefield Publishers, Lanham, Maryland, 2007, pp. 61-92.

[47] Wittgenstein, Ludwig: *Remarks on the Foundations of Mathematics* (Manuscript, 1937-44). Transl. from *Bemerkungen über die Grundlagen der der Mathematik*. Basil Blackwell Publishers, Oxford, 1978.

[48] Wittgenstein, Ludwig: *Remarks on the Philosophy of Psychology*. *Volume I* (Typescript, Fall 1947). Basil Blackwell Publisher, Oxford, 1980. – [TS 229].

[49] Wulf, Volker, *et al*.: 'Engaging with practices: Design case studies as a research framework in CSCW', *CSCW'11*, ACM Press, New York, 2011, pp. 505-512.