

1

Evolution of Architectural Software

Sébastien BOURBONNAIS

Université Laval, Quebec, Canada and

Asynth SAS, Lyon, France

The certainty that the technical evolution goes by its own logic in the direction of the progress of the man is not really accepted anymore, but this doubt does not undermine the evidence that there is a technical evolution

Xavier GUCHET (2005, pp. 23–24).

1.1. Introduction

The software used by architects has evolved, and continues to do so, according to different factors that, at their own pace, modify their practices to varying degrees. In order to understand this evolution, it is important to try to combine different factors in order to not only compare software, but also encompass the modalities and conditions that have made it possible for them to appear. Why did this type of software appear at this time? In such a context? If there is certainly an evolutionary trajectory of software, in reality, this trajectory is strongly disrupted according to its internal and normative logics, and sometimes even aligns itself perfectly with the questions of the architects occupied with these technologies, rather than following a progression only dictated by intrinsic technical potentialities. This introduces power relationships that are reflected in the orientations taken by software developments. In addition, at a broader level, a psycho-sociological and cultural context is added,

Architectural Design Instruments,

coordinated by Sébastien BOURBONNAIS. © ISTE Ltd 2023.

which brings its own expectations into the course of this evolution. It is through this complex intricacy that we must approach this evolution of software mainly intended for architectural practices.

The aim of this chapter is to participate in the ongoing construction of a software culture among architects, following the example of the various thinkers who have tried to raise awareness of the meaning of technology and its objects. In order to establish our analysis, we will present different events, software, objects, practices and debates to understand some key elements of this complex interweaving. Indeed, a number of relatively important events related to computer science and architecture have taken place. They have allowed us to report on the debates that took place and raise problems or questions that architects have faced with the arrival of these technologies. A software series, flagship, detects the implicit ways of thinking about the design, while underlining the limitations of their understanding and explanation. It will also be a question of classifying software, an arduous task to grasp some of the trajectories that have structured the appropriation of software by the architectural community: mainly architect-practitioners and architect-researchers linked to different research institutions.

Starting from the initial CAD project, carried out at MIT in 1959, and from the 1964 meeting, held in Boston, on the theme of the computer and architecture, the aim was to highlight the different types of relationships that had been established between architects, computers and software, mainly in the context of design activity. The choice to focus on design was not arbitrary, but rather corresponded to the choices made by architects, at the turn of the 1970s, to use computers for this activity in particular, to the point of calling it into question. While the achievements of the design programs of this period are generally linked to research institutions, the 1980s, thanks to a number of major technical innovations, saw the creation of a new software market, whose main actors no longer had the same links with the architectural community. This led to a rapid growth of computer technologies, on several of its components, as well as on its manageability and accessibility. These advances saw the computer introduced in a massive way in architectural agencies at the beginning of the 1990s. The computer quickly became an instrument to explore differently, generating a singular episode in the history of digital architecture. Finally, we will end this great panorama on the new bifurcation that has occupied architecture for the last 20 years, with increasingly user-friendly software and the practices described as BIM.

This historical picture also aims to put forward the idea that many of the issues that occupy contemporary architects have persisted for a very long time and that, for many of these questions, they are repeated in almost exactly the same terms, but from a different point of view, in a different context, with new technologies. The

presentation of these different preoccupations, placed in their historical contexts, will also be an opportunity to highlight the processes put in place by architects in order to integrate technologies into their practices. It is a question, no more and no less, of establishing trajectories of appropriation which, undeniably, will have had repercussions on the ways of designing architecture.

1.1.1. *Difficulty of classification*

This work on the classification of software aims to put forward specificities and thus understand the fundamental differences of them, in relation to one another. Our bet is to build this method of analysis and classification from the *mechanology* project, the science of the machines, developed at the beginning of the last century. After a short presentation, it will be a question of following the evolution of this project, and then introducing modifications that are able to adapt to our object of study. This detour, as will be explained below, allows us above all to propose an alternative to software studies. It also allows for a somewhat different arrangement from that established in the exhibition *The Architecture Machine*, which was held in Munich (Fankhänel and Lepik 2020). According to the catalog, the exhibition was divided into four main chapters: the computer as a drawing machine; the computer as a design tool; the computer as a medium for storytelling; and finally, the computer as an interactive platform. These distinctions allow for a first efficient and relevant categorization, which we will in some way deepen and nuance. Our attention will be mainly focused on the emergence of these categories: to see how they are installed within the architectural debates of the time and especially, how they transform themselves, the ones in relation to the others, continuously, in a dynamic way. It is for all these reasons, among others, that we wish to return to the mechanological project.

A mechanology project was first proposed by the French architect and engineer Jacques Lafitte. “The science of machines, or mechanology, a normative science, has no other goal than the study and explanation of the differences that are observed between machines” (Lafitte 1972, p. 32). To this mechanology, a *normative science*, Lafitte added, while distinguishing them, a *descriptive science* “devoted to the history, description, and classification of existing machines” and an *art of building* machines. These three branches can be used as a basis for a reflection on software. Let us also underline that the main interest, as much for Lafitte as for our project, lies in the study of differences, because it is thanks to these that Lafitte managed to raise “the very problem of their existence” (*Ibid.*). This is indeed the objective that we set ourselves with this study, that of raising some of the cogs that allow us to shed light on what could be called *the raison d’être* of a software program; by

describing it, by showing its genesis and functioning when possible and, above all, by underlining its differences with other software programs.

Mechanology did not remain at the stage presented by Lafitte, but evolved in at least three phases (Guffroy and Bontems 2018). As Giovanni Carrozzini noted, a similar project was elaborated by the philosopher Gilbert Simondon, while differing from it on some points. If Lafitte wished to classify: “the machines invented by man during the centuries, Simondon developed an analysis that aimed for the understanding of the roots and dynamics underlying the act, the creative and inventive human gesture that are, in his opinion, the true origin of the technical objects”. (Carrozzini 2009, p. 30) For Simondon, it is not a question of trying to understand closed technical objects, nor of simply classifying them, but of seizing them in their genetic and dynamic aspect, in their “entelechy” (Simondon 2014a, p. 300), which is opposed to the inactivity and the static state of the machine; the philosopher thus aims to seize the objects being *realized* in the action. His approach also differs in terms of the theoretical tools – notions – employed to capture this evolution of objects:

Lafitte’s approach remains completely linked to the traditional scientific method, whose goal is to simplify the complexity of reality, by using schemes, such as those of *species* and *individual* according to Aristotle. [...] On the other hand, Simondon tried to go beyond this opposition [...]. The law of the *recurrent causality* and that of the *relaxations* are not laws *manufactured* by science, by the *thought that reflects on the techniques*: they are laws that the Simondonian reflection *discovers* inside the evolution of the techniques. Moreover, the history of the techniques is not a continuous evolution, such as that of Lafitte... (Carrozzini 2009, pp. 37–38).

While it seems obvious that the technical evolution cannot be thought of in a continuous way, but rather in a “sawtooth” way, we wish to insist on the heuristic effort that will be necessary in order to seize the specificity of this software evolution: how do we define the laws which govern this evolution?

Another distinction that Carrozzini mentions is that Simondon will not only focus on machines, but will integrate a reflection on tools and utensils, and will even integrate a study of information machines and networks. Even if we wish to make this new arrangement of mechanology¹ the basis of our analysis, as Guffroy and

1. As both Carrozzini and Guffroy and Bontems underline, it is not a question for Simondon of taking up Lafitte’s project and improving or modifying it, because, as these authors indicate, it is highly probable that Simondon only became aware of Lafitte’s work after

Bontems point out, this project remains complicated to implement without methodological explanations, as proven by certain attempts made at the end of the 1990s with the notions of technical lineages.

We nevertheless persist in thinking of the relevance of these reflections to begin this work. This reference also allows us to distinguish ourselves from software studies, as exposed by Manovich (2013). Despite the merits of this work in placing software and its interface at the heart of the analysis, our study will differ from it on a few points. If Manovich also uses the notion of evolution, he uses it, by his own admission, in a trivial way without trying to theorize it, relying on a simplified version of Darwinian evolution. On the other hand, his study is based on a problem of the media. Indeed, his main objective is to displace, or even convert, the theory of software (software for editing video, text and image content) into a theory of media: as meta-media. As Manovich himself points out, the medium places and engages reflection on the symbolic and cultural domains: “In short, the computer has moved from being a culturally invisible technology to being the new engine of culture within a decade” (Manovich 2013, p. 21).

Another distinction that we wish to make with respect to Manovich’s study is based on the amalgam suggested by his undifferentiated analyses of software interfaces, whose category, content editor, seems to us to be too broad and, above all, leaves out certain nuances and singularities specific to the contexts in which this software is inserted. As mentioned above, our intention is, on the contrary, to raise the differences between software rather than to try to define their common characteristics. There is no question of making them converge in order to see a great digital turn, nor of understanding them as meta-media. On this point, our position remains close to that defended by the philosopher Jean-Michel Salanskis as to the directing place of the context:

The mutations of the informational revolution, consequently, would not be so much a great and unique disruption, confirmed by its innumerable illustrations, as a heterogeneous plurality of mutations, each substantial but each local, and relative to a context that remains guiding, and is in this sense not truly “revolutionized” (Salanskis 2011, p. 42).

writing his major work on techniques, *On the Mode of Existence of Technical Objects* 1958, but also his course on the psychosociology of technicity and his technical mentality text. For this reason, these two mechanology projects must be taken independently, even if, as these authors suggest, many comparisons are possible.

It is in this sense that particular attention will be paid to the specificity of the contexts in which software appears, develops and transforms. It is for this reason that the analysis of software is not limited to its interface, nor to its programming, but seeks to integrate the user communities that are associated with it, and that participate by the same fact in its creation process. It is moreover because of this opening of perspective that it becomes impossible to establish technical lineages (or lineage diagrams) and that this principle of categorization remains difficult to hold until the end.

It will not be a question of drawing lineages, strictly speaking, but of pointing out the different factors that intervene in the orientations that software development takes. It will be a question here of being attentive to the moment of change, when the main trajectory bifurcates, when another posture is imposed, when a new mentality is openly displayed. We take the way Bergson defined evolution seriously:

The evolutionary movement would be a simple thing, we would have quickly determined its direction, if life described a unique trajectory, comparable to that of a full ball launched by a cannon. But we are dealing here with a shell that immediately burst into fragments, which, being a kind of shell themselves, in turn burst into fragments destined to burst again, and so on for a very long time (Bergson 1998, p. 99).

Before discussing some of the main events that took part in this encounter between architects and computers, let us first present the context in which they were held. We will raise some of the positions that were expressed in the early 1960s regarding the relationship that we humans should have had with the first computer machines. We will then present the distinction that we wish to establish between digital conditions and technological sensibilities, in order to highlight the complex entanglement that governed this evolution of software intended (among others) for architects.

1.2. Human–computer interaction

The first reflections on the human–computer interaction were naturally integrated as a sub-category of the broader ones defined under the name *human–machine*. The cybernetic project, as stated by Norbert Wiener², made this question

2. As cybernetics specialist Ronan Le Roux points out, in his critique of the relationship between cybernetics and cognitive science (Le Roux 2015), the question of human–machine interaction was central, but was not shared by all. It seems indeed that the neurologist Warren McCulloch and the cognitive psychologist Walter Pitts, who participated in a significant way

the heart of its investigation. To be more precise, let us say that the relationships between living beings (humans and animals) and machines, and vice versa, were thought of through the notions of communication and control, which would be translated into information and regulation (Le Roux 2014). There would indeed be an ambiguity about this communication–information transfer, both in terms of their conceptual proximity, intertwined with each other, as well as in their translation into French, where *la théorie de la communication* tends to slip quickly into *une théorie de l'information* (the theory becomes a theory). In spite of these distinctions, communication and information was at the center of numerous interrogations around the machine, mainly when the latter passed from *calculation machine* to *information machine* (see Triclot 2008; Ségal 2003). In spite of the interest and relevance of these notions – information and communication – they would have had difficulty finding an audience in the 21st century. The two notions, indeed, were strangely abandoned and were only very little used by the circles of the architecture, and even more widely, to account for these digital practices.

We will approach this question of communication–information by means of the interaction, and more precisely in the modes of relations that certain individuals proposed to engage with the computer. The famous article “Man–Computer Symbiosis” of the psychologist and *computer scientist* J.C.R Licklider remains a reference for the types of relations to be considered with the computer. The main aims of this symbiosis were:

1) to let computers facilitate formulative thinking as they now facilitate the solution of formulated problems and 2) to enable men and computers to cooperate in making decisions and controlling complex situations without inflexible dependence on predetermined programs (Licklider 1960, p. 4).

Strangely enough, solving problems, even complex ones, will hardly ever represent the only task to be carried out by a computer; it will nevertheless constitute a sort of base, an indestructible but insufficient asset, which will always be a matter of pursuing and improving in one way or another. This lack of interest in formulated problems will find a real echo in design practices, because it allows us to underline the distinction between what belongs to conception and/or creativity versus what belongs to the domain of expertise. As a second point, Licklider raises the idea that decision support should be imperatively solicited in complex situations; consequently, there was no question of resorting to the computer when the situation

in the cybernetic project, were only very little concerned by this question of the interaction, or at least, it was not their essential question.

did not require it, which could lead us to think that the computer does not have the task of taking the place of a human. And, still under point two, the results obtained should not show the logic of the programs used. They had to appear random, indeterminate, unexpected. It was implied that the computer, in symbiotic mode, sought to be similar to the principle of indeterminacy proper to the living, while dissociating itself from it. For Licklider, this symbiosis was based on a particular *partnership*. It was not a question of thinking of the computer as a simple extension, or as an extension of a single organ that can be improved; nor as a completely autonomous reality, such as artificial intelligence, but rather of proposing something between the two, like the insect that allows the fig tree to reproduce, an example that Licklider uses to illustrate his idea of symbiosis. In this logic, the challenge is to share the tasks by relying mainly on the abilities of each person:

Relative to men, computing machines are very fast and very accurate, but they are constrained to perform only one or a few elementary operations at a time. Men are flexible, capable of “programming themselves contingently” on the basis of newly received information (*Ibid.*, p. 7).

Relying on the computer hardware available at the time, Licklider raised some difficulties, such as the problems of memory and language, and underlined the problem of speed lag which requires recalibration and time-sharing in order to make human thought (and not especially the computer) more efficient. However, Licklider did not see the solution to these issues in semi-automatic, or fully automated systems, which he saw as the prolongation of the old, mechanical paradigm, as a mere extension, but wished to propose another way, while being aware that symbiosis will not be the dominant and central paradigm of “complex technological systems” (*Ibid.*, p. 5).

We will dwell on this question of speed differences, which were also emphasized in a short article by Wiener (1960). In this article, as usual, Wiener touched on many subjects, such as the game and the players; machine learning; the opposition between humans and slaves; and the scale of temporalities. This question of different temporalities is the highlight of the article.

We have seen that one of the chief causes of the danger of disastrous consequences in the use of the learning machine is that man and machine operate on two distinct time scales, so that the machine is much faster than man and the two do not gear together without serious difficulties (*Ibid.*, p. 1358).

It is on this point – the speed discrepancy – that the moral question rests, for Wiener. “What are the moral problems when man as an individual operates in connection with the controlled process of a much slower time scale, such as a portion of political history or – our main subject of inquiry – the development of science?” (*Ibid.*).

In this text, the mathematician expresses the wish that scientists get more involved in the development of technologies so that these advances are not only in the hands of the state and the military. He calls on scientists to take on their responsibilities and develop an ethical thought capable of “managing” these technological implications in the long term.

Computer scientists Johnson and Kobler (1962) engage in a discussion with Wiener’s text. They build their argument precisely on this shift in temporality and speed:

Machines can be trained to learn, and computers undoubtedly show originality, particularly in game learning, not only in short-term tactics but also in long-term strategy. The machines can transcend their makers and programmers, and the end point may be creative and new, but not necessarily appreciated by the programmer (Johnson and Kobler 1962, p. 877).

This overtaking, which was possible to appreciate at the time in the game of checkers, for example, offered an interesting aspect of learning, precisely thanks to this shift. For this reason, the authors state that: “the time balance is not necessarily dangerous; its very existence is one of the benefits of automation” (*Ibid.*). Like Wiener, engineers will claim a principle of responsibility that they believe the computer required. It was the role of individuals to establish the different functions of machines, distinguishing their relevance. Indeed, the authors saw an increase “in our knowledge of human functions” in the improvement of computer techniques. This human–computer relationship thus allowed us to get to know ourselves better and, to paraphrase the subtitle of the article, understand the very human human-being. This deepening of the human condition is thanks, precisely, to this shift of speed and scale of time, and is erected in a kind of extension of the scientific instruments which allowed, by other types of change of scale, the measurement and perception of phenomena inaccessible to the senses. The philosopher–epistemologist Gaston Bachelard formulated the notion of *phenomenotechnique* to insist on the “construction” of the phenomena observed by the instruments, without which this reality remained invisible. “[The purpose of *phenomenotechnique*] is to amplify

what is revealed beyond appearance. It takes its instructions from construction” (Bachelard 1985, p. 13).

None of these authors allude to this notion of phenomenotechniques, nor to the thought of the French philosopher, but it goes without saying that this speed gap between the computer and the human individual, introduces an increased and metamorphosed perception, or at least a shifted one, which lets us understand how to shed a new light on the “humanity” of the human. If the very way of defining this “more” human deserved further developments, we nevertheless wish to underline, here, that these three inaugural texts on human–computer interaction, by raising this speed gap, suggest that this interaction-encounter with the computer would bring us closer to us, humans, and to our humanity (hence Wiener’s doubts), than the opposite.

By focusing on architectural design, we can shift this reasoning and see it as a way to deepen, thanks to these digital tools, our knowledge of the processes of design, creation, and why not architecture itself. This idea was raised by some pioneering architects, such as Nicholas Negroponte, who saw in computers a way to design *Toward a More Human Environment* (Negroponte 1970).

1.3. Technological imaginations and consciousness

The introduction of the computer into the world of architecture, and more broadly into the arts, took place at a time when many thinkers were addressing questions about technology and its purposes. Indeed, these new machines aroused the curiosity of many. As the specialist of media Marshall McLuhan said in 1964, “the technologies begin to play the role that art played and make us conscious of the psychic and social consequences of the technology” (McLuhan 1968, p. 12). From this time, it develops, weakly at first, a “technological conscience” in the sense that the philosopher Yuk Hui gives of it: “as a psychoanalysis of the technological unconsciousness, and thus an attempt to liberate the technique from its repression by this *cogito*, symbol of the modernity” (Hui 2021, p. 24). This awareness is all the more complicated to realize because the technicality proper to the digital is smothered by what we could call a “cultural overload”, which comes “to denature” the technologies, by concentrating mainly and exclusively on its utilitarian aspect, rather than on its internal functioning. While some pioneers were sensitive to this dimension, it is true that for a large portion of the architects their question, in the middle of the 1960s, consisted of wondering what these machines could be used for. As we will see, this led to a number of events, of meetings around the computer with an imaginary inherited, in great part, from the machine era which, paradoxically, was coming to an end. These meetings were the means to expose the first

experiments that were with these machines by a restricted group of pioneers. This production remained nevertheless marginal, even hermetic, to finally, towards the end of the 1960s, open and reach a wider public. This change of perspective produced a slight shift, perceptible in the passage from Hard to Soft, from Computer to Software, which enabled renewing the imaginary, by seeking to precisely differentiate itself from that linked to machines. It was successfully through this new imaginary, in construction on the ruins of the era of the triumphant machine, that data processing introduced itself in architecture circles.

In spite of the divergences which exist between the imaginary ones related to the numerical technologies and perhaps even more to the nascent data processing and the real functioning of these machines and its awareness, our conviction remains that it remains possible to approach them in a way to obtain a combination. It would be through this combination that it would become possible to account for this introduction then appropriation of the computer and its software by the architects. The aim of our study is obviously not to isolate each of its meanings, nor to try to oppose them, but rather, as Simondon implied in his approach defended in *Culture et technique* [*Culture and Technics*], that is, to propose a “method which supposes that one always keeps the ‘idea in the back of one’s head’ forcing one to go from one reasoning to the opposite one [...] thanks to a kinematic process of tilting and inversion...” (Simondon 2014b, p. 329). Our attempt will indeed be to operate this kind of back and forth in order to offer a critical and encompassing look.

1.3.1. *First meetings around the computer*

Unlike engineering, where computers were used almost routinely in the 1960s, the art, design and architecture communities were relatively distant from these new technologies and, with a few exceptions, hardly used them at all. A number of events, publications and exhibitions nevertheless allowed architects to become familiar with this costly and cumbersome equipment. Above all, these events raised many questions about the value of using computers in architecture.

One of the first important meetings around the computer was certainly the one held at the Boston Architectural Center in 1964 under the title *Architecture and the Computer*. This meeting, initially modest in intent, quickly grew to enormous proportions, attracting a crowd of 600 people, with an inaugural presentation by the then 81-year-old Walter Gropius, and talks by Marvin Minsky, Christopher Alexander and Steven A. Coons. In addition to this panel of distinguished guests, there were aeronautical simulations from the Boeing Company, kaleidoscopic recordings from Lincoln Laboratory (MIT) and computer-generated films from Bell

Laboratory, each of which created strong impressions in the audience. The event was under the following polemical question:

“What does a playboy of the Western world do with a new toy?” [...] The toy was the computer, and the playboy was the “creative” architect: could the “Mother of All Arts” really benefit from this bulky, expensive and seemingly incomprehensible machine? (Kotsioris 2015)

As the architect and historian Evangelos Kotsioris points out, the first encounter between architecture and the computer began with a contestation, more of a confusion than a frank distinction that “ranged from enthusiasm to sheer terror”. It is true that the 1960s were strongly marked by the quasi-visceral fear to see the machines revolting and reversing the relations of force, in such a way to imprison humans, as was possible to envisage in Kubrick’s film *2001, A Space Odyssey* (1968). It was a fear of this kind that many of the architects present felt, seeing technical “progress” in a negative light. The reasoning of these architects went something like this: if machines could perform their tasks more efficiently and quickly, then it was not impossible that these same machines, in the more or less long term, would come to replace them, hence their main fears.

On the other side, still according to Kotsioris, there were those who saw the computer in a better light, many of whom thought they could benefit immediately from the benefits of machines; these could free them from repetitive and alienating tasks, in line with the ideologies claimed by the leisure society that had been established since the beginning of the 20th century: the machines made it possible to perform the same tasks in less time. In this second group, however, a few individuals stood out, such as Marvin Minsky, considered by many to be one of the central actors in the development of artificial intelligence; for whom the interest was not in immediate, concrete applications, but in long-term developments. “Let’s not worry about [...] how computers are going to help us with small things. For in no more than 30 years, computers may be as intelligent, or more intelligent than people” (Minsky quoted in Kotsioris (2015)).

This event led to, by its diversity of opinion and subjects evoked, several problems coming to the fore, which were defined more clearly with the passing of years, as well as the many registers on which the computer could intervene in architecture coming to be recognized.

In the middle of the 1960s, these kinds of events and publications, attracting a large public, were relatively rare, or at least, they were not important enough to be saved and diffused beyond a relatively narrow circle. It was nevertheless possible to

see the appearance of certain collections, scientific works or exhibitions, which enabled the diffusion of these experiments and reflections in progress.

The Walker Art Center was one of the first to devote an entire issue (66/67) to “Design and the Computer” in its *Design Quarterly*. As we will see later on, several of the authors present in the issue were pioneers in these emerging fields of assisted or augmented design. It is possible to think that this issue allowed, among other means, a wider audience to be reached than the restricted circles in which this research was conducted. Let us mention here some of the authors present in the issue in order to show the ambivalence of the approaches and the subjects treated, as well as introduce several of the key actors of this period; for example, Steven A. Coons introduced *Sketchpad* as “Computer-Aided Design”. Allen Bernholtz and Edward Bierstone, in “Computer-Augmented Design”, built on the main work of Christopher Alexander (the HIDECS 3 program and its tree system exposed in *On the Synthesis of Form*) in order to realize a project according to these laws. Marvin L. Manheim, who worked with Alexander on HIDECS 2, explores different problem-solving processes for highway design. Michael Noll, a pioneer in digital art, presented his work in the field of visual arts, including the possibility of generating works in the manner of Piet Mondrian. This work was awarded a prize in the *Computer Art* competition organized by *Computers and Automation* magazine in 1965.

This magazine, founded in 1950 by Edmund C. Berkeley (the author of *Giant Brain*, in 1950), had a major influence on the introduction of computers and automated processes in the world of art, among other things that launched in 1963 including an annual competition of computer art, which realized known works, mainly visual, with a computer, or more widely with an automated machine. These competitions had a repercussion that went beyond the readership of the magazine and attracted the attention of the curator Jasia Reichardt who, in 1968, organized an exhibition of several works by the prize-winning artists. The exhibition *Cybernetic Serendipity. The Computer and the Arts* was presented at the Institute of Contemporary Arts in London, England. It was certainly one of the first events to gather and expose the potentialities of the computer in the artistic fields. In addition to the exhibition, there were a series of lectures, such as *robots serve man: comments on joy and innovation* given by Gordon Pask, and others on the impact of computers in literary studies and music. This event, strongly inspired by the notion of feedback in cybernetics, and by emerging robotics, highlighted the notion of serendipity, which can be explained as: “the art of finding things by chance while looking for others” (Le Roux 2009, p. 98), which Wiener developed as a principle of reverse invention. This could be described as follows: “Does the solution I have obtained make it possible to solve other problems?” (Wiener quoted by Le Roux, *Ibid.*).

In addition to these events, exhibitions, discussions, performances and reviews, “general public” books began to appear, which had the task of introducing computers to architects. In this type of work, there was usually a substantial part that served to present the hardware and the different peripherals, so as to make this technical vocabulary (now completely integrated) known; these works then raised some of the implications for practice and usually ended with the demonstration of the latest applications available. The book by Campion (1968) will be an example of this type of popular science book. The 1970 edition of *Le carré bleu*, entitled *Informatique et architecture*, is also representative of this type of work.

It was more or less in this period that the notion of software became widespread in the architecture and design circles, even asserting its superiority over hardware, as Campion cautiously pointed out:

Whereas the computer’s central processor unit and its associated peripheral equipment are referred to as *hardware*, the *translators* and other *systems programs* provided by the particular computer manufacturer are referred to as *software*. It is probably true to say that the software is as important as the hardware, and that any major step forward in the use of computers is likely to be in the sphere of the software rather than the hardware. Although we shall probably continue to be amazed at the way in which computers become smaller and faster, we will be primarily interested in the software that facilitates their use (Campion 1968, p. 92).

1.3.2. “Software”, the constitution of a notion

As the media theorist Friedrich Kittler provocatively asserted, “software does not exist as a machine-independent faculty” (Kittler 2005, p. 38). There is indeed a co-constitutive relation between software and hardware that will remain central in the functioning of the computer, even if this combination has become more complex with the connection to the network that puts the unity of the software into perspective³. As we have just seen, it is first of all the notion “computer” which made its entry into the circles of architecture, and not those of software nor of programming. The reception of this notion, outside the narrow circle of the pioneers

3. On this subject, Lev Manovich, like many, does not reduce the notion of software to a program that we “install” on a computer, but opens it up to its functions and operations, beyond its medium. For Manovich, software applications include both software dedicated to distribution; on access, such as web browsers; messaging providers; exchange platforms or content editing software: videos, texts, sounds. For him, they are all Media Software.

having access to this imposing equipment, found two representatives in important figures in the year 1986: Archigram and Reyner Banham.

HARD-SOFT is written in big letters, in the margin and vertically in one of the number 8 sheets of the English group of architects Archigram⁴. The opposition *hard-soft* followed their work on inflatable, transportable and nomadic architecture. “Hardware” refers to any tangible object that can be touched. “Software” is the system, message or program that can be transmitted but not touched (Archigram 1968). The *Soft* played a role, among other concepts (metamorphosis, nomadic, comfort and emancipation), in reinvigorating the architectural scene, which was made possible with the “machine-architecture rhetoric of the futurists”, which Archigram gives as a reference. “In systems planning we are reaching a point where the statement ‘the software’ is sufficient to organise the right (control of/positioning of) arrangement of an environment” (*Ibid.*). Software became one of the solutions to overthrow the existing architecture, the architecture that had run out of steam. “Hardware has limitations. Software is being pitched against it in order to expose the architect’s continued complete hang up on hardware” (*Ibid.*). Software would act as a kind of telescope that would make it possible to see the errors of the past, while allowing the city to be organized in other ways. They consolidated their remarks by ending their leaflet with an instruction, in Algol language, of random generations, which came to participate in this opposition against the “hard”, which characterizes the architecture and its materiality, as well as the way of making the city.

The same year, the theorist Reyner Banham proclaimed the triumph of software over hardware, based on his analysis of Roger Vadim’s film *Barbarella* (1968), which he considered the first post-hardware film. The critic points out, however, that many of the themes present in the film had already been raised for a few years in underground architecture circles. In the article, Banham develops the idea of a responsive environment: “in the sense of not being rigid and unyielding; articulated only by hinges between disparate rigid parts: an ambiance of curved, pliable, continuous, breathing, adaptable surfaces” (Banham 1968, p. 629). It will be this type of environment, covered in fur, that will win over the intimidating, cold, gray architecture: over the machine.

This exhaustion of the machine, its disruption, its replacement, and even its destruction, appeared more and more clearly, as it was possible to observe, still in 1968, with the exhibition held at the MoMA: *The Machine as Seen at the End of the Mechanic Age*. The works presented in the exhibition can indeed be found under the

4. Archigram were also known for their early interest in the computer and its components, producing among others *Plug in City* (Peter Cook, 1964), or *Computer City* (Denis Crompton, 1964).

cover of the mechanical age, that is, in the prolongation of Da Vinci's machines, of Duchamp's devices, such as the large glass, of Laszlo Moholy-Nagy's installations or even, of the folded bodies of the sculptor César.

It was with the exhibition *Software*, presented at the Jewish Museum in New York in 1970, and sponsored by the American Motor Company, that another step was taken. It is enough to put this exhibition in parallel with the one held at the same time at the MoMA, titled *Information*, and which took up the principles of media theory, to understand the specificity of *Software*. In the wake of the *Cybernetic Serendipity* exhibition, *Software* approached it while differing from it on a few points. Without completely denying the reference to cybernetics, this one "is a little too general and *passé*⁵ to computer specialists" (Burnham 1970, p.11). What was put forward, even before the artists' works, was the hardware itself. "Hence the goal of *Software* is to focus on sensibilities in the fastest growing area in this culture: information processing systems and their devices" (*Ibid.*, p. 10). Starting from the observation that these technologies of control and communication invaded society more and more, in a general way, it was a question of showing the effects of these technologies "in the hands of the artists". The exhibited works consequently came to be unequal in their aesthetic qualities, and led to confusion and irritation on the part of many visitors and art critics, as it was possible to note in the virulent criticism of Grace Glueck for *The New York Times*.

The curator of the exhibition, Jack Burnham, fed into this confusion by attributing several possible meanings to the notion of *Software*, covering a wide, heterogeneous and sometimes even contradictory field. In all cases, "the machine in *Software* should not be regarded as an art object". Rather, the desire was to focus attention on the dialogue that is instituted between the work and the public: "We are trying to make that sense of dialogue a conscious event" (*Ibid.*, p. 12). This inevitably raised the question "is it still art?" The actors of the exhibition – artists and curators – did not answer this question deliberately, it was instead put to the visitors. It was up to them to sort it out. The boundaries of art were shaken, both by these objects and by the role to be played by the visitors–spectators–interpreters.

In this exhibition, the SEEK project by architect Nicholas Negroponte was the centerpiece. The project was on the cover of the catalog (Figure 1.1) and created an "infinite fascination" (Glueck 1970) for many visitors.

5. In French in the text.

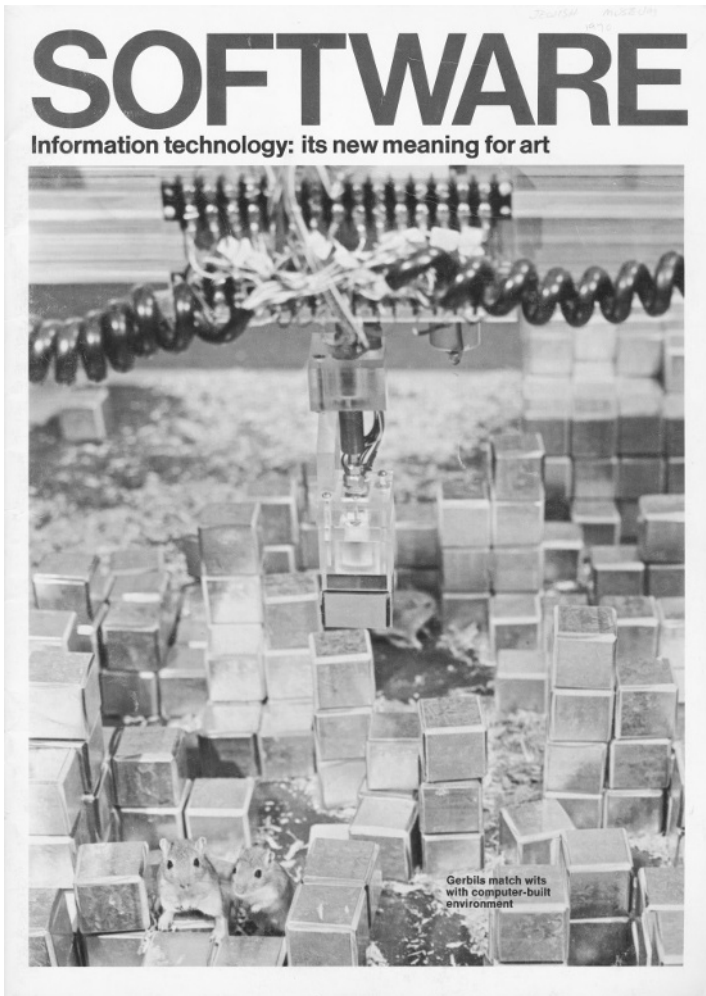


Figure 1.1. Cover of the exhibition catalog *Software. Information technology: its new meaning for art* (1970), with the *SEEK* project by architect Nicholas Negroponte © The Jewish Museum, New York, Dist. RMN-Grand Palais

Negroponte is certainly one of the central figures in the introduction of the computer into architecture, and we will return to this below, but for the moment we would like to focus our attention on the introduction of the notion of *Soft* into the world of architecture, as well as on the shift away from *Hard* – from the machine. Nicholas Negroponte, an architect, founded the *Architecture Machine Group* (AMG) with Leon Groisser at MIT in 1967. The *Architecture Machine* was at the

same time a publication, a research group and the machine on which they carried out certain experiments. We will look at the evolution that took place between two of their publications, spaced only a few years apart. The first was *The Architecture Machine* (singular), published in 1970:

In this book, there is no distinction between hardware and software, between special-purpose computers and general-purpose computers. [...] Our interest is simply to preface and to encourage a machine intelligence that simulates a design for the good life and will allow for a full set of self-improving methods. We are talking about a symbiosis that is a cohabitation of two intelligent species (Negroponte 1970, p. 7).

This work, especially if we compare it with the second one, is limited to the machine, to the technical equipment, and above all, it is mainly interested in the use of the machine for the design processes, which it considers as a kind of partnership à la Wiener or Licklider, improving them, or at least, transforming them. The UBRAN 5 program (see below), presented in the last part of the book, illustrates the aim and purpose of the book.

The interest, for us, lies in the differences that appeared in the second work that Negroponte published, under the title *Soft Architecture Machines* (plural this time⁶). In this second opus, he continued his investigation-reflection in a way, radicalizing it. The interest is not only concentrated on the conception, nor the machine, but opens up to the issue of the intelligence, with the arrival of intellectual reinforcements like the recourse to the works of Marvin Minsky and Seymour Papert. He also widened the role to be attributed to the computer in architecture. “I consider three potentials of the computer: (1) the computer as a designer, (2) the computer as a partner to the novice with a self-interest, and (3) the computer as a physical environment that knows me” (Negroponte 1975, p. i). These three futures represent the three parts of the book, which he produced in collaboration with some of the important protagonists of the time. Negroponte thus established a dialogue with Steven Coons on “seeing and making images”; with Yona Friedman on “participatory design”; and with Sean Wellesley-Miller on “responsive architecture”. These three futures will also be the object of a gradation-evolution, which define the role of the machine: first, it *assists* the designer, then it *augments* them, to finally *replace* them.

6. For various reasons that Negroponte explains, *Soft Architecture Machines* was not published until 1975, although it was ready in 1972, two years after *The Architecture Machine*.

NOTE.— It is worth noting that in this logic of meticulously listing the appearance of the notion of *Soft*⁷, in the world of architecture, it is necessary to mention the short work of the architect Emmerich (1974). If we return to this idea of self-construction, let us say that the use of the term soft in the book is reduced to a single chapter entitled “soft architecture”. In this chapter, the central question concerns the relationship between the built and the society, with the observation that the dwellings are terrible and *antisocial*. It is through this criticism that the architect resorts to the soft of the software, which they use in a way similar to Archigram and Reynier Banham, but even more radical:

Prison from which one escapes with pleasure, the housing and the cities built for the second category of man [that which does not build itself their dwelling], because they are served, that is to say the opposite of this tender, tolerant and intelligent architecture in the sense of soft-ware compared to the hard-ware of the automatisms. This is a hard, durable and definitive architecture, sometimes entirely composed of glittering hardware, which only delinquent action can alter somewhat but not change (Emmerich 1974, p. 60).

The singularity of Emmerich’s thought concerned the idea of self-construction, which has, a priori, little to do with the idea of soft. Nevertheless, another event allows us to put forward a hypothesis in this connection: the working sessions held in 1972 under the theme of the “Responsive House”, in which Negroponte participated with some considerations on responsive architecture, which today are integrated into the Smart logic. Not surprisingly, Negroponte’s intervention remained marginal, still in its infancy, with the idea that the computer itself becomes an environment. In fact, for the majority of the contributions, under the tutelary figure of Thoreau, the question of responsive reactivity mainly deals with the possibility for the occupants to design their habitat themselves. The new techniques would be a way next to prefabrication and Handicraft.

Emmerich, with his *Soft architecture*, continues, in a way, this double thread of the soft, which was to propose an alternative to the rigidity of the architecture in

7. To be even more precise, we should add William S. Burroughs’ unclassifiable *Soft Machine*, written and rewritten between 1961 and 1968. Despite the difficulty of reading, due to the cut-up process implemented in this book, it is not uninteresting to underline that certain passages give us a glimpse of the role of these machines. “In other words, the IBM machine controls thought, feeling and apparent sensory impressions [...] These agents do not even know what buttons to push – whatever you feed into the machine on a sublime level the machine will process” (Burroughs 1968, p. 182).

place, built after the war, as much as the possibility of supporting the techniques of self-construction: “It is the space to each one according to its needs” (*Ibid.*, p. 65).

The presentation of these few events and publications was intended to show that the use of the notion of *Software* in architecture appeared at a relatively precise moment. At the same time as it is in line with the emerging debates on the computer in architecture, it marks a clear distinction, by its frank opposition with the notion of *hard*. In spite of the equivocation, sometimes voluntary, the *soft*, in its opposition to the *hard*, envisaged, from the 1970s, a destabilizing capacity of the computer for the architectural practices.

1.4. The pioneers

We will present here some of the pioneers who were mainly interested in the computer for the fields of conception in the creative circles of architecture and design. Our aim is first of all to describe these first experiments by focusing our attention on the main real or envisaged intentions for these programs, as well as to raise the differences between the postures that were established with regard to the computer. As we will see, there was a relatively frank split, from the beginning, or at least, two divergent points of view were perceptible as for the way of considering the machine–computer in the processes of conception among the main personalities implied in these first developments.

These two points of view have also led in their wake to a number of oppositions and tensions between the actors who would be at the forefront of this appropriation of digital technologies in architecture. It is rather complicated to define this division frankly; we will try the opposition between information-intelligence and communication-interaction. To read *Architectures of information*⁸ by Molly Wright Steenson, a distinction would need to be made between the architects who use the computer as a simple tool, a medium and those who use the computer from a different perspective. For the former, these would use the computer “as a conceptual device to realize the image and control all of its parts” (Wright Steenson 2014, p. 2). In this scenario, the image acts as a vehicle for exploration; where the designer finds a medium in the image for their thinking, which progresses. This technique is similar to the older one of *disegno* as a “vehicle of thought and creation” (Ciaravino 2004, p. 8), for the reason that the *disegno* could be thought of: “as a place of production of the visual form of an idea” (*Ibid.*, p. 23). In this way, it is by communicating and interacting with a form, by manipulating-controlling it via a

8. *Architectures of Information*, was Wright Steenson’s thesis, defended in 2014. She wrote a book, which reviewed and adapted her thesis, entitled *Architectural Intelligence*.

computer, that the object of the conception, the project, takes shape. “The glance which runs along the line is a glance which seeks” (*Ibid.*, p. 26).

The subject of Wright Steenson’s book focuses instead on the second group, composed of architects such as Christopher Alexander, Cedric Price and Nicholas Negroponte. The latter “formatted design as an information problem, visualized informational paradigms in architecture and built generative systems that incorporated models of artificial intelligence” (Wright Steenson 2014, p. 2). This attention to constitutive information, or even process intelligence, implies, therefore, working beyond representation, where design work focuses on defining the processes of future forms, rather than the forms themselves.

While Wright Steenson’s work aimed to rightly offer a counterweight to *Archaeology of the Digital* (2013), which was being developed at the same time under the tutelage of Greg Lynn, at the Canadian Centre for Architecture (CCA), he favored only one of the two groups. It remains nevertheless, from our perspective, important that these two positions be raised together, in their confrontations and sometimes even, in their converging points. Our position will not be to hide this opposition, but not to consider it primarily, in order to show that several practices occupy an ambiguous, and sometimes changing, position in relation to these established categories. This opposition between information and interaction nevertheless allows us to raise some of the shifts that have taken place from generation to generation of architects, sometimes favoring one posture, sometimes the other, and we are interested in understanding why?

1.4.1. The CAD (Computer-Aided Design) project

In December 1959, the Computer-aided Design research project was launched, bringing together various engineering departments, several of which were linked to MIT and the US Air Force. Although the term CAD was introduced into architectural circles at the turn of the 1970s, the term was still seeking its own definition at the time of this inaugural project, as well as its field of intervention. As Douglas T. Ross, one of the co-directors of the project, affiliated with the ESL (MIT Electronics Systems Laboratory⁹), said: “The objective of the Computer-Aided Design Project is

9. When the project was launched, the laboratory was already 10 years old, even though it was originally called the Servomechanism Laboratory. On this subject, see the book by Cardoso Llach (2015, pp. 34–48). It very clearly shows the evolution over these 10 years by insisting, among other things, on the strong involvement of the US Air Force in the various projects of the laboratory, as well as on the precedence of CAM (Computer-aided-Manufacturing) technologies on those of CAD.

to evolve a man-machine system which will permit the human designer and the computer to work together on creative design problems” (Ross 1960, p. iii).

In a relatively consensual way with the acceptance of the human-machine system of this time, Ross posed one of the essential milestones of CAD, the “working together”, which he specifies a few lines later:

Computer-aided design is *not*¹⁰ automatic design, although it must include many automatic design features. [...] Computer-aided design should *not* just be a mirror of present design practice, although it should of course be related to it (*Ibid.*, pp. 2–3).

This objection to automating design, placed at the beginning of the CAD project, and announced as an obvious fact, remained a key idea that was regularly taken up and evoked, as if the temptation to automate everything were too strong and, consequently, it was necessary to reassure and insist on the fact that the designer will not be eliminated from the process by entirely automated processes. There was also this redefinition of design, which also appeared as a recurring issue. This was in fact emphasized by Lev Manovich in his study of software and the idea of remediation; that is, “the representation of one medium in another” (Manovich 2013, p. 59). If the creator of a piece of software reproduces an existing technique, they never wished to reproduce it identically, but always had the ambition to add something more to it.

Although the CAD project did not, strictly speaking, result in a computer program, it was the initiator of many of the components of computer-aided design¹¹. We will see in detail some of the principles of the CAD project stated by Ross in his technical note, but it is necessary before going further, to specify that another co-director was strongly involved in the development of this project: Steven A. Coons. While Ross had the responsibility of the “computer program”, Coons (and Robert W. Mann) had that of graphic design. According to various researchers (Cardoso Llach 2015; Mendes Correia et al. 2018), these co-directors not only had different tasks, but divergent visions of how to employ the machine for design:

Ross envisioned an automatic and programmable design machine making non-interactive computer design while Coons foresaw the computer as interactive as it could be with a man using the screen to

10. Author’s use of emphasis.

11. As Cardoso Llach shows, the CAD project followed various research such as that on the automation of milling machines (the ATP project from 1955 to 1959); and in 1963, the MAC project encompassed the CAD project and that of A.I. The MAC project continued until 1975.

design. [...] The idea of “creative design problems” is very important to demonstrate the fact that Coons envisioned an interactive design system for creative designers, like artists and architects that do not have the idea of what they want to draw right from the start (Mendes Correia et al. 2018, p. 2).

It is moreover because of this difference that we insisted on the distinction between information and interaction, even if we necessarily imply the other. The interest for Ross was not a priori to interact or communicate with the machine, but to provide it with the necessary and relevant information so that it, after manipulation-compilation, gives a satisfactory result. For Coons, it was the opposite. We will look at the positions defended in more detail.

Ross saw the issue of design as nothing more and nothing less, with some distinctions, than a problem-solving activity. Once that was said, and without contradiction, his goal was not that the machine could solve problems, but rather that it would help in the formulation of them. “From the computer applications point of view the primary problem is not how to *solve* problems, but how to *state*¹² them” (Ross 1960, p. iii). In fact, Ross’s idea was to say that if we know the problem, then there is no “real” problem, but only a difficulty in execution. To this statement, he added a distinction between problems defined from the outside in, and those from the inside out. The latter, for Ross, were simpler to solve, while the formulation of problems coming from the outside (to the object) towards the inside were, as for them, much more complex to solve, and besides, it is not obvious that they could ever be “truly” solved. In other words, the problem that interested Ross was: how does the environment (taken in the broad sense) or the general intentions of the project (what is external) shape the objects? And finally, but not unimportantly, the task of the program designer was to define a language, from existing languages and their possible evolutions, in order to make a declarative system specific to the *objects* operational.

On the other hand, Coons would not approach the problem in the same way. Coons placed Interactive Computer Graphics at the center of his concerns. It is possible, in this respect, to consider Ivan Sutherland’s *Sketchpad* as an illustration of the main ideas expressed by Coons. It is not surprising that Coons was strongly involved in the *Sketchpad* program. He made a detailed presentation of it in the issue of the Walker Art Center under the title, precisely, of “Computer Aided Design”. He was also a member of Sutherland’s thesis jury and participated in a more or less important way in his programming. He was also the one to introduce Timothy Johnson’s demonstration of the *Sketchpad* during a television show.

12. Author’s use of emphasis.

1.4.2. *Communicating with the machine, via drawing*

Under the direction of Claude Shannon, Ivan Sutherland created the *Sketchpad* system, which aimed to establish a communication with the machine that would no longer go through written wording, or through punched cards, which at the time was the main, if not the only, way of communicating with this kind of machine. “The Sketchpad system makes it possible for a man and a computer to converse rapidly through the medium of line drawings” (Sutherland 1963, p. 17).

To carry out this project, Sutherland used some of the technologies available at the time, mainly those developed at the MIT Lincoln Electronic Laboratory, such as the TX-2, which consisted of a 7-inch (about 18 cm) visual display, a push button control and a file cabinet, activated by large toggle switches. In addition, there was a light-pen which was used to “draw” lines on the screen. By activating the *start drawing* switch, it became possible to define different configurations: line segments, circular arcs, etc. It was then possible to manipulate these lines thanks to the command box that made it possible, for example, to erase a line or modify its length.

The purpose of *Sketchpad* was not only to reproduce the techniques of drawing, but to implement various relatively simple, but innovative, devices that could augment the techniques of drawing. One of the main features was the ability to define archetypes that Sutherland called “subpictures”. These subpictures made it possible to keep an image in memory, as a model, and to re-use it as we wish. It was also possible to modify this subpicture and see, instantly, the instances of this model being modified automatically. Another feature that was appreciated at the time was the constraints that could be applied to one or more features in order to associate them with each other. It was possible, for example, to make two lines parallel, perpendicular, or to hang a vertex in the middle of the line, connect corners, etc.

Personalities around the computer industry such as J.C.R. Licklider, Alan Kay or Ted Nelson were quickly impressed by this program, because “it enabled envisaging a new way of working and seeing” (Nelson quoted by Rheingold (1993, p. 92)). Indeed, *Sketchpad* allowed operations that were previously impossible on a static physical medium, such as zooming in and out, or even the possibility that the drawn object could disappear from the screen, while remaining present in the computer’s memory. However, the possibilities were still limited and *Sketchpad* was relatively difficult to handle, as Coons noted in his 1966 article.

In this article, Coons already considered *Sketchpad* to be “old” and belonging to a bygone era. He presented the developments that Timothy Johnson would make for *Sketchpad III*, moving from 2D orthogonal projection to 3D representation. Built on

the basis of Sutherland's *Sketchpad*, Johnson proposed a screen divided into four quadrants, presenting the plan, a front view, a side view and a representation of any object in perspective. The specificity was that: "If the plan and the two elevations are drawn, the perspective view of the object will 'automatically' appear in its quadrant" (Coons 1966, p. 11). For Coons, *Sketchpad* made his idea of CAD a reality and for this reason he was critical, pointing out its rigidity. "*Sketchpad* was purely graphical and purely geometrical. One could draw with the light pen on the screen – straight lines, circles and other surfaces. It could not solve any problems that had to do with abstraction, other than the abstractions of geometry itself" (*Ibid.*, p. 8). This reproach by Coons reveals his position in relation to drawing, as well as the place of drawing in design. Indeed, Coons defended the idea that the computer could integrate, by the drawing, not only constraints related to the geometry, but broader constraints that intervened in the process of design:

Many architects, [...] use the word "design" to mean only the innovative, generative, intuitive acts of conception; to them, the necessities of structural analysis, [...] and other analytical methods for coping with the stringencies of nature and nature law, while recognized as essential processes, are not considered a proper part of design, but only of engineering and construction; subservient functions that must be employed to bring a concept realization... The true and complete process of design, it seems to me, consists of an inextricable mixture of these intuitive, imaginative cognitive processes together with analytical, mathematical, rational processes (Coons 1964, p. 26).

It is interesting to note that this "inextricable mix" of intuitive and analytical knowledge remains a central concern for many contemporary architects, and always requires constant recalibration depending on the technology employed and the problem to be solved. More mundanely, Coons' criticisms certainly stemmed in large part from the fact that at the same time he was working on the MAC project at MIT, where he was developing a system for the design of hypersurfaces used in the design of airplanes, cars and ships (Coons 1967).

Note that it may be said in closing that despite Coons' criticism, one of the objectives of *Sketchpad* was to support a certain formal complexity. Indeed, the technical report that Sutherland produced in 1963 does not begin with a technical description of his system, but is introduced by a hexagonal pattern, composed of several hexagons (Figure 1.2). While this type of pattern leads to both his system being presented and his ability to manage a certain graphical complexity being shown, it is interesting to note that it finds a real echo with the topological

explorations that were taking place at the same time. We can mention here, for example, the famous geodesic domes of Buckminster Fuller, but more broadly, these explorations were based on morphological research in crystal biology, which interested artists, architects and engineers of the caliber of Fuller or Alfred Neumann (Neumann 2020). These architects had recourse to these regular and symmetrical polygons precisely because of their capacity for space packing. With respect to this context, it becomes possible to interpret the hexagonal motif used to introduce *Sketchpad*, as a way to integrate, or at least to try to establish a dialogue with the explorations that were taking place at the same time on topological architecture.

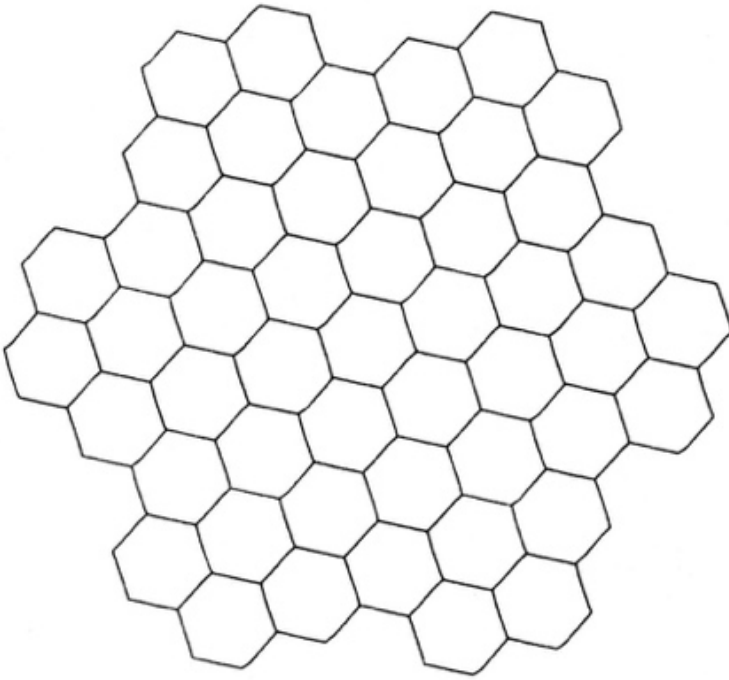


Figure 1.2. *The first example presented by Sutherland: hexagonal patterns. This example serves as an introduction to Sketchpad, even before the technical explanations (Sutherland 1963, p. 10)*

1.4.3. Representing the information

In 1964, the architect Christopher Alexander published an essay that had a considerable role in the appropriation of computer science by several architects in the early 1970s. It was to be a real landmark for rethinking design processes in a

structured and rational way (see section 1.5). The objective of the essay was mainly to “simplify design problems” by proposing a: “way of representing design problems which does make them easier to solve. It is a way of reducing the gap between the designer’s small capacity and the great size of his task” (Alexander 1964, p. 5).

Alexander proposed a system that made it possible to represent and organize design problems according to a tree-like hierarchy that broke down the “program” of the object, building or city to be designed into sets and subsets. The particularity of this method was based on a 0-1 scoring system, attributed to the elements of answers in relation to the different decomposed functions, and where the objective was to obtain the lowest possible result. In fact, Alexander’s system operated from a negative point of view: “good fit [is] the absence of certain negative qualities” (*Ibid.*, p. 19). For him, it was easier to agree, with certainty, on an unpleasant situation than on a pleasant or successful one.

In his book, Alexander uses two main examples to illustrate his tree principle: the design of a kettle, as well as that of an Indian village, designed from patterns. However, Alexander realized how difficult it was to solve a slightly more complex problem with his tree, mainly because of the hierarchy that this system introduced. This observation became clear when, at about the same time, he carried out a concrete experiment through the HIDECS 2¹³ program, carried out in collaboration with Marvin L. Manheim, which consisted of designing a highway interchange. Indeed, this tree principle did not allow for many feedback loops, as each node was composed of only two inputs, making this type of enunciation restrictive and rigid. These limitations led Alexander to improve his system, which he did with HIDECS 3, by creating four subroutines that allowed for round-trips between different criteria, “overlaps and interactions”. It was no longer a question of modeling on the tree, but of proposing *semilattices*, which allowed the whole to be recomposed. The essay *City is not a tree*, published the following year, showed the evolution of Alexander’s thinking by correcting or even abandoning his previous model. Nevertheless, Alexander remained convinced, as did many others, that these design processes could be modeled, and developed an even more complex network a few years later.

Allen Bernholtz (Harvard) and Edward Bierstone (Toronto) tested these methods, and especially HIDECS 3, to design a single-family house. Even if HIDECS 3, with its four programs, brought some improvements to the system, the

13. HIDECS 2: a computer program for the hierarchical decomposition of a set to which a line graph is associated.

authors show that the tree-based, hierarchical system still had some advantages, as it sort of reproduced the hierarchies intrinsic to the design itself:

These programs [the 4 in HIDECS 3], however, introduced a new problem: decomposition in a single step eliminates hierarchical ordering, which is necessary in the design process because it allows for a design program starting with finding solutions for subsets beginning on the lowest level (Bernholtz and Bierstone 1966, p. 42).

They tried to overcome this deficiency by first introducing groupings of subsets (Figure 1.3). After having defined 72 misfit factors, they proposed 17 groupings of these factors which were realized via punched cards. In this logic, the machine (an IBM 7094 from the University of Toronto) did not draw the house, nor even the plans, but proposed groupings of factors that allowed undesirable or unpleasant situations to be avoided. As many architects and users of these methods realized, these processes mainly led them to question their own understanding of their design process, as the authors note at the end of the article:

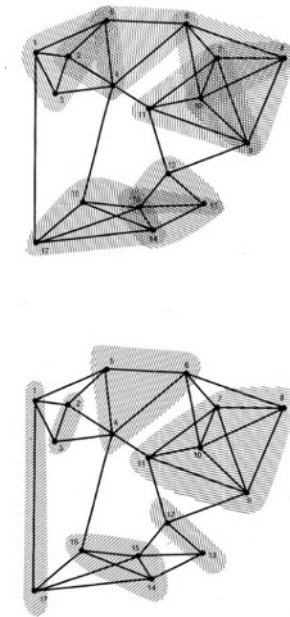


Figure 1.3. *Different formalizations of the problems according to the method developed by Alexander. They not only show the change in representation, but also the different ways of associating factors with each other (Bernholtz and Bierstone 1966, pp. 41–42)*

But the procedure itself possesses no magical power. Its effectiveness depends on a thorough understanding of its use in the context of the entire design process. The designer must be able to provide a comprehensive set of misfit factors and their links, and to solve the hierarchy of subproblems in the specified order (*Ibid.*, p. 51).

Whether it was the use of HIDECS, or what was put forward in Alexander (1964), the central question was indeed that of enunciation, that is, to deconstruct a general design problem into a number of sub-problems in a graphical form, more or less complex (from the tree to the network through the semilattices). The more exhaustive this enunciation work would be, the more it would become possible to understand the problem in all its subtleties and thus ensure a better result. This type of system was based on the following idea: “a problem clearly stated is half solved” (*Ibid.*, p. 51).

1.4.4. The computer: an entity in its own right

At the end of the 1960s, Nicholas Negroponte and Leon Groisser presented a general reflection on the place that the computer should occupy in architecture. They tried to engage in a constructive dialogue with the machine, considering it as an entity in its own right, intelligent and which, in the long run, would be a machine in which we could live. It is through this issue of intelligence that Negroponte sought to distinguish himself from interactive experiments such as those done with the *Sketchpad*. While Negroponte regretted the use of the notion of *Sketch* to name a program that was far from allowing the creative impulse of the sketch, nor its flexibility, the reproach was more percussive as to the use that Sutherland made of the machine. In this respect, he proposed the distinction between Computer-Aided and Computerized, which can be described as an opposition between a *computer-assisted* design and a simply *computerized* design. Negroponte found it reductive to consider “Aided” only in terms of the instantaneous interaction with a screen. This did not prove that the machine was intelligent or that there was a real assistant or help for the design.

For Negroponte, there were two elements that allowed for a certain degree of intelligence in a machine: on the one hand, its context dependency, in other words its capacity to grasp the context in which it found itself, and act accordingly, and, on the other hand, its capacity to understand missing information and innuendo, as two old acquaintances would do during a discussion. Beyond the technical problems linked to the limited capacities of machines at the time, Negroponte was convinced

that the obstacle was rather linked to culture and mentalities. Part of the problem, he said, was that “machines are commonly assumed to be ‘doers’ rather than ‘understanders’” (Negroponte and Groisser 1970, p. 41). For them, one of the ways to make the computer understand was to place semantics (meaning) before syntax (action), and not the other way around. They suggested that the interfaces should be split: it would be necessary to add “the non-material interface of the human speech, or at least that of the natural language” (*Ibid.*, p. 149) to the material interface, screens or other peripherals.

These differentiations inevitably had significant repercussions as to the effort to be used for certain specific points, as well as the place to be given to these machines. Negroponte developed this idea, among others, in two projects: the URBAN 5 program (1968), as well as the SEEK experiment (1969), which, as we have seen, underlines the evolution of Negroponte’s thinking.

URBAN 5 was a program designed to help the architect remember everything in the complex design of a piece of city. “The principle of this research is the creation, through the computer, of an omniscient and omnipresent partner capable of critically reflecting the decisions and criteria of the user” (Courtieux 1971, p. 48). By formulating design criteria – for example, each room must contain at least one opening, a family home must be within 500 m of a park, etc. – the machine acts as an infallible memory aid and thus becomes capable of guiding the designer’s decisions. With each manipulation, the machine evaluates the model to validate its consistency. It enables raising incompatibilities and conflicts. Incompatibilities respond to a contradiction with the fundamental elements of the design, for example, the machine signals if a part is floating in a vacuum, etc. Conflict, on the other hand, responds instead to inconsistencies with the design criteria formulated by the architect, so as to point out contradictions “between what the architect has said and what they have done” (Negroponte 1970, p. 83).

The program thus acquired a critical and reflexive character in the sense that it made it possible to become aware of the validity, or otherwise, of the criteria formulated. Negroponte had been sensitive to Stanford Anderson’s speech in London, in which he tried to propose an alternative to problem-solving, with the idea that we should rather be problem-worrying (Anderson 1966). In this intervention, Anderson sought to define the specificity of architectural design in relation to engineering design, which was developing at the same time, based on the “creative” tactics proposed in the games. For him, it was essential to insist on the

fact that in architecture, the problems were not given at the beginning and that they evolved as the design process progressed:

He [the architect] sees that any solution, any form, has implications beyond those that were intended, *including implications for the reformulation of the original problem or need*¹⁴. Consequently, he is as interested in the form as in the problem; he sees this dynamic interrelation of form and problem as of the first importance. And it is this reciprocity of form and problem that is not sufficiently recognized by the problem-solving designer (*Ibid.*, p. 8).

Negroponte would not say otherwise when he stated the necessary back and forth between form and criteria, between architects and machine (Negroponte 1970, p. 87). It is for this reason that URBAN 5 was not so much intended to address real design problems, but rather was a means by which to learn from experience. It is in this logic that Negroponte asserted that “URBAN 5 is not a tool, it is a toy” (*Ibid.*, p. 95).

Note that as its name indicates, URBAN 5 was first and foremost an aid to the design of cities or parts of cities. This detail – to favor the urban scale – is not insignificant. Trained as an architect, Negroponte had been surprised early on by the scale with which most architects designed. For him, architecture was a room in the city. URBAN 5 attempted to reconcile these two scales by placing 10-foot cube boxes in an urban configuration. This box-cube design was a constant for the AMG and was repeated for the SEEK project.

In a scientific approach, it is not abnormal to underline the problems encountered, nor to draw attention to the errors and weaknesses of the methods used, as they allow them to be improved the next time. Indeed, Negroponte raised four points on which URBAN 5 was insufficient, or had failed. Without going into detail, let us just say that the first three points were related to its rigidity, linked to its prefigurations which did not allow for the context in which the object was inserted to be taken into account, in addition to the few criteria formulated. The fourth point was that the program had to be operated mainly by architects and it was very difficult for the future residents, non-specialists, to use it without prior training. It should be noted that one of the intentions of the AMG was to use the machine to eliminate the middleman: in this case, the architect. In the line with the participatory design that Negroponte had been able to observe with Yona Friedman, he had the conviction that if a system was really intelligent, it would have the capacity to communicate directly with its users, without an intermediary, bringing itself the

14. Author's emphasis.

necessary knowledge. Under this logic, the inhabitant takes control, without the interference of the middleman, of the design of the building in which they would live. “In short, each person becomes their own architect” (Negroponte 1975, p. 101). This, in the medium term, would have implications for ways of building and thinking about the city. For Negroponte, the machine was not only intended to provide clients with a result, but its primary purpose was to provide tools that would amplify and clarify the processes of learning and thinking for ourselves (*Ibid.*, p. 108).

The SEEK project sought to address these shortcomings, and more importantly, as we have seen, was part of an expanding body of thought in which Negroponte was no longer just interested in the design process, but was opening up to the problem of intelligence, in which the machine itself became architecture. SEEK was an interactive model that, driven by a learning system, represented a city on a reduced scale. Composed of cubes of 2 x 2 inches and arranged according to an initial composition, the idea was to introduce a population of gerbils which, in their movements, moved the cubes in a more or less random way. The role of the machine was first to detect the displacement of the cubes and then to return them to their initial position. If, on the other hand, a cube was moved several times, then the machine considered the new location as more relevant and thus modified the overall composition. The latter then becomes dynamic, seeking to adapt continuously to the changing needs of the user-inhabitants. Here again, certain limits can be raised, including those of attributing any intelligence to the machine, even according to the criteria established by Negroponte. First, there was no real change in the context, only one criterion, the location of the cubes. Therefore, unlike Urban 5, SEEK was not reflexive about design processes, but about modes of “using” the space. And this “mirroring” was all the more inoperative because it was impossible to get feedback from the gerbils as to whether or not the environment met their needs.

Even if Negroponte pursued this reactive architecture path on a few occasions, such as during the *Responsive House* colloquium in 1972 (see above), and despite his marginality, it was nevertheless possible to observe similar proposals during the same period, with, for example, some productions by the architect Cedric Price. It is interesting to follow the chronology closely, because the Fun Palace (1964) had in a way initiated this principle of changing architecture, without, however, proposing a technical demonstration of it, as the AMG did with SEEK. Price only became interested in this dimension, with the technical support of John and Julia Frazer for the *Generator*, at the end of the 1970s. These three projects participated in this idea of “responsive” architecture, where the computer enabled considering a dynamic architecture that would adapt to the needs of its occupants through a number of

sensors and automated devices¹⁵. In spite of everything, this type of architecture did not really experience a continuation before the beginning of the 2000s, with, however, several bifurcations compared to the initial intentions. This idea of interactive architecture, as proposed by Negroponte, was put aside for the limits we have just seen, but especially for the reason that a more burning question captured all the attention: that of the design.

1.5. The complex problem of design

It must be said from the outset that these questions about design processes and their representation emerged in several fields at once, in different forms, and were not specific to architecture, nor to architects working with computers, even if, as we have just seen, several questions emerged from these practices. It should be remembered that, with the exception of a few pioneers, the computer in architecture until the end of the 1960s was used mainly for management tasks, to produce contractual documents or to carry out administrative tasks (Campion 1968, pp. 2–9), and that it was used very little for the actual activity of design. We have also seen that technological advances were not limited to the design activity, or even to the computer. It is enough to think of the manufacturing machines that “preceded” the computer at MIT in 1949–1959, the different robots like the one developed for the Stanford Hand-Eye Project (1969), or the reactive architecture of Price or Negroponte, or even the virtual reality helmet proposed by Ivan Sutherland in 1968; to note that the design activity represented, at the end of the 1960s, only one possibility, among others, as for the use of the computer in architecture. Our hypothesis is to say that there was a fertile connection between the line of questioning around the computer and the numerous line of questioning on the activity of conception.

Initiated among others by the “Conference on Design Methods” in the early 1960s, this research was undertaken in order to explain and clarify the design processes in the most rational way possible. The work of Herbert Simon and Allen Newell had a special place in these investigations. Simon’s work, *The Sciences of the Artificial* (1968), proposed a “science of design” that he inserted into an expanded thought of the artifact, while expressing surprise at the lack of reflection on the design of objects and the lack of interest in the points of view of the designer-inventor.

15. It should be noted that it was around the same time, in the mid-1970s, that the first home automation systems appeared.

Many of Simon's research projects were aimed at defining an effective decision support system for problem-solving, especially for fuzzy and often ill-defined problems. One of the key ideas is: "Problem solving by changing their representation" (Simon 2004, p. 236). The method of shifting the point of view in order to approach a problem differently, to facilitate its resolution, was adopted by several architects. The tree principle or Alexander's semilattices can illustrate this tendency.

At about the same time that this examination of the design processes was taking place, the accessibility of the computer was growing, despite its still large dimensions (about 2 m² on the ground). In France, the association between computer and conception occurred at the beginning of the 1970s, as it is possible to observe clearly in the introductory speech by the architect J.-P. Maroy at the events organized by the IRIA (French Institute for Research in Information and Automation) in 1971, entitled precisely *Informatique et conception en architecture*:

It seemed to us that for this first meeting on computer science in architecture, it was important to look first at design and to show that the problem was not simply the automation of an existing process but the more precise and objective definition of that process and thereby, at times, its questioning (Maroy 1971, p. 5).

With a 10-year gap, we find the same two premises stated for the CAD project, namely that there was no question of automating design and that it had to be questioned. This questioning can be found in Alexander's book, *Notes on the Synthesis of Form*, a real guide to thinking differently about design processes, by a change of representation. The book was translated in 1971 and was widely distributed in research circles, with introductions to "Alexander's methodology" (by J.-P. Maroy in the special issue on computer science of *Techniques et Architecture*), despite the fact that Alexander himself had already evolved his thinking by renouncing several essential elements, including the tree. Nevertheless, it was the introductory chapter on the "need for rationality" that was mainly mobilized, suggesting that the computer advocated analytical, scientific thinking, whereas until then, common sense suggested that design activity was rather governed by intuitive processes that were difficult to explain and, consequently, not calculable. To express this state of mind, shared by many, a distinction was proposed by John Christopher Jones between the *designer as a magician* with his irrational character and the *designer as a computer*, rational or systematic¹⁶. By a sort of analogical process, several architects have tried to integrate

16. This distinction was proposed by Jones (1969) and seems to be a faithful representation of the different attitudes that architects of the time could adopt. To be precise, this distinction rested on three figures and not only two. The third figure is the designer as a self-organized system.

this rational aspect into their practice. To rationalize, mainly meant “to make explicit procedures that were used intuitively” (Berthelot 1974, p. 27).

This work of explicitness was undertaken in two directions, simultaneously and which could be linked. On the one hand, the architects sought to modify their way of thinking about the project, in line with what Alexander advocated, and, on the other hand, certain functions linked to the programs required a procedural description of the steps to be carried out. For the sake of clarity, we can compare this type of approach to the one called information-intelligence, which valued work on procedural logic and not on real-time interaction with a machine. As we can imagine, one of the interests in making these processes explicit is that it becomes possible, then, to delegate them to the machine so that the latter carries out the calculations and proposes an answer. It was possible, for example, to generate plans, automatically, from the definition of the relationship between the different spaces, or to establish a formal grammar capable of generating new sentences. A consequent work also took place during this period to “encapsulate” various operations related to the design in the form of an algorithm.

This decade of the 1970s thus initiated a profound redefinition of design in architecture via, among other things, this question of making the information necessary for the design activity transparent. This information allowed the representation “to increase” in some way.

1.5.1. *Generating drawings*

At the same time that these questions about design were developing, a number of explorations were carried out, which were grouped under the name *spatial allocation*. The programs LOKAT (1969) or DOMINO (1972) enable the evaluation of the capacities of the computer to generate spatial combinations, from a mathematical approach.

In the tradition of Conway’s Game of Life and cellular automata, the idea was to generate plans, in an automatic (or semi-automatic) way, from a grid and some rules. It was a question of inserting a first block, more or less large, in this grid, which generally represented a room, and of adding a second one, according to relations of neighborhood and priority. It was then possible to make the system more complex by adding criteria. In keeping with Alexander’s thinking and his approach to rationality, it was a matter of deconstructing a part of the project – in this case a plan – into a number of architectural elements in order to recompose them, in the manner of what was traditionally called architectural composition. The program thus made it possible to generate plans, or at least to define “assignment” zones according to

predefined relational criteria. It was then a matter of the architect evaluating these results, generating new ones if necessary, in order to transform them into complex architectural proposals after some manual adjustments, sometimes consequent.

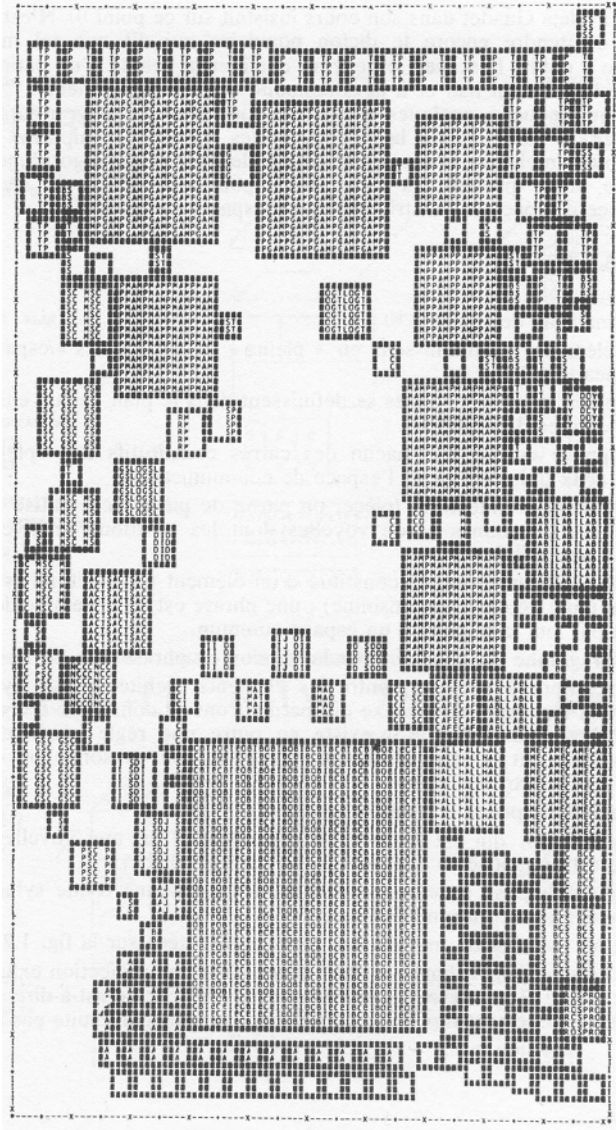


Figure 1.4. *Simulation of the Luminy site with LOKAT, simulation carried out by the GAMSAU, extracted from Coll. (1971)*

The LOKAT Project, by Allen Bernholtz and Steve Fosburg, allowed for the integration of about 10 criteria but, as noted by the architect Paul Quintrand in his critique of spatial allocation (Quintrand et al. 1985, p. 39), this process was cumbersome and needed to be simplified for reasons of efficiency and program stability. A rendering required between five and seven minutes of computation and was sent directly to a plotter. The LOKAT Project, as advocated by Bernholtz, was “A *Quantitative* Approach to *Qualitative Design* Decision-Making” (Bernholtz 1968, emphasis added). This program was relatively used, for example, by the Luminy project, in 1971 on an IBM 360/44, realized by the GAMSAU¹⁷ team (Figure 1.4). According to its users: “LOKAT is not only a new methodological means to create better and in less time, but necessarily forces us to ask questions about the relevance of the fundamental relations which are at the base of the grouping game” (Coll. 1971, p. 19). While the computer made the automated production of plans possible through spatial combination, the effort lay mainly in defining these relations and prioritizing them in such a way as to obtain exploitable results.

William J. Mitchell and R. Dillon proposed the DOMINO program, which, with a similar logic, generated plans for office buildings. It was a design assistance program that generated spatial organization schemes according to a heuristic process, built from four elementary principles: “1) the use of simple and short rules for the generation of complex forms, 2) the use of hierarchies for decision rules, 3) the use of economic rules, and 4) the use of divergent and convergent rules” (Coll. 1972, p. 25).

These rules were used to generate plans, as well as to perform an initial evaluation. In this respect, DOMINO had integrated a function that allowed backtracking:

It is often desirable, in heuristic search programs, to incorporate a capability to recognize that a particular sequence of decisions has led to a ‘dead end’ from which no satisfactory solution can result, and to backtrack to a previous vertex and restart the search along an alternative branch (Mitchell 1977, p. 463).

This function made it possible to readjust, manually, by going backwards, the processes so as to obtain a more appropriate spatial configuration, according to another evolutionary scheme. In 1977, Mitchell presented the operation of his program using the example of an office in Los Angeles. As we can see in Figure 1.5, after assigning certain fixed zones, such as elevators and stairs, DOMINO made it

17. *Groupe d'étude pour l'Application des Méthodes Scientifiques à l'Architecture et l'Urbanisme* (study group for the application of scientific methods to architecture and town planning).

possible to locate different zones and, from this “first” delimitation, the architects translated and transcribed these zones into habitable spaces.

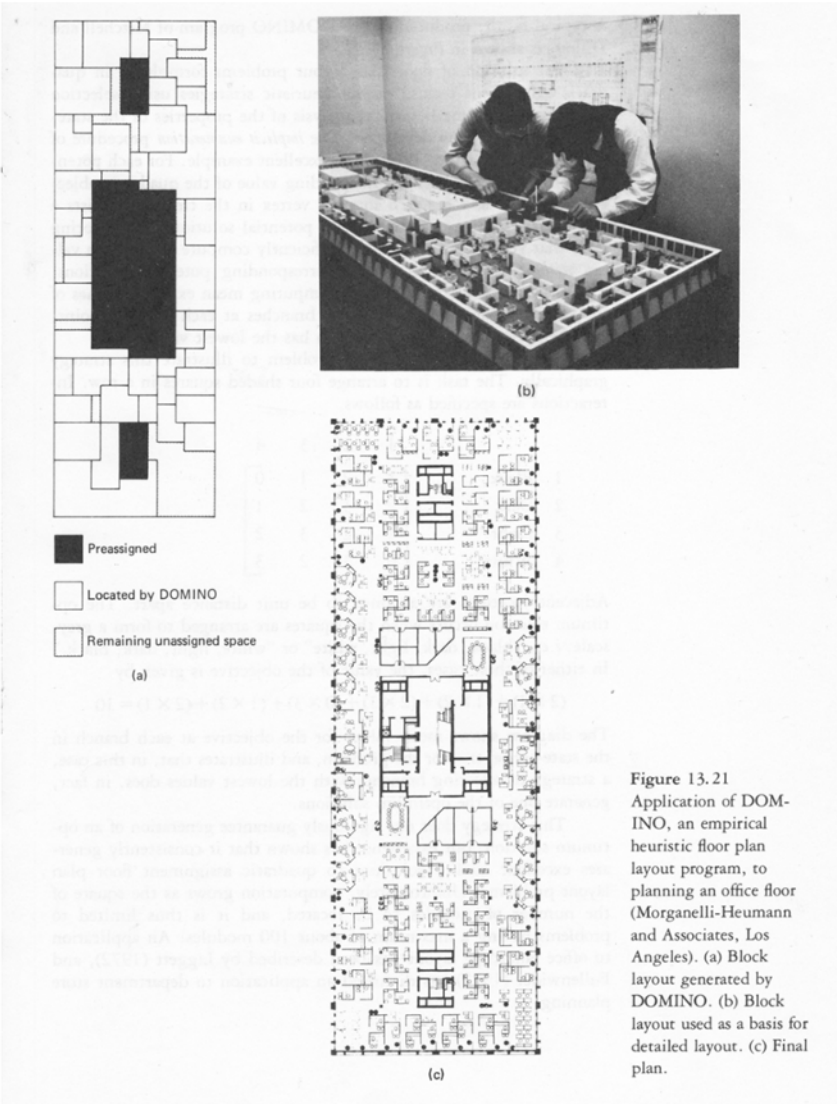


Figure 13.21
Application of DOM-
INO, an empirical
heuristic floor plan
layout program, to
planning an office floor
(Morganelli-Heumann
and Associates, Los
Angeles). (a) Block
layout generated by
DOMINO. (b) Block
layout used as a basis for
detailed layout. (c) Final
plan.

Figure 1.5. Plan made with DOMINO. Top left, the assignment of zones by the Domino program; top right, a mock-up that allowed for detailing; and bottom, the final plans for an office building by Morganelli-Heumann & Associates, L.A. (Mitchell 1977, p. 461)

In spite of these few achievements, this type of approach did not reach a large audience, and remained in an exploratory state. The limited capacities of the equipment certainly contributed to this failure, but one of the reasons, not to be underestimated, was the numerous criticisms raised about the fact that “the work of spatial allocation is not architectural” (Quintrand et al. 1985, p. 39). Spatial allocation may have offered some assistance, but in the end, it was a pittance compared to the whole process. As Quintrand pointed out, these ways of doing things did not take into account the formal aspect, which, according to the architect, represented a large part of the work. To illustrate his point, Quintrand had recourse to the demonstration made by Lionel March and Philip Steadman in the early 1970s, where they precisely showed the limitation of this kind of exploration. Using three house plans made by F.L. Wright in the late 1930s, they demonstrated that they were in fact all governed by the same relational pattern. This type of blueprint was not a representation of the building, but “a means of structuring information” (March and Steadman 1971, p. 28). If the authors had carried out this demonstration in order to show the similarity of the topological logics for houses of very different appearance, Quintrand, in his chapter “the insufficiencies of the mathematical tool” considered this demonstration as “the statement of the non-reducibility of the form to a functional graph” (Quintrand et al. 1985, p. 52).

1.5.2. Improving accurate algorithms

The decade of the 1970s was also devoted to the development of numerous algorithms dedicated to a singular problem, several of which were necessary to see the emergence of efficient modelers in the 1980s. One of the problems that occupied many architects was to find an algorithm capable of eliminating hidden features in a mainly three-dimensional visualization, such as perspective or axonometry, so as to leave only the visible features visible. In 1974, Sutherland, Sproul and Shumacker analyzed 10 such algorithms. Quintrand and his team, in 1985, presented three of them (de Montanari-Galimberti (69), De Newell and Sancha (72) and the Z-stamp). We will present here some of the aspects that had to be taken into consideration in order to approach this type of problem. First of all, the hardware was a determining criterion as to how to make these algorithms work. In fact, there was a distinction:

between screen-oriented and plotter-oriented algorithms, and in each case, depending on the drawing production technique: – line drawing adapted to plan and wireframe drawing production (raster scan screen, pen plotter) – elementary task drawing adapted to image production (raster scan screen, electrostatic plotters) (*Ibid.*, p. 187).

In addition to these specificities related to the machine used and the production technique, some of the important characteristics to be taken into account were the capacity to pass from 3D space to 2D space, as well as the capacity to compare the depth between two points (*Ibid.*, p. 186). The improvement of these algorithms, again according to these authors, would be an important occupation of the 1970s, but would be more or less settled by 1985, or at least “no longer considered a fundamental problem” (Gardan and Lucas quoted by Quintrand et al. (1985, p. 186)).

Another algorithm was the one developed by Charles M. Eastman and Christos I. Yessios, on the efficiency of the spatial operations of union, intersection and difference. As they noted, “To date, no efficient algorithms have been devised for executing all these operations on spatial domains when they are represented as closed curves” (Eastman and Yessios 1972, p. 1). For these authors, it was primarily a question of qualifying these curves, so as to distinguish between spaces inside and outside the curve, which they interpreted in a positive or negative order (Figure 1.6). The curve was the result of these two spaces and was then broken down into points. “The points lying on a curve (and therefore belonging to neither the inner or outer space) will be called *boundary points*. They are of two kinds: *corner points* (to be abbreviated as p-points) and *flat points* (f-points)” (*Ibid.*, p. 3)¹⁸.

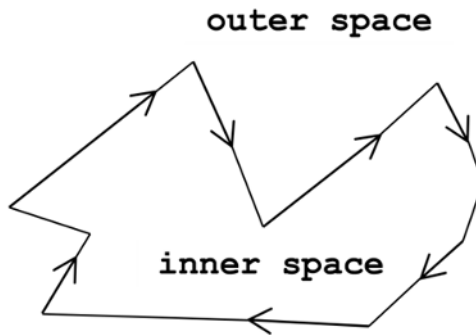


Figure 1.6. Presentation of the principle of defining the curve as the resultant between an interior and exterior space (Eastman and Yessios 1972, p. 41)

Thanks to this device, they managed to set up a whole system for formulating elements, here the curve, which were transformed in a mathematical way into an approximation of points and lines which allowed the operations mentioned above to be carried out with more facility. This work of definition by algorithms became the base on which more complex and complete programs were able to be carried out.

18. Author's emphasis.

This work in particular was prolonged a few years later by Yessios himself in the modeling of the void (Yessios (1987), see also Bourbonnais (2020)), which was constitutive to the Form-Z software.

Other algorithms were also developed during this period in order to respond to the various specific problems that arose, affecting many registers. Let us mention, for example, the surface subdivision which allowed a surface to be more fluid, *smoother*, and which played an important role in the morphing technologies explored by several architects in the 1990s (see *infra.* section 1.8). Some of these algorithms are still used today, such as the surface subdivision by Catmull-Clark (1978) or that of Doo and Sabin (1978)¹⁹. The objective here is not to be exhaustive, nor even to present the logic of these algorithms, but to underline the considerable and consequent work that was done during this period. In spite of the little light given to these experiments and developments, they had repercussions on the coherence, fluidity and stability of the software that would be developed later.

This work on algorithms was not limited to solving technical problems, or to improving existing methods, but found an astonishing extension in *Algorithmic Aesthetics*, proposed by Stiny and Gips (1978). The objective of this aesthetic was mainly to integrate the theory of communication, in line with the works of Shannon and Weaver, in order to elaborate a new theory of art. To constitute this aesthetic, the authors leaned on the interpretative and critical aesthetic of Beardsley, on the art of the appreciation of Osbourne, or on the aesthetic measure of Birkhoff. The particularity of this aesthetic theory, in addition to its systematization, was its double range: it was not only interpretative and critical, but was also generative, that is, able to generate works of art, according to the aesthetic system chosen beforehand.

In the interpretative and critical part, the algorithms were first used to operate an aesthetic criticism, by the description, the interpretation and the judgment-evaluation of the existing works. Each of these criteria – descriptive, interpretative and appreciative – gave rise to a precise algorithm, and the interest was to manage to combine the greatest number of them in order to grasp the complexity of the analyzed works. The objective was not to formulate a great and unique definition of what is a work of art, but rather, to try to list all the possible ways of making art, of considering its objects and, from this heterogeneous and sometimes contradictory set, to offer a meta-system which would include them all:

19. On this subject, see Teyssot and Jacques (2009, pp. 112–113), which offers a flashback and returns to the year 1978 to discuss the work, publications and careers of Edwin Catmull and Jim Clark, as much as those of Daniel Doo and Malcolm Sabin.

We see each of the ways of understanding and evaluating objects proposed in art and aesthetics as providing the basis for a definition of “work of art”. Just as we do not require that each way of looking at art be all-inclusive, we do not demand that each definition of “work of art” be all-inclusive. We are satisfied if each way of looking at art, and each corresponding definition of “work of art”, is internally consistent and coherent (Stiny and Gips 1978, p. 197).

This aesthetic seems, however, effective for a rather limited field, as the examples chosen to illustrate this theory show it: the experimental music, with its system of notation, the abstract art and the grammars of forms, to which Stiny and Gips had largely contributed. This aesthetic theory wished in a certain way to validate and ratify the grammars of shapes.

Shape grammars emerged more or less with the wave inaugurated by the linguistic turn, a few years earlier, where it was a question of establishing a grammar and syntax adapted to architecture, which should make it possible to generate forms automatically according to a set of relatively simple rules:

Shape grammars are similar to phrase structure grammars. Where phrase structure grammars are defined over alphabets of symbols and generate languages of sequences of symbols, shape grammars are defined over alphabets of shapes and generate languages of shape (*Ibid.*, p. 128).

Stiny distinguished between two types of grammars: an *original grammar*, which sought to generate new “designs”, and which was based “on a shape vocabulary and on the spatial relations between the shapes”, and an *analytical grammar*, which “made it possible to understand existing languages” (Eloy and Duarte 2014, p. 153), by typology or by architects. This type of exploration, which had almost completely disappeared by the mid-1980s, is experiencing a real renaissance since the advances of A.I. in the early 2010s and occupies many researchers.

We wish to emphasize here this distinction that was also put forward in the *Algorithmic Aesthetics* between, on the one hand, describing, interpreting and judging the existing and, on the other hand, the idea of generating the new. This question of originality could also go both ways. It was possible to obtain something new from an original recomposition of existing elements – which many architects are used to doing – but the original grammar was at another level, where the elements were not yet defined, without any *preconceived ideas*, and with which it would be possible to obtain a result that would exceed expectations.

1.5.3. Integrating building descriptions

While some architects were busy exploring the capabilities of computers to automatically generate and combine forms or relational schemes, another type of research was conducted, this time with the ambition to approach the complex reality of architectural practices. As it has been observed, the computer made it possible relatively early on to support architects in several management and organization tasks; tasks that are useful for reducing the interval between drawings and construction. Charles Eastman was sensitive to these issues and sought to propose a language that could encompass several elements of the design process, not just the act of manipulating a form (directly or indirectly). It is in this logic that he proposed the GLIDE language (Graphical Language for Interactive DEsign), which he presented alongside Max Henrion at the Siggraph meetings in 1977. This language inherited a prototype called BDS: Building Description System.

According to the authors: “GLIDE is intended to combine the advantages of an interpreted command language for interactive design with those of a procedural programming language” (Eastman and Henrion 1977, p. 25). This combination of the two languages, which can be understood as a reunion of the opposing attitudes of the inaugural CAD project (see above), was mainly intended to be adaptable to different types of users, thus allowing both the initiated and uninitiated to use these languages according to their skills in a more or less thorough manner. Several ideas were developed through GLIDE, including the idea of embedding the descriptions of an object in a machine; that is, not only working with the geometric representation of a building, but also including its description in the form of information. To do this, Eastman and Henrion defined three levels of representation: the image of the model, the shape of the model and the object of the model which “includes functional information, such as material, weight, rigidity, etc. Thus shape may be only one attribute among many others which describe the object” (*Ibid.*, p. 24).

At the time, the interest of this type of representation laid mainly in the possibility of making construction drawings with increased reliability, precision and efficiency. For example, thanks to the distinction between form and attribute, it was possible to attribute individual characteristics (e.g. color, materials, etc.) to objects of the same form (e.g. a door). This augmentation of the design model offered a better grip on data management and justification, cost estimation, as well as on the evaluation of the design of the structure²⁰.

20. As we will see later in Chapter 5, recent advances around this type of augmented representation will have even greater implications for how we think about design itself.

```

POLY PROCEDURE spiral.step(POLY centre;
    REAL riser,radius,r,angle,th)=
BEGIN
    POLY support =
        triangle(radius*0.95,-riser*0.8,th);
    POLY collar = column(12,riser,r);
    POLY plate = wedge(radius,th,angle);
    ! return the result of shape operations;
    CUT centre FROM COMBINE collar WITH
        COMBINE support WITH plate
END;

! To make spiral staircase, (dimensions in inches)
SET PROCEDURE spiral.stair(ht,radius,angle)=
BSET; INTEGER numsteps; REAL riser;
numsteps ← ht/8.0;
riser ← ht/numsteps;
POLY centre = column(12,ht+32.0,5.0);
POLY step = spiral.step(centre,
    riser,radius,3.0,angle,0.625);
FOR i TO numsteps
    DO COPY step={0,riser*i\0,angle*i}
ESET;

SET stair1 = spiral.stair(100.0,46.0,30.0);

```

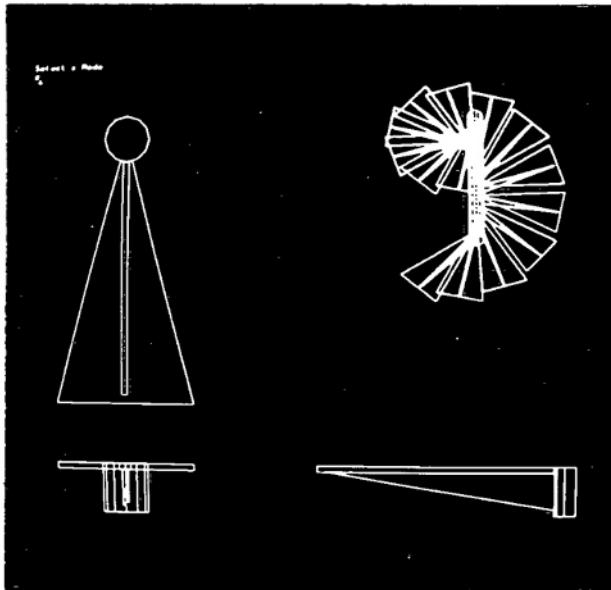


Figure 1.7. Example of the combination of two types of representation, realized with the Glide language (Eastman and Henrion 1977, p. 33)

This “augmentation” of geometric projections is one of the constants in Eastman’s reflections on the place of digital tools in architecture. It was in this sense that 10 years later he took a stand against the solid modeling that was beginning to interest architects. It was not so much that he was against modelers per se, but he considered that the specificity of architectural design was to combine several types of representation, each partial. “How to organize and use multiple representations becomes a research issue in design problem solving.” (Eastman 1987, p. 139).

1.5.4. Increasing the objects of the design

This idea of augmenting design had already been stated in the 1960s, with, among others, the article cited earlier by Bernhotz and Bierstone (1966), which, as we have seen, carried out an experiment of HIDECS 2 and 3 for a house. While this kind of formalization–decomposition–recomposition, according to another type of representation could be perceived as an augmentation of the design, this idea experienced a real boom at the beginning of the 1980s. As we have just seen with Glide, it became possible to add information to the graphic representation, because it was no longer a question of manipulating lines, but of composing with objects. Other projects appeared with similar ideas, such as the ARMILLA project, developed at the University of Karlsruhe in the early 1980s. Under the tutelage of the architect and designer Fritz Haller, this program was designed to facilitate modular design and mainly, “Haller’s software makes mostly hidden technical installations visible and autonomous” (Verhaeghe in Fankhänel and Lepik (2020, p. 192)). It was used, among other things, to solve the location of pipelines in an existing structure. In the early 1990s, ARMILLA served as the basis for a group of American researchers who sought to create a “knowledge-based approach to building design and performance evaluation” (Van Wyk et al. 1991). Their interest in ARMILLA was based on its interface which supported three design modes:

- (1) as a conventional CAD system with graphics and interactive tools to define and manipulate building objects; (2) as an expert diagnostic system extendable by the user to perform consistency and plausibility checks or other diagnostic knowledge; and (3) as an expert automated design tool to implement design knowledge in standard situations (*Ibid.*, p. 5).

Like the Glide language, ARMILLA sought to combine the different types of representation, while integrating a system of expert diagnoses based on different integrated knowledge that allowed us to evaluate the performance or issue a diagnosis. This idea was already known since the 1960s under the name of expert

systems. These systems appeared in force in architecture precisely during this period in the 1980s. As Jean-Louis Laurière explains in his general introduction to expert systems, or *reasoning aids*, it was necessary “for these programs to work in well-defined domains, to know how to learn the knowledge in this specialty, to know how to manage and structure it” (Laurière 1982, p. 26). Some knowledge was obviously more amenable to being managed and structured. This observation was made in an important inventory of expert systems in architecture, construction and public works (Guéna and Zreik 1988), where a multitude of ES were listed, with vast fields of application, but which nevertheless remained limited.

1.6. When industry takes over the software market

The beginning of the 1980s marked the rise of microcomputing and the emergence of personal computers. For the architectural community, the computer had already been in use for about 10 years, but there were many changes in the supply, capacity-manipulability and distribution of software. Numerous companies dedicated to software were born with the specificity that the main actors no longer needed to be linked to research projects and, with a few exceptions, were external to architecture. These new actors, detached from the institutions that supported the first experiments, brought with them new points of view regarding software, its use and its evolution–promotion–distribution logics. Software became increasingly specialized and tried to distinguish itself from the others. It is impossible to list all the programs, dedicated in one way or another to architecture, that tried to break into the market during the 1980s.

The creation of this market also had the consequence of progressively sidelining the architect-researchers who had the task of developing algorithms, languages and programs. This transition led architects to seek new places of expression, creation and especially reflection. For this reason, many associations were set up in the early 1980s²¹. The main tasks of these different associations were to organize meetings to discuss increasingly specific issues, as well as to present the latest technological advances.

21. ACADIA (Association for Computer-Aided Design In Architecture) was created in 1981, by, among others, C. Eastman, C. Yessios and W. J. Mitchell; eCAADe (Education in research in Computer-Aided Architectural Design in Europe) in 1983; Caad futures (Computer-Aided Architectural Design Future Foundation) in 1985; Caadria (Computer-Aided Architectural Design Research In Asia) in 1996; and SCAN (*Séminaire de conception architecturale numérique*) in 2005.

If we try to follow this history of the CAD industry, and especially the one related to architecture, we are immediately surprised by the few books that address the subject head-on and the rather disparate aspect of the information collected. Nevertheless, it is possible to find several accounts or interviews online with the main actors of this history, of which the most consequent and professional is undoubtedly the imposing *The Engineering Design Revolution*, by David E. Weisberg. In the form of 22 chapters, Weisberg retraces the story of these mainly American computer industries from the inside. There is also the *Autodesk File*, reissued²² many times, written by one of its founders, John Walker, or *The Graphisoft Story*, by its founder Gábor Bojár (2005).

From these different texts, it is possible to observe relatively similar beginnings in terms of the steps taken to see a software program break through to the market. For most of them, it was often a question of taking an initial program, a kind of prototype that came from personal research, with its particular context, and improving it, testing it and finally marketing it, following a path more or less full of pitfalls. As Bojár said about ArchiCAD: “We did not know how far our so-called prototype was from the real product” (Bojár 2005, p. 41). It is therefore quite difficult to define exactly when the software is created, or to fix an origin for it. Is it the moment when the prototype is created? Or even before, with the formulation of the principles of the software? Or only afterward, when it is demonstrated as a computer fair? Or when it is marketed? Or even later, when the first users use it?

It should also be remembered that the software market was not autonomous from the start. While the main financial contributions no longer come from the institutions that saw the birth of the first software (military and academic), it was the important industries that played a springboard role before the market acquired its full autonomy. There was a group of leading industries that developed their own software, in-house, perfectly adapted to their needs. As the main and only investor, these companies had perfect control over their product. This is often the reason why engineers left them to create their own software company, forcing them to acquire new customers. This transformation, progressive, required setting up a whole device in order to support the sale and use of these software and was not limited only to its technical development. This system included what we would today call marketing: presentations at trade shows, fairs (such as A/E/C systems), advertising, packaging, setting up sales outlets, providing after-sales service and, more generally, establishing the type of company that this small group of computer scientists wanted to found, finding a good name for it, etc. For these companies, there was also the

22. The latest version, the 5th, is nearly 900 pages. It is only available online. The latest hardback format, the 3rd, was published in 1989 and is approximately 550 pages.

need to choose the hardware on which the software could be used and be compatible with. It should be remembered that at that time the choice of hardware was important: Sun Flex, Sierra Data System, Victo 9000, Apollo, IBM workstation, just to name a few of the most important. On which medium: 8", 5" or 3 ½" floppy disk? Should we favor systems with hardware locks? These were also the kind of questions that these new companies had to answer.

Without going into the details of these well-documented stories, we are interested here in presenting the main orientations that served to build these programs and their interrelations with the architectural community. What kind of practice were they trying to answer? What questions did the software raise for the architects-users? Our objective is to shed light on the remote collaboration between these two domains. In this logic, our interest will focus on the very constitution of these software, their main bifurcations, in order to understand what constitutes them, fundamentally, and what distinguishes them from others: to grasp their *raison d'être*.

1.6.1. Drawing programs

Daniel Drake and John Walker founded *Marin Chip's Software* company in 1977, which operated until 1980. Two years later, in association with 15 other people, they founded *Marin Software Partners*. Then, for various reasons, mainly related to the choice of the economic model, they decided to create the company *Autodesk*²³. Around the same time, Mike Riddle²⁴ was working for the Frank Lloyd Wright Foundation and using INTERACT, a program that was mainly used to (re)draw the American architect's buildings in plan and perspective.

It was a system that combined a drawing tablet with a computer, where it was possible to redraw the plans in a computerized way. It was INTERACT that formed the basis of AutoCAD. In fact, INTERACT was losing momentum and needed better hardware to work. *Marin Chip's Software*, Walker in particular, provided him a motherboard S-100 and a TMS-9900 processor which enabled him to make it

23. Autodesk was first named Desktop, but that name was rejected by the Secretary of State. In The Autodesk File, Walker describes the relatively long and complex processes involved in defining the name of a piece of software, product or company.

24. Mike Riddle occupies a special place in the Autodesk adventure. He was one of the founding members, but the many conflicts with John Walker led to him being sidelined from the CAD programs, and made him leave the company quickly. He nevertheless sold them INTERACT which constituted the bases of AutoCAD. At the end of the 1980s, he was at the head of the Evolution Computing company and developed FastCAD, then EasyCAD.

function at its full capacity. It was then transferred into a program called Micro-CAD which became AutoCAD and was exhibited at the 6th *West Coast Computer Faire* in 1982. AutoCAD-80 (for its compatibility with the Intel 8080 and the Zilog-Z80 microprocessor) was released in 1983. In reconstructing AutoCAD's beginnings, it is also important to add to this list the working paper presented by Walker to the *Marin Software* partners in January 1982, the one that "served as the introduction of the concepts and the invitation to the meeting to organise the company" (Walker 2015, p. 13).

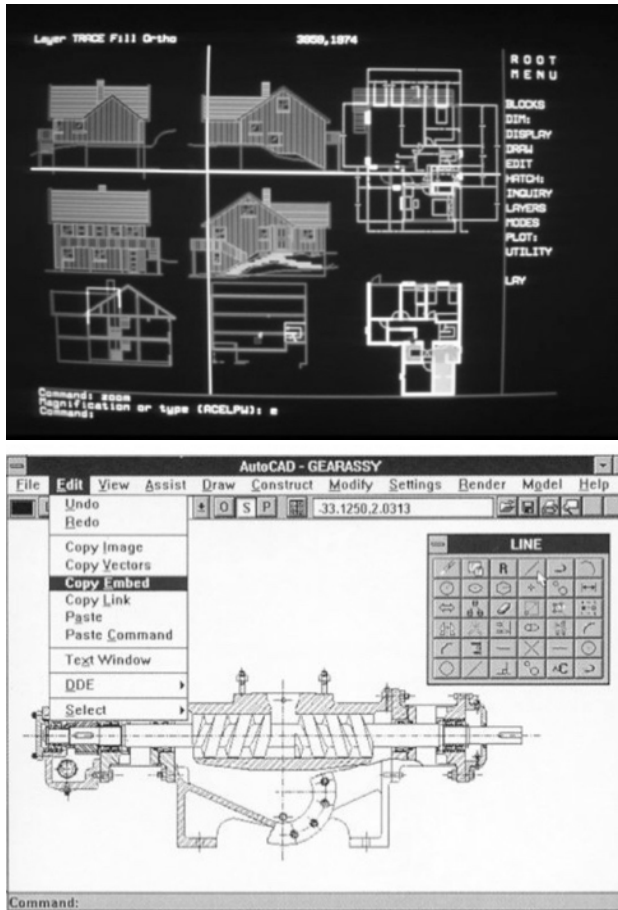


Figure 1.8. Evolution from AutoCAD 80 (1985) to AutoCAD 12 (1992) © Autodesk.
For a color version of this figure, see www.iste.co.uk/bourbonnais/architectural.zip

The company then went through several stages of development, including significant and rapid growth starting in 1985, but which, according to Weisberg, seemed to have no clear direction, with an almost non-existent customer relationship. At the end of 1987, Autodesk experienced a major crisis with its flagship software AutoCAD. The price of this software was much higher than other equivalent CAD software on the market, and the software was falling behind on the technical level: mainly on the 3D modeling and the user interface. While different causes can explain these delays, the consequences were that AutoCAD never really became a software used, mainly, for its 3D modeling capacity. It remained above all a drawing program, described in a computer promotional magazine as: “a design and drafting program designed to speed up repetitive drawing tasks and simplify drawing revisions” (Apple Company, Inc. 1989, p. 5).

It is only from version 11 onwards that an advanced modeling extension was acquired and developed, mainly from the solid modeler Padl-2, from the University of Rochester. This solid modeler interested several architects including Marcos Novak, who used it for the West Coast Gateway project in 1987, or Chris Yessios. The Padl-2 was then used in architectural circles because it allowed the realization of the first solid geometries, by integrating boundary representation techniques. As for the user interface of AutoCAD, which was not considered very user-friendly, it was improved and underwent significant changes with version 12.

The choice of favoring drawing techniques – from geometrical projection to execution drawing – by quickly integrating functionalities linked to drawing, such as automatic hatching; easy manipulation of lines, polylines, arcs, fillets, blocks, explosions; snaps, etc., clearly indicates the type of practice with which this software sought to associate itself.

At the same time, Keith Bentley created Pseudo STATION for the DuPont chemical company. After getting the company’s approval to market it, Keith partnered with his brother Barry to create Dynamic Solution in 1983, which became Bentley System the following year. Pseudo STATION was compatible with Intergraph CAD and could be used with a low cost terminal (DEC VT-100). Pseudo STATION was used to create Micro STATION. It was this program, running on a Compaq 286, that the Bentley brothers presented at the 1985 Intergraph user meeting in Orlando. The target audience was clearly the same as that of AutoCAD, as revealed in the manual *MicroStation for AutoCAD Users. A Bi-directional Guide*, or the comparable price of 3000 US\$ (in 1989). In the 1990s, Bentley began working to ensure compatibility with .dwg files. They succeeded in importing these files from AutoCAD to Micro STATION thanks to a small company, Marcom AutoDirect toolkit. Even as AutoCAD was being restructured at the turn of the

1990s, Micro STATION was known for its efficiency of use, as research consultant Joel Orr, who started out at Autodesk, pointed out before he was laid off. “The system [Micro STATION] is designed for production. You can sketch, design, play around with it, but its primary features are speed, power and ease of use contrasted with ease of learning)” (Orr quoted in Weisberg (2005, p. 10-12)). This speed and power was their leitmotif, even if it meant that its intuitive accessibility was sometimes complicated. At the end of 1994, PowerDRAFT was released, a drawing software specially designed for the design of complex projects. It was chosen by architect Greg Lynn “for its ability to model elaborate geometries, particularly precisely defined curves” (Bird and La Belle 2010, p. 244).

If we return later to the explorations of the 1990s (see *infra.* sections 1.8, 3.5 and 4.2), let us specify here a strategy that was implemented for many software companies: the diversification of their products. Indeed, in addition to a flagship software, which saw many improvements, from version to version, appeared with more or less success, new products, complementary, generally dedicated to a specific task. These new products wished, on the one hand, to acquire a new market, as, for example, Micro STATION TRIforma, a 3D modeler, especially conceived for the building industry or, for other reasons, like the not insignificant one to adapt to the transformations of the practices, like the arrival of the BIM technologies, which saw a new range of software appearing, whose compatibility of one to the other became paramount.

1.6.2. Representation of spatiality

In 1982, Gábor Bojár and István Gábor Tari founded Graphisoft in Hungary. Previously, Bojár worked for the Institute of Geophysics and had the task of “drawing high quality topographic maps, from which it was possible to visualize them, from different angles, thanks to a system that allowed us to rotate these 3D surfaces in space” (Bojár 2005, p. 23). After this work, he was hired by the Atomic Reactor Company in Paks, where he was asked to make a tool that could model a pipeline system for a reactor. He decided to use his first tool used for maps as a basis, even though the similarities between geographical features and pipelines were rather meager. There was, however, a common theme in the need to import a large amount of data from several hand-drawn plans. It is important here to point out that the realization of a piece of software is usually done by addition: recovering the code of one, to redeploy it in another, modifying it according to its specific needs. It is also noteworthy to specify that for these two companies, the software was intended for a single user. The program was tailor-made for the Institute of Geophysics or for the reactor company. For the latter, it was Tari who developed the

programming language (which his student would develop) based on the fact that the dimensions of the conduits for the pipeline were relatively constant, which allowed for a more efficient flow of information. They developed a geometric description language, GDL (Geometric Description Language), which would always be used for parametric objects in ArchiCAD (*Ibid.*, p. 30). For Bojár and Tari, this experience with pipelines was instructive in several ways. Among other things, it gave them the ability to distinguish between visible and non-visible elements, with particular attention on how to think about the arrangement of objects – in this case, mainly pipes – in space.

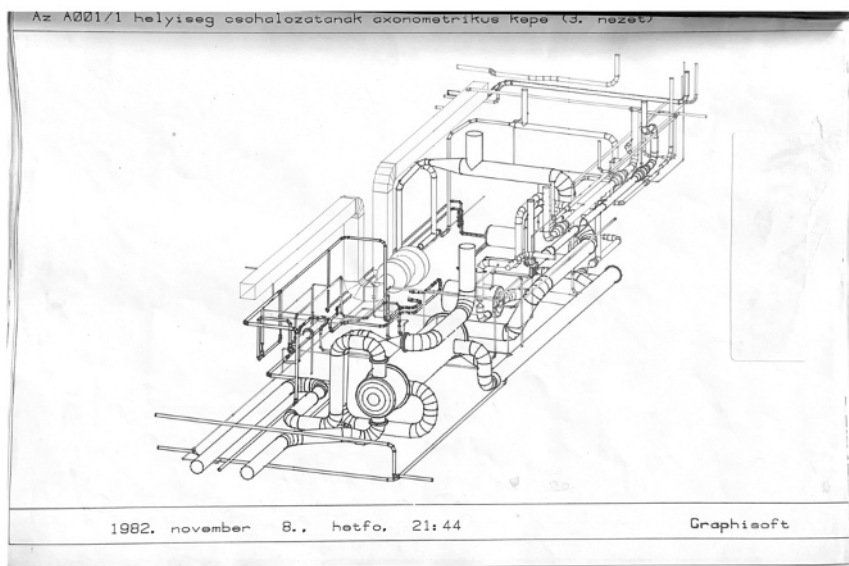


Figure 1.9. A 3D model of the proposed pipeline network for the first nuclear reactor in Hungary. Image courtesy of Gábor Bojár. For a color version of this figure, see www.iste.co.uk/bourbonnais/architectural.zip

In the particular political context of Hungary in the 1980s, Graphisoft managed to create a fast and stable program on a simple HP computer; their program could compete with programs that ran on much more powerful and expensive machines. Through a series of circumstances and tricks that Bojár recounts in *Graphisoft Story*, this Hungarian entrepreneur managed to obtain a visa that allowed him to leave Hungary and export his product. Confident in the quality of his computer program, Bojár was largely surprised, and disappointed, that no one wanted to buy it, even though it was cheaper and just as good. After some simple observations, he realized

that these companies did not want it, simply because they had already spent a lot of money to acquire the equipment to perform the same tasks and, in addition, their employees had “adapted” to these programs. Bojár’s solution to this problem was to target another market, the one that could not afford these large, expensive machines: architects!

Graphisoft developed the program Radar (RAumliche DARstellung (spatial representation)), which came in two versions: Radar CH was specially designed for architects, while RAPID kept the pipeline design as its main use. It was thanks to a combination of circumstances that Radar was transferred to the Apple Lisa system. As Bojár recalls, the presentation of a computer was not interesting in itself, it had to work. It was in this logic that an Apple representative, having seen a first demonstration of the software in action, asked him if it was possible to rewrite it on the Lisa, quickly. Managing to transport a Lisa to Hungary and thanks to some talented engineers, Radar could be presented a few months later at one of the biggest computer fairs in Europe (CeBIT). The demonstration created an event that went all the way back to Steve Jobs. Radar became ArchiCAD.

Only 14 copies sold of Version 1.0 of ArchiCAD were sold, in Italy and Germany. It is not insignificant to specify these countries, because local construction standards influenced some of the software’s functionalities, such as the units of measurement. In Germany, for example, the very strict construction environment required architects to be accurate to the half-millimeter, while the US operated in foot-inches. Version 2.0 was released on the Apple Macintosh, a more robust machine that had just replaced the Lisa. Because of the internal incompatibilities of the two machines, they had to rewrite all the instructions in order to “transfer” their program from one machine to the other.

It was only then that the company really took off, and Bojár defines this as the real birth of the software (1986); the moment when ArchiCAD was used by architects for real buildings and when they received their first customer feedback. There was indeed a whole system of dealers who acted as intermediaries between the client and the companies, and who ensured the after-sales service. By exchanging letters, the dealers would relay the problem that a user had encountered to Graphisoft and they would seek a solution in order to provide an answer to the customer, or to offer him compensation.

The user interface was also one of the main concerns that Graphisoft insisted on, for example, in version 3.1, which proposed an interface that was self-explaining, playful and had sex appeal (*Ibid.*, p. 77). It was developed by Tamas Hajas and

Gyuri Juhasz, an architect, who a few years later founded his own consulting firm for interface ergonomics.

In 1989, the software was presented as follows:

ArchiCAD is a high-end two-dimensional/three-dimensional architectural building modeler that optimizes the process of producing plans and obtaining elevations, sections, bills of materials and three-dimensional visualizations. With a well-integrated Macintosh interface, ArchiCAD speeds up design by providing a library of building components and by letting designers make selections from a palette of elements such as slabs, walls, and roofs (Apple Company Inc. 1989, p. 4).

Unlike AutoCAD or Micro STATION, ArchiCAD was no longer a drawing software, but concentrated its efforts on the representation of elements, in space, from component libraries. As we have tried to show, the implementation of a piece of software, and its evolution, responds to a series of factors that largely exceed the only technical dimension, even if this last one remains structured in the orientations which these software will take.

1.7. User interface improvements

The 1980s also marked important advances in terms of the question of the graphic user interface (GUI). While, as we have seen, there have been visual displays since the end of the 1950s, there was a changeover with, among others, an important project carried out by the XEROX PARC, around the personality of Alan Kay. For Kay (1972), the computer should no longer be used only by engineers, nor by adults, but by “children of all ages”. For this reason, Kay was very interested in the research carried out by the cognitivist Jerome Bruner, who sought to update the thinking of Jean Piaget, by being particularly interested in the stages of development in children. For Piaget, these stages are: kinesthetic, visual and symbolic. Bruner suggests that there are distinct mindsets corresponding to these different phases, and that these mindsets continue to exist as the child grows: that is, one mindset does not replace the previous one, but is added to it. Bruner gave slightly different names for these mindsets: “he called them *enactive*, *iconic*, *symbolic*” (Kay 1990, p. 195).

Kay literally took up these different mentalities to approach the type of relationship to be engaged with the computer. This could be translated as follows: “Know where you are, manipulate” was handled by the mouse; what was iconic,

“recognize, compare, configure, concrete” was related to the images generated in the window – the computer icons – and finally, the symbolic, defined by “tie together long chains of reasoning, abstract” (*Ibid.*, p. 197), was related to the programming language, for example, the object-oriented one in SMALLTALK. All manipulations performed by a user via the interface involve “constant interplay between different mentalities” (Manovich 2013, p. 99). This tiered layout of the interface as well as its deployment became a standard that led to a specific relationship with the machine, favoring a real-time interaction with icons, a cursor, on a simulated desktop and where applications appeared-transmitted “on” a window. This arrangement was a success that participated in the democratization of the computer and its programs, thanks to a series of possible applications, thus paving the way for even more direct, simpler and user-friendly access.

Despite its centrality, the interface nevertheless occupies a paradoxical place in software practice as much as in software studies. In 1990, cognitive psychologist and design expert Donald Norman pointed out in an interview for *The Art of Human-Computer Interface Design*: “The interface is the wrong place to begin” (Rheingold 1990, p. 6). Norman’s point was to emphasize that the interest is not in the interface itself, nor in the computer, but rather in the task it enables. As Manovich would point out a few years later: “People prefer it [this GUI] not because it is ‘easy’ or ‘seamless’ or ‘intuitive’. It is successful because it was designed to help them think, discover, and create new concepts using not just one type of mentality but all of them together.” (Manovich 2013, p. 100). This work on the interface, its accessibility, its availability, led to an intense period of questioning around architectural design and its very redefinition. These interrogations and debates were placed in the prolongation of the questions raised by the pioneers and that of the decade which followed, with the particularity, because of the advances of the interface, to direct many of these debates on the question of the representation, the drawing and more largely of the image in architecture.

1.7.1. *The questioning of drawing*

The question of drawing occupies a special place in the history of architectural design, and even of architecture itself. As Cardoso Llach points out, following Mario Carpo, a clear opposition has been established between the drawing and the building to be constructed, since at least the Renaissance and the treaty of Leon Battista Alberti. This prevalence of drawing also resides in the place that it occupies in the teaching of architecture, because drawing, as Quintrand recalled: “is not only the tool of representation essential to the architect for the study and communication of the project, but also the tool for the acquisition of knowledge [...]; it is the means of learning to

see” (Quintrand et al. 1985, p. 59). Drawing is not only used to conceive and imagine. It is also through drawing that the buildings reached (with the photographs), until recently, the architects and future architects, and thus participated in the constitution of a collective memory. While drawing, its variations and derivatives – from the sketch to the execution drawings through the draft or the perspective – occupied a natural place and was considered by many as The Technique for architecture: the arrival of the computer called into question this indestructible place of drawing. As we have observed in the first parts of this chapter, the pioneers as well as the experiments of the 1970s with the idea of changing the type of representation had in a way “strangled” drawing, to use Quintrand’s expression, or at least, they had tried to displace the main medium of design. Indeed, with the exception of *Sketchpad*, the task assigned to the computer was no longer that of drawing. There were (almost) always geometric representations (albeit sometimes abstract), but these were no longer the (only) focus.

However, as we have just seen with the appearance of the market, the techniques of drawing came out triumphant, occupying a predominant place of the software offer, so much so that Robin Forest, one of the participants in the CAD project, underlines this passage from CAD, d for design to CAD, d for drawing:

[So] CAD really meant more in 1965 than it does nowadays [1989] ... The “D” became not design, but drafting. I once asked a vendor what his program could do other than produce drawings. He looked absolutely amazed that you would ever want to do anything other than produce drawings. (Hurst et al. 1989).

If indeed a large part of the market proposes software centered on drawing techniques, the “rewriting” of these techniques makes their artificiality as well as their exteriority visible, whereas they had been considered as self-evident, almost natural to the architect: as an extension of themselves. Yet, the drawing is also constructed, requiring learning and repetition so that it becomes operative; even for the drawing, it is a question of acquiring know-how. This was also pointed out by the philosopher Bernard Stiegler in the *Historique du dessin*, where the philosopher shows how memory and imagination are constructed:

This amounts to saying that memory is “objective”, and above all that imagination, which is at the same time the forgetfulness of memory and the work of memory, is also objective, that it is elaborated in a technological practice, which is a practice of repetition – and that the idiomatic differentiation does not originate simply in the subject, but supposes a complex of repetition, of rereading of which the “subject”

is a carrier instance, but not the source. Imagination is not a psychological faculty (Stiegler 1986, p. 282).

Through a re-reading of drawing, Stiegler showed that the imagination is largely constituted outside the subject, thanks to its technical projections, which are internalized by repeated gestures. It is thanks to this learning, to this internalization by repetition, that the drawing seems natural and that it follows the imagination more closely. This is even truer when compared with the processes generated by software.

Some artists and architects have attempted to highlight this discrepancy by reproducing the gesture, without changing its nature, on a new instrument, such as the “digital sketches” of artist Ilona Lénárd, a member of the ONL architectural group (Figure 1.10).

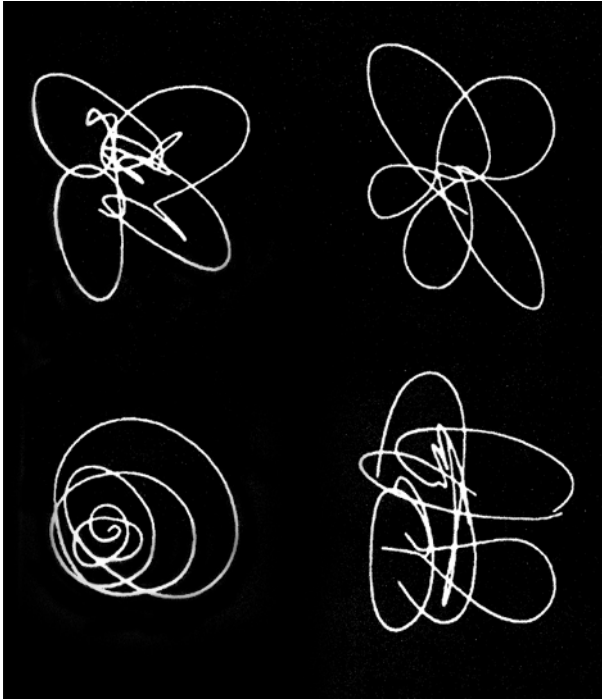


Figure 1.10. *Digital sketch by Ilona Lénárd, Sculpture City, Project by Attila Foundation 1994, © RAM Gallery, Rotterdam Netherlands*

“Nothing seems more contradictory than sketching on the computer. A sketch is intuitive by definition. [...] Intuitive computer sketching exploits the mutant power of the program. [...] The designer accumulates his energy and discharges it in a series of mouse movements, becoming visible in digital space” (Lénárd 1998, pp. 4–5). These digital sketches were then integrated into an architectural practice, constituting the “genes” that initiate the forms of the buildings, where the latter become building-sculptures as in the *Sculpture City* project.

Beyond these few explorations, which have not been numerous among architects, it should be noted that there is a type of software where it is a question of “drawing” by tracing lines, via a peripheral: mouse, “passive stylus” or simply with the tip of the finger. While this type of software, such as SuperPAINT, which appeared in 1972–1973, was ultimately of little interest to architects, it is possible to observe a spectacular return in architecture schools to this type of practice where it is a question of drawing lines, via a digital instrument. This return has been possible thanks to the recent advances proposed by some applications, but especially thanks to the graphic tablets and their passive pens, which make sketches, in a relatively precise way, on a tablet (Lescop and Chamel 2020)²⁵.

While, as we have just briefly seen, drawing has “traced” its way through and around digital technologies, even absorbing a change in nature, a different way of designing appeared in the mid-1980s: that of designing from a volumetric model and no longer through orthogonal projections, nor by arranging more or less defined objects in a space.

1.7.2. Design with a 3D model

In the mid-1980s, several debates animated architects around the progress made in computer science. One of them was the question of whether it was essential to maintain the supremacy of drawing, and more specifically of orthogonal projection – plan, section, elevation and perspective – now that it was possible to design directly in volumetry thanks to the computer. The source of this questioning came largely from engineering solid modelers, such as PADL-2, which architects took advantage of before quickly becoming disillusioned:

Ironically, as we fell in love with solid modeling and its set theoretic operations of union, intersection, and difference, we somehow failed

25. This return to drawing is also observable in much more experimental practices made with different machines, such as robotic arms. On this subject, see the “Machining Emotion” series by Ilona Lénárd, and Chapter 4 on the diversions of the machine.

to recognize its inefficiencies for architectural modeling. By the early eighties, we had to conclude that solid modeling is simply not capable of expressing and accomodating architecture. (Yessios 1987, p. 143).

Faced with this disillusionment, two types of responses were formulated. The first position was defended by architects like Charles Eastman who considered that the difficulties of solid modelers were not only related to technical problems of performance or workability, but to other deeper reasons “inherent to architectural drawings and practice” (Eastman 1987, p. 133). As we have seen, Eastman was not against modelers, but they needed to be integrated into a larger system that combined different types of representation. In 1987, his critique was based on valuing drawing techniques. “Drawing allows “fuzziness” in defining the artifact, which solid modeling does not.” (*Ibid.*). Drawing facilitated decision-making because it allowed problems to be addressed in isolation “without requiring full definition of the 3-D geometry of the components involved. That is, designers could abstract the full design to focus on particular issues.” (*Ibid.*, p. 137). In contrast, the geometric model was too rigid, required a lot of information to be relevant and could not handle the vagueness and uncertainty²⁶ unique to architectural design, which is why it inevitably had to be combined with other types of representation.

Nevertheless, there was a growing audience that appreciated the rigidity of the model, as it allowed for “objective and complete images” (Quintrand 1986, p. 241). Even if Quintrand considered the plan more effective in giving the measure than the geometric model, the latter nevertheless allowed for evaluating and measuring “the state of the architectural project in all its three-dimensional complexity and to manage its coherence” (*Ibid.*, p. 240). It came closer to the building to be constructed and, in this sense, there was no contradiction with the idea developed by Eastman and Henrion, that of increasing the modeling. For this reason, it is worth mentioning that Eastman’s criticism in 1987 also concerned the distinction between geometric projections and construction drawings. Indeed, “the reason architects do not design directly in 3-D is the lack of sufficiently flexible representations for 3-D design” (Eastman 1987, p. 135). Modeling became useful, practical or powerful especially when augmented, to automatically obtain contract documents for construction. It was on this point that the dispute with Yessios lay.

Chris Yessios’ criticism of solid modelers was different. For the Greek architect, the solid modelers used in the industry were not suitable: they were inefficient for architectural practice. However, this did not mean that he rejected all types of modelers; on the contrary, this observation allowed him to develop a type of

26. We use this notion of *uncertainty* on purpose, because at the same time, Lebahar (1983) proposed to think of architect drawing as a process of reducing uncertainty.

modeler specific to architectural design. His idea was that modelers should no longer be based on a design from solids, but from voids. To this end, Yessios described Space Enclosures as skins of voids that were generated from an axial skeleton defining a contained part and a container. This way of modeling, starting from voids, allowed a greater efficiency in order to respond to the requirements of architectural design, when it was a question of realizing relatively complex forms, of designing irregular spaces, or of realizing several manipulations on any modeling. Yessios saw the modeling of the void, this spatial envelope and consequently the relationship between interior and exterior, as one of the essential and constitutive elements of architectural composition. It is this property that he wished to integrate into the logic of the modelers of this era.

This idea was pursued and made a reality by the creation of the Form-Z software. Yessios announced from the outset that: “Form-Z is an argument against drafting” (Serraino 2002, p. 6). Yessios’ challenge was to transfigure the sketch with a flexible and intuitive device that would offer the fuzziness that Eastman reserved for abstract drawings, without resorting to geometrical drawing, but directly from volumes. It was not a question of approaching the building to be constructed, by accentuating its rigidity, nor of offering different layers of information, but on the contrary, of seeking to offer an instrument which, in the same way as the sketch or the draft, would allow us to conceive directly in volumetry.

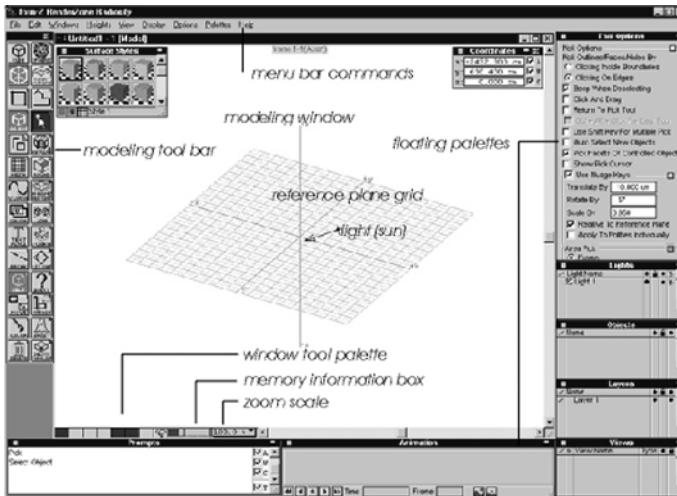


Figure 1.11. The default interface of Form-Z software displays a three-dimensional design © AutoDesSys, Form-Z. For a color version of this figure, see www.iste.co.uk/bourbonnais/architectural.zip

Form-Z favored the volume-based design, as can be seen from its default graphical interface, which revealingly represented a three-dimensional space, with the three-axis system represented (x, y, z coordinates) (Figure 1.11).

This way of designing from volumetric models became a real possibility for architects thanks to the arrival of many 3D modelers which, accessible at the beginning of the 1990s, became more efficient, more accessible and more adapted to architecture. In addition to Form-Z (1991), a number of modelers appeared on the market that architects took advantage of. We can name *3D-Studio* (1990) by Autodesk, *Solid Works* (1995) by Dassault Systèmes or *Rhinoceros 3D* (1998) by McNeel and Associates. These modelers each offered specific modes to build volumes, favoring such types of manipulations–transformations according to the type of modeler (e.g. surface or polygonal modelers). It should also be noted that they were not only used for design, but also to generate renderings or animations. Again, these modelers inherited from prototypes and research done in other contexts²⁷.

Handling 3D models, working directly from volumes, has certainly contributed to offering an alternative to drawing-based design, but its real contribution lies elsewhere. It is indeed possible to identify two phenomena on which 3D modeling has been the driving force of change. These modelers participated in the renewal of practices in the 1990s, with, among others, mapping and morphing techniques that made it possible to animate forms through their images. It is very likely that this design from a 3D model has had its say in the central place occupied by the digital model in BIM practices (see section 1.9.3 and Chapter 5).

1.7.3. Moving images

The possibility of animating images thanks to computational processes could constitute, by itself, a whole book. We will underline here only some elements that will allow us to understand the shift that took place in the way architects did things in the 1990s, limiting ourselves to the field of art and architecture. Even before tackling this question of animation, it seems possible to establish two types of

27. As an indication, we can present these prototypes and research: The *Thub* program, created in 1988 by T. Hudson, G. Yost and J. Powell, was used for *3D-Studio*; Void modeling research, carried out in 1987 by Yessios for *Form-Z*; *Premise 87*, produced by G. Hirschtick, at the MIT CAD Laboratory, under the supervision of D. Gossard, to then be purchased by computer vision and become *CADDS5*, which then took part in the creation of *Solid Works*; and finally *Sculptura*, a “mesh” modeler made by Gibson, then *Sculptura 2* in 1993, which became *Rhinoceros 3D*.

distinction. On the one hand, the “object” in the image, which in art history has long been defined under the opposition form-ground, and, on the other hand, the very functioning of animation, or the setting in the motion of images.

Animations in architecture can thus simulate: (1) a movement *in* architecture, a “walk through” as a stroll around and in a building; (2) a movement of the buildings themselves, in line with the projects of the Archigram group, such as *Walking City* (1965); and finally (3) an internal movement of the object itself according to internal or external movements–forces, such as the morphogenesis of an architectural object.

As far as the functioning of animations is concerned, we will say, based on the article by Fankhänel (2020), that the first animations realized by computers were made from a series of fixed images and that they were then put in movement, like the animation techniques for cartoons or cinema. Michael Noll’s *Computer Ballet* in 1965 or the first “walk throughs” such as those made by Donald P. Greenberg, among others, between 1969 and 1972, at Cornell University, used this type of process to “simulate” movement. This process was long and tedious:

A few minutes’ film of this type often took hours and many night shifts to create on mainframe computers with the required computing power. The result was an analog film comprising digitally programmed images. The process was not only time-consuming, but laborious and costly. (*Ibid.*, p. 122).

As Fankhänel explains, this type of process was mainly due to the visual displays used at the time (such as the TX-2) where it was not possible to obtain a stable and continuous image, and certainly also due to the low data storage capacities. It was in the mid-1980s that the changeover took place with a series of technical improvements that allowed the emergence of what would be called computer graphics and infographics²⁸.

A number of architects-researchers became interested in image technologies and began to realize synthetic images and quickly sought to animate them. It is possible to mention the research carried out by certain institutions, such as those of the CIMA (*Centre d’informatique et de méthodologie en architecture, Ministère de*

28. It is also not insignificant to see the emergence in architectural circles of a marked interest in the image in general, with various events such as the interdisciplinary symposium *Imaginaire numérique* (Renaud 1986), which was held during the International week of the calculated image in Saint-Etienne, the exhibition *L’architecture et son image* organized by the CCA in 1989, or the work of the architect Jean-Paul Jungmann, *L’image en architecture* (1996), to name some of the main events in French.

l'Équipement et du Logement), which, after computer-generated images, became an important axis of study; they questioned “the nature and functioning of the language of creation of the animated computer-generated image, as well as the improvements that could be brought to it by the use of artificial intelligence techniques” (Arnold 1988, p. 58). Within the framework of their research, they experimented with these techniques to realize a series of synthesis films (Michel Bret and Annick Emery) or a “tele-opera” (Sabine Porada). These applied research studies showed the displacement which took place in the construction of the “objects” of the image. As the researcher Madeleine Arnold underlines it, it was not questioning the simulation of the phenomenon, but its effect, nor the representation of the objects, but their process of generation (*Ibid.*, p. 77).

These experiments in the field of architecture were not isolated, but were carried out in several fields at the same time and resulted, a few years later, in the creation of numerous companies occupied with image, computer graphics, one of their main tasks being to create special effects for the cinema. To name a few examples, let us recall that *Softimages*, developed by Daniel Langlois, realized the special effects for *Jurassic Park* (1993), or that the *Alias Power Animator*, developed in Toronto, was used for *Abyss* (1989) and for the T-1000 in *Terminator 2*, or that the *Thomson Digital Image* (TDI), created in France, produced numerous short animations for French television. Over the years, a fierce competition between the main players took place with numerous mergers and takeovers²⁹. While the evolution of animation software follows a conventional pattern of technical development, in four phases and drawing an “S” curve: emergence, growth, maturity and saturation (Figure 1.12), it is interesting to note that each of these phases is occupied by a different player, or group of players.

During the emergence phase, many laboratories, such as CIMA or researchers such as Daniel Langlois, work, in a relatively isolated way, on the premises of these technologies. Then, a phase of growth takes place in the fields that have the means to realize this technological fluctuation-acceleration. In the case of the moving image, these were the film (special effects) and animation industries. During this phase, where software is generally developed in domains outside of architecture, it is possible to see a series of “borrowings” appear, where some architects are fascinated by these technologies, seize them and divert them, in order to explore them in their

29. To illustrate this ferocity, let us say that after a few transactions, Silicon Graphics Incorporated (SGI) acquired Alias and Wavefront technologies (which had already merged), in the same year that Softimages was purchased by Microsoft. In 2006, Autodesk brought together the advanced features of Wavefront's *Advanced Visualizer* (in California), TDI (in France) and *Alias Power Animator* (in Canada), to create MAYA. Autodesk acquired Softimage in 2008 and therefore became one of the main players in the sector.

own domain of expertise. Finally, there is a phase of maturity and stabilization, where other companies come to recover the technical efforts to integrate them in less specialized and consequently easier to access tools, like Maya, which allow the experiments that had been realized in the previous phases to be made accessible. This evolution continues and ends with a saturation, which shows both the limits, and requires, according to this scheme, a substitution technique in order to renew itself and engage on a new curve.

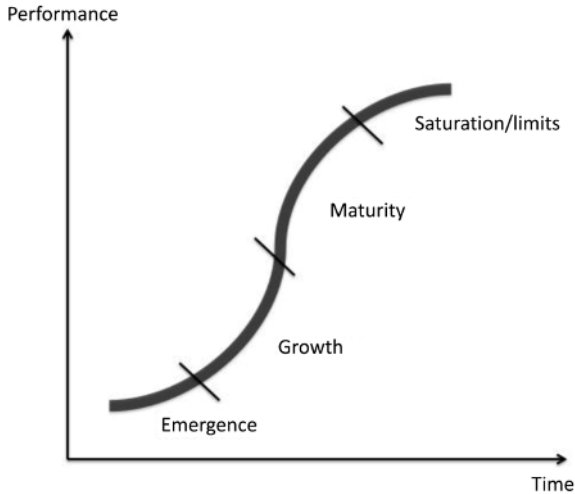


Figure 1.12. “S” curve explaining the different phases of technical development. Diagram inspired by Ait-El-Hadj (2009, p. 64)

1.8. Architects explore differently

The 1990s marked a turning point in the use of computers by architects, as attested by several historian-architect-researchers (Kipnis 1993; Picon 2011; Teyssot and Bernier-Lavigne 2012; Carpo 2013; Bourbonnais 2014). Different factors, again, intervened in this shift from design practices to exploratory practices. In the early 1990s, the computer was no longer used only by a minority of architects, but had become an increasingly common work tool, thanks in part to its size and accessible cost. In addition to this, the software offer was booming with, as we have just seen, a number of improvements in drawing, modeling and animation software. At the same time, a change in mentality was taking place with regard to these machines, whether it was with regard to the computer in the form of a PC, or even more so, with game consoles that had become fully integrated into the home. As the cybernetics

specialist Mathieu Triclot shows, the video game craze made it possible to introduce, after the arcade period, these machines into the intimacy of everyday life, thus allowing the adoption of “new relationships with the machine” (Triclot 2011, p. 15). This domestication of the computer, via video games, also brought a new regime of experience for users. This new type of relationship, according to Triclot, is clearly visible with the arrival of Pong. “The video game was until now the paradigm of a calculable, countable universe. Pong brings out the opposite power, of the loss of control and the loss of self in the game” (*Ibid.*, p. 141). It is not impossible to see in this a new disposition, a state of mind, which influenced the architects of the 1990s in their relationship with the different software. Indeed, the wave of architects using software no longer sought to rationalize the process and optimize structures, but on the contrary, they sought, via these new instruments, to perceive things that they had not believed possible before. This series of technical advances, as much as the new relationships established with the computer, found a point of convergence and expression with a particular “theoretical” context, a boiling point in certain architectural circles. Indeed, several radical investigations were led by a number of architects, as, for example, the research studies of *deconstruction* of the fundamental elements of the architecture (see *infra*. Chapters 2 and 3), or the stakes raised by the possibility of a generalized, omniscient cyberspace, where the architect would have a role to hold in its constitution-realization-actualization. These questionings of ontological order certainly encouraged this exploratory movement, of which the technologies of the computer constituted, among others, a device which allowed the foundations of architecture to be transgressed.

In this logic of rethinking the premises of architecture from top to bottom, through the renewal of processes, several architects showed a frank infatuation for variation. The explorations, carried out under these conditions, mobilized important theoretical paraphernalia, so as to seize, as precisely as possible, the implications that these technologies had on the ways of designing the object-project of architecture. These different theorizations, or research by concepts, which served both to grasp the specificities brought about by these simulation software and the driving force of exploration, led to a clear demarcation between the tool-makers, those who knew how to *build* software (or at least, who had a basic knowledge of programming) and the *tool-users*, those who *only* knew how to use it. Agitated and conflicting debates took place over a decade between these two groups, but eventually became obsolete, on the one hand, because of the evolution of the sensibilities of some architects confronted with the problems of manufacturing and, on the other hand, with the arrival of some software, plug in, that allowed for accessibility of programming environments (due to being visual).

1.8.1. *The craze for variation*

Among the different processes that software offered at the beginning of the 1990s, that of variation had a significant impact. Indeed, several expressions were put forward in order to capture the deformations–variations that the computer could generate. Whether these deformations are smooth, flexible or liquid, or whether they result from a process of folding, animation or digital simulations, they keep the variation of a form over time as a constant, bringing in its wake disturbances on space and on the objects-architecture. The latter became *Objectiles*, as announced by the architect Bernard Cache and Gilles Deleuze, with the observation that: “we are moving from the mold to modulation. We no longer apply a determined form to an inert material, but we decline the parameters of a surface with variable curvature” (Cache 1997, p. 67). This explanation was based on the Simondonian modulation, according to which:

actualization is almost instantaneous, there is never a stop for demolding, because the circulation of the energy carrier is equivalent to a *permanent demolding*; a modulator is a *continuous temporal mold* [...] To mold is to modulate definitively; to modulate is to mold continuously and perpetually variable (Simondon 2005, p. 47).

Some software contributed to making this change effective. As we have seen, it was a question of introducing time – a duration – via movement-variation which was different from that which was present in architecture until now: whether it was the movement-time perceptible *between* objects by the experience of a subject (such as a walk-through), or that of the building which moves itself. It was rather a question here of considering the building itself as a temporal object: of considering it as an organism that evolved. It was also under this logic that several architects had recourse to the notion of the virtual, as defined by Deleuze, from a reading of Bergson. By referring to Anne Sauvagnargues’ commentary on Deleuze’s philosophy, it became possible to understand the infatuation of certain architects with the movement-variation and their recourse to the virtual concept:

Movement does not consist in a displacement in space, but in a change in time, and such a change does not only affect the relation of the parts between them, but the whole in which they change. [...]. Finally, the concept of the virtual transforms our conception of the whole (Sauvagnargues 2010, pp. 109–110).

Beyond the terms used, these different programs allowed them to become aware of the implicit hierarchy that existed between the part and the whole – between form and substance – and, in a second step, invited them to change them. As Greg Lynn

said, it was not so much in the variations of the form themselves, that the changeover was achieved, but in the change of point of view on the background in which these variations took place. The architectural object became a temporal object when this changed its background more than its form: when its background was dealt with differently.

After this first wave of formal explorations, rather chaotic, or at least frenetic, a second wave appeared following the exhibition *Architectures non standard* held at the Centre Pompidou in 2003 and 2004. From this period, several architects present at the event became aware of two things. On the one hand, the importance of construction, of the materiality of architecture, which led to a phenomenal interest in the digital continuum (file to factory³⁰), and consequently in the software that made this continuum possible, and to another extent, in the CNC (Computer Numerical Control) manufacturing machines. On the other hand, the need to control processes, as the architect Lynn himself stated: “working forever on the basis of a few happy accidents does not work, because once you have produced something by chance, you must be able to reproduce it intentionally the next time” (Lynn 2003, p. 92). It was such admissions and comments that in some way indicated the criticisms that had been made by the tool-makers.

1.8.2. Tool-users and tool-makers

A clear opposition appeared at the turn of the century between tool-makers and tool-users. This opposition, which mainly sought to distinguish different levels of technical skills, could only be expressed by the fact that it became possible for a non-expert user to manipulate the sometimes complex instruments and obtain a result. This opposition also came in reaction to the place that theory had occupied in this appropriation of technology, inherited in large part from the deconstructivist movement. It was the relationship between practice and theory that had been destabilized. It was for that matter on this point that the interrogations of the Greek architect Kostas Terzidis rested. According to him, a truly digital theory could only emerge from a practice engaged with the “manufacturing” of tools, especially since at the time when the architect was making these criticisms, the architectural media scene was putting forward the opposite position, which consisted, still according to him, of covering practices with theory:

In contrast, the second category [tool-users] sought to connect humanistic philosophies with digital phenomena. In doing so it had to

30. What Bernard Cache showed during the exhibition based on his main idea of “associative architecture” and his use of TOP SOLID (see Cache 2003).

search for ideas or principles within the humanities that may explain or address digital phenomena. For instance, Lynn argues that the plasticity of computer generated forms may be associated with Deleuze's descriptions of smoothness and continuity, as if software is associated with softness. While this may hold some value at a phenomenal level, it certainly holds no truth at a mathematical level (Terzidis 2006, p. 56).

We see here that Terzidis's criticisms do not only concern the level of skill and mastery of computer processes, but the way in which the problem is approached: on the visible phenomena or on the procedural and mathematical enunciation. It is hardly surprising to find again one of the fundamental distinctions that was set up at the beginning of CAD, between those who approach the design problem from phenomena, interacting directly with a form on a screen, and those who, rather, approach the question from enunciations. The fact that this distinction is maintained and claimed at different times invites us to take a moment to find the basis of this differentiation. This distinction can be pursued to find a more ancient question, in philosophy, between the continuous and the discrete. This question had moreover been forcefully revived with the arrival of the computer. As Gérard Chazal reminded us in his introduction to John von Neumann's text on automata:

The existence of the continuous is not in doubt, at least in the way we perceive the world; at stake is the possibility to say it, to describe it and to operate with it by means of discrete realities. To want to give an account of continuous phenomenon with the help of discrete measures, it is to think to be able to solve it by the analysis (Chazal 2004, p. 39).

As mentioned in the introduction to this chapter, our objective is not to favor one point of view over the other, nor to seek to rebalance the story, but rather to contextualize the postures adopted by the architects; to try to understand their motivations and intentions.

This detour allows us to insist on the fact that it is a question of approach, of posture and of commitment towards technologies. As such, it is not uninteresting to return to the distinction that Gilbert Simondon had established as for the different modes of relation that it is possible to maintain with the technical object. Simondon distinguished mainly two: a status of majority and a status of minority (Simondon 2001, p. 85). The philosopher developed this distinction between a *learned relationship in a child* and a *thought relationship in an adult*; it also declined under the figures of the craftsman and the engineer. The child, as the craftsman, maintains

a particular relationship with the object that we can think in terms of attention to the global and less to the particular. The child listens to a story and remains attentive, even if they do not understand all the details and even if they lose bits of it, they nevertheless grasp the essentials; they function in a syncretic way. In comparison, the adult, with a relationship to the object similar to that of the engineer, measures, analyzes and thinks; they understand the logic of the object and know the logic of the whole.

This distinction is interesting in several respects. On the one hand, these positions, with their limits, make it possible to insist on two ways of approaching and integrating digital technologies into practices, as well as two distinct learning modes, which lead to two acquisition regimes. It is then not superfluous to say that these types of relationship can have an impact on the evolution of a software program, either by privileging one mode, to the detriment of the other, or on the contrary, by seeking to combine the two. This attempt to combine these two positions can approach the possible reconciliation that Simondon had raised, between these two statuses, by a median way, possibly thanks to an encyclopedic spirit that would manage to seize their common nature, by the capacity to acquire a “technological conscience” (Simondon 2001, p. 94).

1.8.3. *The connection with the construction (CAD-CAM)*

As we have tried to show in this section, the decade of the 1990s led to a series of architects turning to software not intended, a priori, for the field of architecture, such as those offered by the market. Like the architects who were fascinated by software from the cinema and animation fields, some architects also sought alternative paths, outside of drawing and modeling. A few architects dared to use a sophisticated software to facilitate the connection with construction: known as CAD-CAM. Some of these architects maintained a singular and close collaboration with software companies. These collaborations offered a dual adaptation. On the one hand, the architect experienced the technical and mathematical logic of the software they used for their design and manufacturing and thus had to adjust their methods, and, on the other hand, the companies could closely observe the logic of use of some of their privileged users and thus respond to their particular needs. While this kind of collaboration was relatively rare in the 1990s, it has been widely developed recently, and architects and researchers are now called upon to participate in the development of some functionalities for some software. We will present here two of these pioneering collaborations in this search for a software adapted to respond to this idea of file to factory.

The now world-famous architect Frank O. Gehry was among the first, in the early 1990s, to turn to software “outside” architecture that allowed him to manage the mathematical complexity of the curvilinear forms he wished to achieve for his buildings, such as the Guggenheim Museum in Bilbao (1991–1997). As Cardoso Llach points out, Gehry is certainly part of a group of architects whose work focuses on questions of materiality, its expressiveness and, more broadly, of tectonics. The use of powerful software is not simply a way to explore form but, as the researcher-architect states, to document explorations of matter. “The chief purpose of software in Gehry’s practice is thus documenting and evaluating technical aspects of designs informed on material explorations” (Cardoso Llach 2015, p. 95). It is with similar concerns that Gehry resorted to CATIA software, initially intended for aeronautics. Created in 1981, Dassault Systèmes is an extension of Dassault Aviation, and thus inherits the expertise acquired for the design of aircraft and more particularly on the aerodynamic optimization and the issue of smoothing curves.

CATIA³¹ inherited a series of technical innovations that can be grouped into two branches: one linked to 3D with GEOVA, which was developed internally for almost 10 years, and another linked to 2D drawing and manufacturing, with CADAM (Computer Augmented Design and Manufacturing), a 2D drawing software developed by Lockheed in the USA, developed under the name DRAPO (Design and Production of Aircraft by Computer). It was the combination of these two branches that led to the creation of CATIA in 1981. This journey shows above all the expertise acquired by Dassault Systèmes in the way of managing, calculating and organizing complex shapes, as well as their ability to machine them. It also shows why architects like Gehry would be interested in this kind of know-how “encapsulated” in software. This fortunate association led to the creation of Gehry Technologies (GT) and the *Digital Project* (2003), a software program tailored to a type of architectural practice that is becoming increasingly popular. In light of the few interviews the researcher-architect Cardoso Llach has conducted with some of the members of GT, a gap remains between the software company and design team. “[The] separation between GP [Gehry Partners] and GT continues to inscribe a kind of compromise protecting the social role of designers as authors while allowing technicians the freedom to market their expertise” (*Ibid.*, p. 98). While Gehry has sought to bring architectural projects and technologies together to an uncommon degree, these two entities will nevertheless remain autonomous, following their own evolution.

Without going into such detail, let us just say that other architects will be interested in software “outside” architecture, and that they will maintain similar

31. CATI release 1.0 came out in 1977 and became CATIA in 1981.

collaborations, such as the fruitful association of architect Bernard Cache and his Objectile group, who participated in the development of the TOP SOLID software by the Missler brothers, in the 1990s (see *below*, section 3.3.2). It was thanks to this software that they were able to achieve the “associativity” which he advocated. While this associativity was based on a software capable of following the whole process – “from the first hypotheses of design to the piloting of the machines” (Cache 2003, p. 43), Cache as much as Gehry showed that it is important to change the very logics of design; these logics must adapt to the logics of the materials and their implementation.

While the relationship to CNC machines was not the same for these two architects, there is nevertheless a similarity in the transformation of design processes in order to facilitate the connection with construction. It was obviously no longer a question of drawing lines and points, nor only of thinking by objects, but of designing from objects in relation. While this way of doing things had been thought of since Eastman’s BDS project – to reduce the gap between design and construction – the computers of the 1990s allowed this type of process to find its real efficiency and operability.

1.9. A new bifurcation

The explorations of the 1990s and 2000s were a media craze; they came to occupy many debates on the architectural scene and have been considered by some as a turning point (Carpo 2013). Whether it was the “calculated variations” (Lynn 2003) thanks to animation software or the CAD/CAM craze, the last few years have shown a new bifurcation that, paradoxically, has brought many of the debates back to where they were left before the 1990s–2000s turning point; in other words, design from a 3-D model and ease of use. On closer inspection, this new bifurcation is not really a bifurcation, as advances in these themes have continued, only to be overshadowed from the media stage by the architects who were to revolutionize the “whole” of architecture.

It is important to remember that the turn of the millennium corresponded, for many architectural firms in English-speaking countries, to the moment when the use of computers became widespread and strongly established in practices. As the architect and software specialist Pierluigi Serraino points out: “As of 2003, the sight of a drafting board in an office is often interpreted either as the tangible remainder of an obsolete mode of production or a nostalgic holding on to a bygone approach to the making of architecture” (Serraino 2003, p. 187). The all-digital nature of the architectural community seems to be in place, although it should be noted that “digital” does not necessarily imply advanced technological awareness. This change

in distribution has made it possible to ensure a certain security for the markets and has allowed the major players to develop certain aspects of their products, such as their stability and performance, as well as to consolidate their offers with a particular interest in “more user-friendly” logics. This user-friendliness was generally based on an intuitive interface, offering easy access, execution and handling. It also enabled the clientele to increase to a larger and larger public, of non-experts. This idea of offering easy-to-use software has also led some companies to offer alternatives to powerful software, but sometimes difficult to access for the uninitiated. This “other lineage” has made its way, without making too much noise, in parallel to the aforementioned ones, but has inserted itself into several environments in a massive way, including that of architecture schools (Lescop and Chamel 2020) and those of small- and medium-sized agencies (Stals 2021). This posture, contrary to those exposed in section 1.8, no longer pretends to transform the ways of designing, but on the contrary, seeks to get as close as possible to them in order to reach the greatest number.

We will end this chapter by pointing out different trends that have been guiding software development for the last 15 years and their repercussions on many practices.

1.9.1. *The end of unique software*

At the turn of the 2000s, when the exploratory period was at its peak, several commentators, historians and architects expressed fears about this enthusiasm for specialized software. While, as we have seen, several objections were based on the values of the drawing, part of the concerns also resided in the fear of seeing a homogenization of the architectural expression according to the software used. This fear was based on the idea that software, because of its internal programming, contained a predetermination that would inevitably affect the form of the building. The formula was launched: *Form Follows Software*³². It was the architect and software specialist Pierluigi Serraino who put it forward in a study of the use of 3D modelers in six American agencies. Serraino’s starting hypothesis is clearly formulated in the title of his article. More precisely:

Morphological manipulations and their resulting forms can hardly be separated from the software that defined them. The construction of the graphical interface, the arrangement of the modeling and rendering icons, and the organization of interrelated menus structure the task environment and the input of topological data. As a result, their digital

32. For architects, this formulation is now unknown. It references “form follows function”.

outcomes are bracketed in geometrically identifiable limits and bring about strongly patterned sets of architectural propositions. It follows that the choice of software is a primary, if not the utmost variable, bearing heavily on digital design (Serraino 2003, p. 187).

In his study, Serraino presents three 3D modelers (Form-Z, 3D-Studio Max and Maya), raising the presuppositions they contain and what type of practice they promote. This presupposition was based on the fact that certain commands were easier to perform with one program than with another. Serraino showed, through an example, that the same operation (extrusion, for example) could be done in two steps in Form-Z, whereas it required six steps in 3D-max. This type of demonstration made it possible to establish his reasoning that each software program, because of its operation, recommends certain ways of doing things. Despite his effort to define the specificities of each modeler, as well as the scope of his study to six agencies, Serraino ends his article by stating the impossibility of validating his hypothesis, without additional case studies and quantitative tests. After his study, it is impossible for him to “further corroborate the statement that as of today form follows software” (*Ibid.*, p. 204). By observing the six practices analyzed, it is indeed possible to identify “strengths” for each of the software, but it is by no means possible to affirm that these have repercussions on the form of the buildings generated. Indeed, the analysis tends to show that the architects, being aware of and attentive to the specificities of each software, navigate from one software to another according to their needs, or even use them in an “unconventional” way, and therefore, in no case is it the software that “dictates” their way of doing things, and even less so the shapes that result from it. Instead of confirming its initial hypothesis, this study shows that the architects’ understanding of the software allows them to move away from this predetermination.

This type of practice, fragmented, passing from one piece of software to another, seems to have been integrated into several architects’ practices, as the study by Georges Teyssot and Olivier Jacques on the practices of the architect Jürgen Mayer also shows. This investigation confirms that architects no longer limit themselves to a single software, but use several, switching from one to another depending on the task at hand. Here, Mayer’s team relied on the level of approximation offered by each piece of software to make their choice:

As in most of J. Mayer H.’s projects, three software packages were used to handle various levels of approximation: Maya (Autodesk), Rhinoceros 3D (Robert McNeel & Associates), and ArchiCAD (Graphisoft). In order to translate the programmatic sketch into a sculptural configuration, Maya computer graphics software was used.

This model was then imported into Rhino, because of the excellent precision in the definition of the curves. At the end, to provide the documents to the different trades, but also to parameterize the sunshades on the product banks entered in BIM, the models were reproduced in ArchiCAD (Unser in Teyssot and Jacques (2009, p. 111)).

This fragmentation of practices, by the passage from one software to another, is obviously based on the *technological awareness* of architects, capable of evaluating the relevance of one logic in relation to another, as well as, it goes without saying, that these passages are facilitated (or not) by the compatibility (or not) of computer languages. It is obvious that the degrees of openness and closure of software contribute to this breakdown, and above all, contribute in a certain way to this awareness³³ among users.

1.9.2. “More user-friendly”

Illich (1973) develops the notion of conviviality in his book, with the objective of proposing a new articulation between humans, tools and society, so that individuals reach the perfect mastery of their tools in order to integrate them into the community. As the philosopher of technology Carl Mitcham, a great reader of Illich, points out, this argument is based, among other things, on the separation between “the way to make things work” and “the way things work”. The important thing is not that an individual can make an object work, but rather that they can understand how it works: that they possess neither more nor less a *technological consciousness*. If Mitcham, following Illich, calls for a rethinking of teaching and education, and even of society, in order to respond to the “current multidimensional crisis”, he nevertheless raises the difficulty of considering the interface, and our relationship with it, with the term conviviality. Referring to Illich, Mitcham noted:

Even when we learn “how to operate the TV or the telephone... their workings are hidden” from us. By contrast, from personal trial-and-error engagement it is possible to learn not only how to use a convivial tool, but how it works. What are today called “*user-friendly* interfaces” in no way address the disparity Illich identifies (Mitcham 1991, p. 28).

33. It also seems possible to note a heightened awareness of architects using free software. An in-depth study should be carried out to confirm this hypothesis.

While the general thesis defended by Illich can be summarized as follows: “technical objects must be made for man and not enslave him” (Simondon 2014, p. 453), it is not certain, on the other hand, that the trends and logics behind the more user-friendly respond exactly to this societal project. It is in this sense that the notion of “usability”, first developed in the fields of engineering in the mid-1990s by Nielsen and Norman, seems to better capture the questioning of the logic behind the programs: their effectiveness and efficiency. Usability thus insists on the comfort of the interface, its manageability as well as its simplicity of use.

One of the objectives behind this usability is also to acquire a new, non-expert public. However, it is not at all obvious to link this objective of making software accessible to as many people as possible via the more-user-friendly approach to the makers’ movement of the 2000s, nor to participatory architecture as we have seen it with the expert systems of pioneers such as Negroponte or Yona Friedman, or even self-construction. Rather, we can say that for the usability of the interface, several software companies have sought to adapt their offer to the demands and needs formulated by the market.

A whole series of devices have been put in place to support users so that they can make their software work properly: from the instruction manuals provided with the purchase of software products in the early 1980s to the discussion forums or community organizations available online, via the magazines published by software companies at the turn of the 1990s. It is obvious that the Web has brought these two entities closer together, but without really “opening” the software to its users. The possibility of sharing a Beta version with a restricted and sorted public offered a greater porosity. The user, thanks to their feedback, could participate in improving certain aspects, functionalities or correcting certain malfunctions. These different features can be seen as having the task of improving the *usability* of the interface and making it *friendlier*. It is thanks to this back and forth, despite the good intentions of the companies to adapt their product to the users’ requirements, that we note the most frank opposition with the “open source” culture. This open culture does not only claim an understanding of the workings, but an “opening” of the software itself, that is, a free access to the codes that constitute it. This question would deserve a much more thorough analysis, especially since this opposition – open-closed – seems to raise Illich’s question on the type of relationship favored between technologies and community again.

1.9.3. *Restoring the simplicity of use*

Over the past 20 years, software such as *SketchUp* has flourished in various architectural circles, both at the professional and academic levels. This success can

certainly be attributed to its ease of use, based on intuitive drawing techniques, as Brad Schell, one of its founders, stated. “*SketchUp* had a simple interface that captured the ‘spirit of drawing’, which made it easy for amateurs to use”.

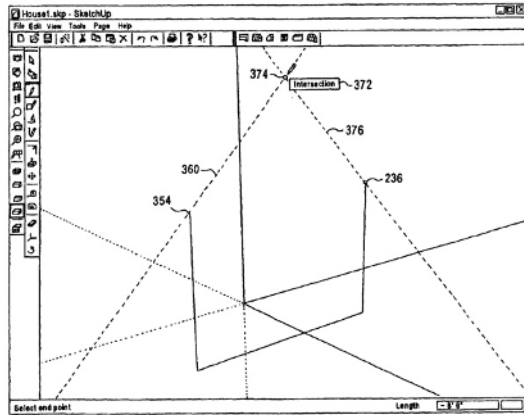
SketchUp was developed by the company @last with Brad Schell and Joe Esch as its founders. Schell, a free spirit, co-founded *CadZooks* in 1989, a small software company whose main goal had been to simplify the making of complex construction drawings, through the automation of certain tasks that were previously performed by skilled draftsmen capable of 3D thinking. *CadZooks* was acquired by Autodesk in 1996. Schell then looked for new opportunities. He then partnered with Joe Esch, a software developer, to create a new company in 1999. According to journalist Eric Gershon, “Schell shared some notes and sketches he’d made over the years for easy-to-use 3D modeling software. Esch liked them, and they founded @Last in an office at 10th and Pearl in Boulder” (Gershon 2018). As we have seen with Form-Z, among others, volumetric modeling software was for the most part opposed to drawing techniques: it operated under a different logic; it was no longer about drawing lines, but defining volumes. *SketchUp* was built on this distinction. And, despite the different versions – Pro, Free, Shop and Make – SketchUp remains above all a 3D tool that reproduces the “feel and freedom” of drawing. Their site, before the takeover by Google in 2006, read:

We envisioned: A software program that would allow design professionals to draw the way they want by emulating the feel and freedom of working with pen and paper in a simple and elegant interface. A very powerful software program that would be fun to use and easy to learn. Computer technology that could be used during the design exploration process and would allow designers to play with their designs in a way that is not possible with current design software³⁴.

The idea of this software was to propose an alternative to the 3D applications present on the market such as *3D Studio Max* or *Maya*, but “whose mastery requires a significant amount of time” (*Ibid.*). They decided to make it easy to use by returning to the logic of drawing, with a fun interface that the designer could play with.

34. The @last site disappeared several years ago, but it is possible to find parts of it on the IMVU site, including this quote. The IMVU page also disappeared in 2010, but it remains accessible thanks to Web.Archives.

FIG. 3K



U.S. Patent
Sep. 30, 2003
Sheet 19 of 159
US 6,628,279 B1

Figure 1.13. Figure 3K from the U.S. Patent 6,628,278, which shows how to “build” your object according to a drawing logic, by drawing lines

After a few years of successful activity, *SketchUp* was bought by Google in 2006³⁵. This transaction was symptomatic of a more important transformation in the public to which this software was addressed. It was not impossible that the culture of the amateur, the DIY (Do It Yourself), as much as their revival at the turn of the century³⁶, contributed to this rise of the kind of practice where the privileged public are no longer, a priori, designers-architects, but creative individuals wishing to renovate their kitchen, design storage spaces or fit out an extra room in their second home.

A new market³⁷ has opened up, with various applications available online, which offer easy-to-use design environments with vast libraries of objects to insert. These applications, initially modest and also simplified in order to function correctly online, have undergone real technical improvements in recent years, combining stability and performance. Some applications are specialized and offer, thanks to

35. *SketchUp* was bought again in 2012 by Trimble, an American company specializing in geo-spatial software (according to Wikipedia).

36. As an indication, it was in 1998 that Neil Gershenfeld began his course “How to do (almost) everything (or anything)” at MIT.

37. Among the first, the free software *Sweet Home 3D*, by e teks in 2006; then Autodesk Labs offered *Home Styler* in 2009; then *Space Designer 3D*, from the Asynth SAS group in 2010; and *Homebyme* by Dassault systèmes in 2011; furthermore, the application *Planner 5D* (2011).

different integrated devices – dashboard – expertise on a particular criterion, such as thermal or light analysis and simulation. Others offer virtual reality experiences, thanks to immersive headsets. Faced with this craze, the majors have also seized this market, creating their own online application, or simply, acquiring an existing one that was working well.

1.9.4. Manipulating parameters

As mentioned above, the issue of controlling digital design processes became central around 2005. It is important to remember that during the period 1990–2005, it was not easy to control these processes by writing algorithms and that only a handful of architects could claim to be tool-makers. Software like Processing (2001), by artists Benjamin Fry and Casey Reas, offered more accessible programming environments for non-computer scientists, but for this type of software, it was still a matter of learning a language and writing the instructions that the user wished to see executed through lines of code. It was not until a few years later that it became possible for architects to order and manage the variation functions through a “visual” programming environment, thanks to what was called parametric design, which made up for this lack of process governance. Parametric design really took off during this period, even leading some architects such as Mark Burry to affirm: “My contention, then, is that the new generation of designers are required to be versed in at least one of these techniques – parametric design and design computer programming – as a new definition of design skill essential for full participation in today’s environment.” (Burry quoted by Goulthorpe (2007, p. 80)).

Faced with this interest in a better control of processes, a better understanding of the logic behind them, an effort has been made by the software companies themselves to adapt their offers by proposing digital environments that allow access to higher levels of these processes, so as to manipulate their very logic. Bentley’s *Generative Component* program (2003) and McNeel & Associates’ *Grasshopper* plug-in (2007) are archetypal examples of this type of program, which allow the manipulation of both parameters and functions, as well as their different arrangements and relationships. These parameters and functions can be varied: from structural data to sociological data to morphological data. The architect Patrick Schumacher advanced the term *parametricism* to define architecture realized according to these principles (Schumacher 2009).

Beyond the repercussions on the designed and/or built objects, this type of practice, which manages and organizes the relations between functions and parameters, approaches a programming thought, or at least, obliges the designers to deconstruct, in an analytical way, their design process, according to intentions that

they must explicitly state. However, this enunciation is no longer done by writing algorithms, nor by direct interaction with the form, but through an in-between, which is located in a new window and which is situated between the geometric representation of the object and the code that allows it to be generated. The changes made to the parameters, functions and their relationships are reflected in the digital model, which is thus modified. The added value lies in the fact that the programming environment is visual, that is, algorithms are encapsulated in functions that must be connected to each other. It will be thanks to their combination with a traditional modeler, for example, Rhino for Grasshopper, that it will become interesting to simulate different variations, on a precise parameter, exploring, quickly, a great number of different scenarios. This type of combination has led to a series of functions such as *run*, which executes an encapsulated function, or *bake*, which “fixes” the variations and thus, find a model.

Thanks to this visual programming environment, software companies have found an efficient way for architects to deepen their understanding of processes without going through the arduous task of learning a programming language. It is not uncommon to see some of these users learning programming afterwards. This in-between has introduced a new way of apprehending the design of a project and of engaging it under new logics, combining the divergent positions exposed above.

1.9.5. BIM qualified practices

The objectives of BIM practices are based on the technical capabilities of certain software programs, but go far beyond them, in several aspects: whether the impact on the different levels of organization of the project as much as the application of standards, via the data associated with a digital model. While several authors (among others: Cardoso Llach 2015; Cristia 2020; de Boissieu 2021) have raised the many implications that this BIM model has on practices, around surveys carried out in the context of concrete projects, our observation here will focus on some elements in particular, those that closely affect the capabilities of the software and its use. As this historical overview has sought to expose, contemporary technologies and practices do not appear suddenly, but inherit many characteristics, both technical and socio-cultural, that are possible to identify. For example, it is not difficult to draw links between certain functionalities of contemporary BIM technologies and the intentions expressed in the 1970s, for example, with the GLIDE language or the BDS (Building Description System). It is true that BIM is based on data management, thanks to the description of the objects that make up the building, but this information has been transformed in recent years. This data has become massive, in relation to the objects used in the model, their precision and

complexity, as well as since these objects are connected to construction companies, through different platforms, such as BIMObject (or formerly Polantis, which was acquired in 2019 by BIMObject), where it is now possible to access numerous and impressive catalogs. It is in this logic, where it becomes possible to integrate “real” data into the project, that companies support reducing the gap between design and construction and thus offer an advantage for the realization of the building.

This possibility of sharing a large amount of information, via a data flow, has known intermediary devices before finding the form of collaborative platforms, such as the “plan cabinet”, which constituted a sort of analogue to its physical device: a place that contained the plans of a project and was accessible to the different stakeholders of the project. The example of the Stade de France, at the beginning of the 1990s, allows us to show the kind of difficulties that the idea of sharing and conservation could entail at the time. There were, for this particular project, more than 6,000 plans stored in a “plan cabinet”, which was in fact a Prosis system, developed by the company Serete. It was a question of “making the computer files readable, and in the same form for everyone” (Zublena and Duplay 2000, p. 28). The plan cabinet was, however, limited in memory and could not manage “big” files of 15 or 20 Mb. In addition, there were compatibility problems, as explained by one of these architects where, for another project, the team had to use *MicroCADAM*. This software was not used by the architectural team, but one of the consulting firms was only equipped with this software, so the architects had to comply with this constraint in order to ensure the consistency of file sharing.

It also seems possible to establish a link with the massive arrival of high-performance 3D modelers in the early 1990s, which in a way legitimized this digital model that has become central to these BIM processes. This 3D model is now at the center of interactions between the various actors of a project, as well as at the center of the system of representation of the architect. As Cardoso Llach points out, this system is a kind of distortion where, due to the skills of the different actors of a project, the project is first “drawn” using sketches and plans that are then transferred to a centralized digital 3D-model where, once the design is completed, the model is used to produce the technical drawings of execution used for construction. This back and forth process shows precisely, by the disturbances it causes, the extent of the restructuring that this type of practice wishes to bring about, as much as its harshness. This “forced” centrality rests, as Cardoso Llach underlines, on the ideology that the industry conveys, which has implications on the actors, and their relations of force.

These practices called BIM inherit a wide range of technological advances, built up over the years, with the particularity of seeking to reconcile design and

construction. The imposing technical and normative devices contained in the so-called BIM software, as well as the ideas of centrality that are attached to it, have, however, led to a technology that is heavy to handle. As it will be possible to see in Chapter 5, or in the work of Cardoso Llach (2015), there are ways to make this model flexible by offering the possibility, for some architectural elements, parts of buildings, to integrate more exploratory design processes and be governed by other methods that are less restrictive.

1.10. Conclusion

This evolution, outlined in broad strokes and far from being exhaustive, has sought to present some benchmarks in order to raise some of the striking trends, as well as their articulations between them, in order to give meaning to this appropriation of digital technologies by architects. Following this presentation, the correlation between the technical logic and the issues related to design practices seems undeniable, even if this co-constitution is never clearly fixed once and for all. The important thing is to remember that these stakes and practices have always been strongly associated with actors-milieux and that the latter are undergoing constant transformations. At first, it was possible to identify a few changes in the actors involved in the production–distribution–maintenance of software, including a major one around the beginning of the 1980s. If the pioneers involved with software development were generally linked to research institutions, there was an important change with software companies that, on the one hand, came to gradually take over the market and, on the other hand, forced these architect-researchers to find new places of expression and research. Secondly, we realize that the users and defenders of software – architects, for this study – have never been the same, but, as we wished to expose, different postures have been identified, sometimes repeating themselves, yet never identical. It remains surprising to find on many occasions, with some variations, the initial distinction that appeared in the CAD project of 1959, between, on the one hand, those who militate for the enunciation of a clear interaction carrying information and, on the other hand, those who advocate a fluid and efficient interaction. If the experiments of the 1990s, as well as the simplifications of use, suggested a real ascendancy for the interaction with a form-model on a screen, the explorations made from the GAN (Generative Adversarial Network) or the DDD (Data Driven Design, see *infra*. Chapter 5) again show a possible reversal.

These numerous reversals, passing from one posture to another, have nevertheless shown the prevalence of certain conceptual and mental schemes, both

in the type of task to be accomplished with the computer and the type of relationship to be maintained with it. We note with a slight astonishment the return of the same themes, the same problems which reappear with, however, a technological equipment so different that it remains complicated to establish continuous and rectilinear links between them. It therefore seems important to us to insist on the non-continuity of this software evolution. We wanted to capture, each time, the concerns that were at the source of these developments, in their dynamic aspect. As if everything had to be started again, each time, from the new equipment – new socio-technological infrastructures – in which these software programs appear and are inserted. But this renewal is never complete; it always remains a residue of the previous experiences that have been kept in the techniques themselves. It is on these remains, we could say, which are maintained from an experience to the other, from an epoch to the other, that we can perceive the “real” technological advances of the software. Simondon used the notion of technicity:

Now, [...] technicality must never be considered as an isolated reality, but as part of a system. It is a partial reality and a transitory reality, both the result and principle of genesis. Being the result of an evolution, technicity is also the *depository of a capacity to evolve*, precisely because, as solution to a first problem, it has the capacity to be a mediation between man and the world. (Simondon (2001, p. 157), our use of emphasis).

It is on this evolving power of technicality that the capacity of architects to individuate themselves, to individuate their practice, rests: *to individuate themselves through their practice*. This leads us to believe that technologies affect sensibilities in an individual way, and not so much in a general way. We could even argue that, at first, technologies do not make us *more human* than *more individual*. If these technical individualizations of practices have inevitably had repercussions on the ways of doing architecture and on the architecture that is built, the question is whether we, as Yuk Hui suggests with his essay on *cosmotechnics*, could seek to re-engage this question of technique, on the scale of the instruments of architectural design, in such a way as to discover what these instruments unconsciously express of the ways of thinking of architects.

1.11. References

Aït-El-Hadj, S. (2009). Les apports de la “théorie de l’objet technique” de Gilbert Simondon à une théorie opératoire unifiée des systèmes techniques. In *Les systèmes techniques, lois d’évolution et méthodologies de conception*, Aït-El-Hadj, S. and Boly, V. (eds). Hermes Lavoisier, Paris.

- Alain, (1983). *Le système des beaux-arts*. Gallimard, Paris [Original published 1926].
- Alexander, C. (1971). *De la synthèse de la forme. Essai*. Dunod, Paris [Original published 1964].
- Anderson, S. (1966). Problem-solving and problem-worrying. *Conference given at the A.A. School of Architecture*, London.
- Apple Computer, Inc. (1989). Apple Engineering/Scientific Solution Guide. The Compag Company, San Francisco.
- Archigram (1968). Archigram #8. *Archigram*, London.
- Arnold, M. (1988). L'image de synthèse animée. *Communication et langages*, 77(3), 58–81.
- Bachelard, G. (1934). *Le nouvel esprit scientifique*. PUF, Paris.
- Banham, R. (1968). The triumph of software. *New Society*, 12(318), 629–630.
- Bergson, H. (1998). *Creative Evolution*. Dover Publications Inc, New York [Original published 1917].
- Bernholtz, A. and Bierstone, E. (1966). Computer-augmented design. *Design Quarterly*, 66/67, 40–51.
- Berthelot, M. (1974). Langage formel en allocation spatiale un exemple : Fosplan. *Bulletin GAMSAU*, 3(1), 27–43.
- Bird, L. and LaBelle, G. (2010). Re-animating Greg Lynn's embryological house: A case study in digital preservation. *Leonardo*, 43(3), 243–249.
- de Boissieu, A. (2021). Architecture et pratiques orientées sur les données : pour un décloisonnement du BIM et du design computationnel. In *Anticrise Architecturale ?*, Claeys, D. (ed.). LOCI, Louvain.
- Bojár, G. (2005). *The Graphisoft Story. Hungarian Perestroika from an Entrepreneur's Perspective*. Manager Kiadó, Budapest.
- Bourbonnais, S. (2014). Sensibilités technologiques : expérimentations en architecture numérique 1987–2010. Thesis, Paris-Est and Université Laval.
- Bourbonnais, S. (2020). Quand l'ordinateur devient générateur de forme. Retour sur l'expérience du Fractal Studio. In *Actes du 9^e SCAN Culture numérique et conception architecturale... Retour vers le futur*, Derycke, D., Martin, G., Ben Rajeb, S. (eds). EDP Sciences, Les Ulis.
- Bourbonnais, S. (2021a). Rétroaction dans les processus numériques de conception. *Netcom* [Online]. Available at: <http://journals.openedition.org/netcom/5426> [Accessed 15 May 2021].
- Bourbonnais, S. (2021b). Évolution des milieux de la fabrication numérique. In *Anticrise architecturale ?* Claeys, D. (ed.). LOCI, Louvain.
- Bret, M. (1984). L'image numérique animée. Thesis, Paris 8.

- Burnham, J. (1970). Notes on art and information processing. Software Information Technology: Its New Meaning for Art. Exhibit, Jewish Museum, New York, 10–14.
- Burry, M. (2007). Towards a new praxis. In *dECOi Architects – From Autoplastic to Alloplastic*, Goulthorpe, M (ed.). Hyx, Orléans.
- Cache, B. (2003). Vers une architecture associative. In *Architecture Non Standard*, Migayrou, F. and Mennan, Z. (eds). Centre Pompidou, Paris.
- Campion, D. (1968). *Computer in Architectural Design*. Elsevier Publishing Company, London.
- Cardoso Llach, D. (2015). *Builders of the Vision. Software and the Imagination of Design*. Routledge, New York.
- Carmo, M. (2013). *The Digital Turn in Architecture, 1992–2012*. Architectural Design Reader, John Wiley & Sons, New York.
- Carrozzini, G. (2009). Gilbert Simondon et Jacques Lafitte : les deux discours de la “culture technique”. In *Cahiers Simondon – Numéro 1*, Barthélémy, J-H. (ed.). Harmattan, Paris.
- Çelik Alexander, Z. and May, J. (2020). *Design Technics: Archaeologies of Architectural Practice*. University of Minnesota Press, Minneapolis, MN.
- Chazal, G. (1996). La pensée et les machines : le mécanisme algorithmique de John von Neumann. In *Théorie générale et logique des automates*, von Neumann, J. (ed.). Champ Vallon, Seyss.
- Ciaravino, (2004). *Un art paradoxal. La notion de disegno en Italie (XVe–XVI siècles)*. Harmattan, Paris.
- Coll. (1971). LOKAT. *Bulletin GAMS AU*, 1(2–3), 16–20.
- Coll. (1972). DOMINO : un programme d’aide à la conception. *Bulletin GAMS AU*, 2(4), 25–37.
- Coons, S. (1966). Computer-aided design. *Design Quarterly*, 66/67, 6–13.
- Coons, S. (1967). Surfaces for computer-aided design of space forms. Technical report MAC-TR-41, MIT, Cambridge, MA.
- Courtieux, G. (1971). Récents développements dans les applications des ordinateurs à la construction civile, à l’architecture et à l’urbanisme. *Techniques et Architecture, special edition: Informatique et architecture*, 33(4), 48–49.
- Cristia, E. (2020). Projet architectural et maquette numérique BIM : enquête ethnographique au sein d’un grand projet de construction. Thesis, École doctorale Abbé-Grégoire, Paris.
- Crotty, R. (2012). *The Impact of Building Information Modelling: Transforming Construction*. Routledge, London.
- Doyle, M.R. and Alvarez-Marin, D. (2020). When tools become instruments: Introduction to the special issue. *Contour*, 5, 2–5. EPFL, Lausanne.

- Eastman, C.M. (1987). Fundamental problems in the development of computer-based architectural design models. In *Principles of Computer-Aided Design: Computability of Design*, Kalay, Y.K. (ed.). John Wiley & Sons, New York.
- Eastman, C.M. and Henrion, M. (1977). Glide: A language for design information systems. *Siggraph*, 24–33.
- Eastman, C.M. and Yessios, C.I. (1972). An efficient algorithm for finding the union, intersection and differences of spatial domains. Technical report, Computer Science Department, Carnegie Mellon University, Pittsburgh, PA.
- Eloy, S. and Duarte, J.P. (2014). Inferring a shape grammar: Translating designer’s knowledge. *Artificial Intelligence for Engineering Design Analysis and Manufacturing*, 28, 153–168.
- Emmerich, D.G. (1974). *Soft Architecture : essai sur l’autoconstruction*. Institut de l’environnement, Paris.
- Estevez, D. (2001). *Dessins d’architecture et infographie : l’évolution contemporaine des pratiques graphiques*. CNRS, Paris.
- Fankhänel, T. and Lepik, A. (2020). *The Architecture Machine. The Role of Computers in Architecture*. Birkhäuser, Basel.
- Gershon, E. (2018). Last: How Google came to Boulder. *Coloradan: Alumni Magazine* [Online]. Available at: <https://www.colorado.edu/coloradan/2018/03/01/last-how-google-came-boulder> [Accessed 12 February 2021].
- Glueck, G. (1970). Jewish museum’s “software” confusing. *New York Times* [Online]. Available at: <https://www.nytimes.com/1970/09/26/archives/jewish-museums-software-confusing.html> [Accessed 29 March 2021].
- Goulthorpe, M. (2007). *dECOi Architects – From Autoplastic to Alloplastic*. Hyx, Orléans.
- Guchet, X. (2005). *Les sens de l’évolution technique*. Léo Scheer, Clamecy.
- Guéna, F. and Zreik, K. (1988). *Systèmes experts dans l’architecture, le bâtiment et les travaux publics. Étude de cas : la France, le Japon, les États-Unis et le Royaume-Uni*. CIMA, Ministère de l’équipement et du logement, Paris.
- Guffroy, Y. and Bontems, V. (2018). La mécanologie : une lignée technologique francophone ? *Artefact. Techniques, histoire et sciences humaines* [Online]. Available at: <http://journals.openedition.org/artefact/2218> [Accessed 5 January 2021].
- Hui, Y. (2021). *La question de la technique en Chine : essai de cosmotechnique*. Divergences, Paris.
- Hurst, J., Mhoney, M.S., Taylor, N.H., Ross, D.T., Fano, R.M. (1989). Retrospectives I: The early years in computer graphics at MIT, Lincoln Lab and Harvard. *ACM SIGGRAPH’89 Panel Proceedings*, New York, 19–38.
- Illich, I. (1973). *La convivialité*. Le Seuil, Paris.

- Johnson, D.L. and Kobles, A.L. (1962). The man and computer relationship. The potential contributions of computers crucially depend upon their use by very human human beings. *Science*, 138, 873–879.
- Jones, J.C. (1969). Design methods in architecture. In *On Design Methods*, Broadbent, G. and Ward, A. (eds). AA Paper, London.
- Kay, A. (1972). A personal computer for children of all ages. *Proceedings of the ACM National Conference*, Boston, MA.
- Kay, A. (1990). User interface: A personal view. In *The Art of Human-Computer Interface Design*, Laurel, B. (ed.). Addison-Wesley Publishing, Boston, MA.
- Kittler, F. (2015). *Le logiciel n'existe pas. Mode protégé*. Les presses du réel, Dijon [Original published 1991].
- Kotsioris, E. (2015). The architecture and the computer: A contested history. *The Architectural Review* [Online]. Available at: <https://www.architectural-review.com/archive/architecture-and-the-computer-a-contested-history?blocktitle=essays&contentid=11449> [Accessed 6 January 2021].
- Lafitte, J. (1972). *Réflexions sur la science des machines*. Vrin, Paris [Original published 1932].
- Laurière, J.L. (1982). Représentation et utilisation des connaissances. Première partie : les systèmes experts. *Technique et science informatiques*, 1(1), 25–42.
- Le Roux, R. (2009). De Wiener à Simondon : penser l'invention avec et sans Bergson. In *Cahiers Simondon – Numéro 1*, Barthélémy, J.-H. (ed.). Harmattan, Paris.
- Le Roux, R. (2014). Présentation de l'édition française. In *La cybernétique. Information et régulation dans le vivant et la machine*, Wiener, N. (ed.). Le Seuil, Paris.
- Le Roux, R. (2015). À propos de la filiation entre cybernétique et sciences cognitives. Une analyse critique de *Aux origines des sciences cognitives* de J.-P. Dupuy (1994). *Bulletin d'histoire et d'épistémologie des sciences de la vie*. Éditions Kimé, Paris.
- Lebahar, J.-C. (1983). *Le dessin d'architecte. Simulation graphique et réduction d'incertitude*. Parenthèse, Marseille.
- Lénárd, I. (1998). *Kas Oosterhuis, Architect; Ilona Lénárd, Visual Artist*. 010 Publishers, Rotterdam.
- Lescop, L. and Chamel, O. (2020). La place des outils numériques dans l'enseignement du projet : comparaison entre une école française et une école américaines. In *Actes du 9^e SCAN Culture numérique et conception architecturale... Retour vers le futur*, Derycke, D., Martin, G., Ben Rajeb, S. (eds). Brussels.
- Lichnerowicz, A. (1972). Préface. In *Journées informatique et conception en architecture 1971*, Lichnerowicz, A., Marcel, P., Nivat, M. (eds). IRIA, Rocquencourt.
- Licklider, J.R.C. (1960). Man-computer symbiosis. *IRE Transactions on Human Factors in Electronics* (HFE-1), 4–11.

- Lynn, G. (2003). Variations calculées. In *Architectures Non Standard*, Migayrou, F. and Mennan, Z. (eds). Centre Pompidou, Paris.
- Manheim, M.L. (1966). Problem-solving processes in planning and design. *Design Quarterly*, 66/67, 30–39.
- Manovich, L. (2013). *Software Takes Command*. Bloomsbury, New York.
- March, L. and Steadman, P. (1971). *The Geometry of Environment: An Introduction to Spatial Organization in Design*. Routledge, New York.
- Maroy, J.-P. (1971). Introduction. In *Journées informatique et conception en architecture 1971*, Lichnerowicz, A., Marcel, P., Nivat, M. (eds). IRIA, Rocquencourt.
- McLuhan, M. (1968). *Pour comprendre les médias. Les prolongements technologiques de l'homme*. HMH, Montréal [Original published 1964].
- Mendes Correia, R., Paio, A., Soares, A.L. (2018). Architectural design digital change: Interactivity policy. *Sigradi 2018 Technopoliticas*.
- Mitcham, C. (1991). Tools for conviviality: Argument, insight, influence. In *Europe, America and Technology: Philosophical Perspectives*. Durbin, P.T. (ed.). Kluwer Academic Publishing, Dordrecht.
- Mitchell, W.J. (1977). *Computer-Aided Architectural Design*. Van Nostrand Reinhold Company, New York.
- Montebello, P. (2006). Simondon et la question du mouvement. *Revue philosophique de la France et de l'étranger*, 3, 279–297.
- Negroponte, N. (1970). *The Architecture Machine. Toward a More Human Environment*. MIT Press, Cambridge, MA.
- Negroponte, N. (1975). *Soft Architecture Machines*. MIT Press, Cambridge, MA.
- Negroponte, N. and Groisser, L.B. (1970). The semantics of architecture machines. *The Architectural Forum*, 133(3), 38–41.
- Neumann, A. (2020). *L'architecture de la morphologie*. Éditions B2, Paris [Original published 1963].
- Quinrand, P. (1986). L'image en architecture. In *Imaginaire numérique*, Renaud, A. (ed.). Hermès, Paris.
- Quinrand, P., Autran, J., Florenzano, M., Fregier, M., Zoller, J. (1985). *La conception assistée par ordinateur en architecture*. Hermès, Paris.
- Rheingold, H. (1990). An interview with Don Norman. In *The Art of Human-Computer Interface Design*, Laurel, B. (ed.). Addison-Wesley Publishing, MA.
- Rheingold, H. (1993). *La réalité virtuelle. Quand l'illusion à toutes les apparences de la réalité*. Dunod, Paris [Original published 1991].
- Ross, D. (1960). The computer aided design: A statement of objectives. Technical report 8436-TM-4, Electronic Systems Laboratory, MIT, Cambridge, MA.

- Salanskis, J-M. (2011). *Le monde du computationnel*. Les Belles Lettres, Paris.
- Schumacher, P. (2009). Parametricism: A new global style for architecture and urban design. *Architectural Design – Digital Cities*, 79(4), 14–23.
- Ségal, J. (2003). *Le zéro et le un : histoire de la notion scientifique d'information au 20e siècle*. Syllepse, Paris.
- Serraino, P. (2002). *History of Form Z*. Birkhäuser, Basel.
- Serraino, P. (2003). Form follows software. *ACADIA 22*, Indianapolis, IN, 185–205.
- Simon, H. (2004). *Les sciences de l'artificiel*. Gallimard, Paris [Original published 1969].
- Simondon, G. (2001). *Du mode d'existence des objets techniques*. Aubier, Grenoble [Original published 1958].
- Simondon, G. (2014a). Mentalité technique. In *Sur la technique (1953–1983)*. PUF, Paris [Original published 1961].
- Simondon, G. (2014b). Culture et technique. In *Sur la technique (1953–1983)*. PUF, Paris [Original published 1965].
- Simondon, G. (2014c). Sauver l'objet technique. In *Sur la technique (1953–1983)*. PUF, Paris [Original published 1983].
- Stals, A. (2021). Panorama des pratiques numériques dans les bureaux d'architecture belges de petite et moyenne taille. *DNArchi* [Online]. Available at: <https://dnarchi.fr/analyses/panorama-des-pratiques-numeriques-dans-les-bureaux-darchitecture-belges-de-petite-et-moyenne-taille/> [Accessed 1 February 2022].
- Stiegler, B. (1986). Repro-graphismes : l'historique du dessin. In *Imaginaire numérique*, Renaud, A. (ed.). Hermès, Paris.
- Stiny, G. and Gips, J. (1978). *Algorithmic Aesthetics: Computer Models for Criticism and Design in the Arts*. University of California Press, Berkeley, CA.
- Sutherland, I.E. (1963). Sketchpad: A man-machine graphical communication system. Technical report 574, MIT, Cambridge, MA.
- Terzidis, K. (2006). *Algorithmic Architecture*. Architectural Press, Oxford.
- Teyssot, G. and Bernier-Lavigne, S. (2011). Forme et information. Chronique de l'architecture numérique. In *Action Architecture*, Guiheux, A. (ed.). La Villette, Paris.
- Teyssot, G. and Jacques, O. (2009). Faire parler les algorithmes. Les nuages virtuels du Metropol Parasol (Séville). *Le visiteur : revue critique d'architecture*, 14, 101–121.
- Triclot, M. (2008). *Le moment cybernétique. La constitution de la notion d'information*. Champ Vallon, Seyssel.
- Triclot, M. (2011). *Philosophie des jeux vidéo*. La Découverte, Paris.

- Van Wyk, C.S.G., Bhat, R., Gauchel, J., Hartkopf, V. (1991). A knowledge-based approach to building design and performance evaluation. *ACADIA Conference Proceedings Reality and Virtual Reality*, Los Angeles, 1–14.
- Vardoulis, T. and Touloumi, O. (2019). *Computer Architectures: Constructing the Common Ground*. Routledge, New York.
- Walker, J. (1989). *The Autodesk File: The No-Holds-Barred Story of the Making of a Software Company*. New Riders Publishing, San Francisco, CA.
- Walker, J. (2015). *The Autodesk File: Bits of History, Words of Experience*. New Riders Publishing, San Francisco, CA [Online]. Available at: <https://www.fourmilab.ch/autofile/> [Accessed 5 April 2021].
- Weisberg, D.E. (2005). The engineering design revolution. The people, companies and computer systems that changed forever the practice of engineering. *CADHistory* [Online]. Available at: <https://www.cadhistory.net/> [Accessed 25 May 2021].
- Wiener, N. (1960). Some moral and technical consequences of automation. As machines learn they may develop unforeseen strategies at rates that baffle their programmers. *Science*, 131, 1355–1358.
- Wright Steenson, M. (2014). Architectures of information: Christopher Alexander, Cedric Price, and Nicholas Negroponte and MIT's Architecture Machine Group. Thesis, Princeton University, NJ.
- Wright Steenson, M. (2017). *Architectural Intelligence: How Designers and Architects Created the Digital Landscape*. MIT Press, MA.
- Yessios, C.I. (1987). The computability of void architectural modeling. In *Principles of Computer-Aided Design: Computability of Design*, Kalay, Y.K. (ed.). John Wiley & Sons, New York.
- Zublena, A. and Duplay, C. (2000). Le stade de France, de la main à la machine, interview. *Le Carré Bleu, La création architecturale et l'informatique*, 1/2, 26–39.

