Flexibility out of standardization

Giacomo Cabri
Department of Physics, Informatics and Mathematics, University of Modena and Reggio Emilia, Modena, Italy, and
Guido Fioretti
Department of Management, University of Bologna, Bologna, Italy

Received 14 November 2020
Revised 1 July 2021
9 January 2022
25 March 2022
Accepted 8 April 2022

Abstract

Purpose – This article aims to provide a theoretical unifying framework for flexible organizational forms, such as so-called adhocracies and network organizations.

Design/methodology/approach – In this article, organization practices that are typical of the software industry are analyzed and re-interpreted by means of foundational concepts of organization science. It is shown that one and the same logic is at work in all flexible organizations.

Findings – Coordination modes can be fruitfully employed to characterize flexible organizations. In particular, standardization is key in order to obtain flexibility, provided that a novel sort of coordination by standardization is added to those that have been conceptualized hitherto.

Research limitations/implications – This article highlights one necessary condition for organizations to be flexible. Further aspects, only cursorily mentioned in this paper, need to be addressed in order to obtain a complete picture.

Practical implications – A theory of organizational flexibility constitutes a guide for organizational design. This article suggests the non-obvious prescription that the boundary conditions of individual behavior must be standardized in order to achieve operational flexibility.

Social implications – This theoretical framework can be profitably employed in management classes.

Originality/value – Currently, flexible organizations are only understood in terms of lists of instances. This article shows that apparently heterogeneous case-studies share common features in fact.

Keywords Lean manufacturing, Adhocracy, Coordination by standardization, Flat organizations, Network organization, Self-managed organizations

Paper type Research paper

Introduction

Flexibility is a business buzzword since at least the 1980s, with fashionable, innovative, cutting-edge organizations being labeled – among other features – "flexible" in some respect. At least two names have appeared in organization studies to characterize this new, emergent organizational form. The first one, the *Adhocracy*, stresses with its root *ad hoc* the ability of flexible organizations to re-configure themselves for each novel problem or environment they face in an ever-changing world (Mintzberg, 1979). The second one, the *Network Organization*, rather focuses on mutable and flexible linkages between organization members, eventually fostered by widespread usage of information technologies (Miles and Snow, 1986) [1]. Both of them implicitly or explicitly assume that organizational flexibility stems from some sort of openness and adaptability of individual behavior and interpersonal relations, which allows to combine freedom of action with a high degree of interpersonal coordination.

Thus, the discourse on flexible organizations runs parallel with calls for exploration vs. exploitation (March, 1991), with exploration of novel possibilities providing the benefits of innovation and better tracking of consumers' needs but also generating additional costs with

International Journal of Organization Theory & Behavior
Vol. 25 No. 1/2, 2022
pp. 22-38
Emerald Publishing Limited
1093-4537
DOI 10.1108/IJOTB-11-2020-0197

respect to long-term planning and mass production of standardized goods. Or, equivalently, the benefits of free entrepreneurship in spite of the notoriously high failure rate of new enterprises vs. the rigidity of monopolies and – in the limit – planned economies. Entrepreneurial or intrapreneurial behavior is costly for organizations as well as for society at large, but returns substantial benefits in terms of innovativeness and flexibility.

Given this trade-off, organizations generally strike a balance between the amount of exploration enabled by organizational flexibility and the amount of exploitation enabled by planning. Sometimes they purposely overshoot, allowing exaggerate levels of flexibility in order to harness innovations that they subsequently exploit by means of more rigid organization structures (Fioretti, 2012). Given the importance of such policies for organizations of any sort, a theory is in order to provide guidance on how flexibility can be attained. Henceforth, we shall demonstrate that one essential block of such a theory does not require novel constructs but rather an extension of classical concepts.

Our elaboration is based on interpretive frameworks that have been developed by software engineers out of practical experiences in their industry. The remarkable fact about the software industry is that it extensively dealt with the problem of designing and managing flexible organizations, solved it, and theorized the solution unaware of organization principles but in accord with their basic tenets. We submit that, by means of a proper generalization of their findings based on extending the scope of the coordination modes identified by classical organization studies, the trend towards flexibility can be understood in terms of one and the same principle in any industry where it occurs.

The discovery of this principle did not occur in the software industry by chance. Computer code is being written by increasingly larger teams that require a huge degree of coordination because any tiny decision made by one single programmer is likely to reverberate onto other sections of the code affecting the work of all other programmers. Close supervision is impossible, for it would be extremely cumbersome for a supervisor to understand exactly all the implications of all lines of code that all programmers are writing. Likewise, standardizing work processes is impossible beyond a limited extent because writing computer code requires creativity and problem-solving skills.

The software industry solved its organizational problem by standardizing the boundary conditions for otherwise unsupervised behavior, sort of rails within which programmers are free to express their creativity and desire for exploration. Upon casting their solution in more general terms, we re-interpret a few canonical examples of flexible organizations that have been widely discussed in organization studies.

In detail, the rest of this article is organized as follows. First of all we review the literature on coordination modes, highlighting linkages with flexibility. Secondly, and most importantly, we analyze the concepts and techniques that emerged in the software industry in order to organize programmers' work. Subsequently, we re-interpret several instances of flexible organizations in the light of what happened in the computer industry. A final section concludes.

Coordination modes

The idea of deriving organizational processes from coordination modes can be traced back to March and Simon (1958, § 6.4) but it was greatly extended by Thompson (1967) and further refined by Mintzberg (1979) [2]. Sometimes these authors re-named existing concepts, a circumstance which forced us to accept multiple names:

(1) *Coordination by feed-back* (March and Simon, 1958) / *Coordination by mutual adjustment* (Thompson, 1967) [3]. This coordination mode comes about when organization members interact attentively and creatively, organizing their activities

out of their own efforts at coming to terms with one another. Classical examples include doctors and patients, as well as airplane crews and sporting teams. It is the one coordination mode upon which organizations can count in order to attain creativity and flexibility [4].

(2) *Coordination by plan* (March and Simon, 1958; Thompson, 1967) / *Coordination by (direct) supervision* (Mintzberg, 1979) [5]. This occurs when someone plans the activities of someone else. Supervision is then necessary in order to ensure that the plan is correctly executed. Unlike coordination by feed-back/mutual adjustment, coordination by plan/supervision requires a hierarchy to be put in place.

(3) *Coordination by standardization* (Thompson, 1967; Mintzberg, 1979). The greater the standardization, the easier it is for organization members or organizational units to come to terms with one another. Standardization can either emerge out of spontaneous initiatives of coordinating actors, or it can be imposed by a hierarchy.

Coordination modes can be used to issue prescriptions for organizational design. For instance, Thompson (1967, p. 60) suggested that organizations should first of all design elementary units whose members would coordinate by mutual adjustment, then embed these units in a plan for coordinated action whose execution would be supervised by a hierarchy, and finally arrange units together that could be bound to one another by standardization. By contrast, Puranam *et al.* (2012) have maintained that after designing elementary units whose members would coordinate by feed-back, the other coordination modes could be used in any suitable combination. Other prescriptions are obviously possible, depending also on the sort of organization that is being designed.

In this respect, it is interesting to remark that although one may think of exceedingly formal organizations where coordination by feed-back/mutual adjustment is nearly absent, as well as small anarchistic organizations where coordination by plan/supervision does not exist, it is extremely difficult to think of organizations where standardization has no place. In certain organizations standardization may be possibly absent at a very early stage, when possibilities are still being explored, but sooner or later sets in.

Given the importance of standardization for organizing, it is surprising that it received so little attention in the scholarly literature. March and Simon mentioned instances of standardization – such as homogeneity of goods, interchangeable parts and buffer inventories – that, as they put it, aimed at "reducing the infinite number of things in the world (. . .) to a moderate number of well-defined varieties" (1958, p. 159). Thompson (1967, p. 56) employed a broader definition of standardization as "rules which constrain action (. . .) into paths consistent with those taken by others in the interdependent relationship," but went on clarifying that standardization required stable, repetitive situations to the point that even coordination by supervision would allow more flexibility than the amount standardization could tolerate: "Coordination by (. . .) [supervision] does not require the same high degree of stability and routinization that are required for coordination by standardization, and therefore is more appropriate for more dynamic situations (. . .)" (Thompson, 1967, p. 56). In these quotes standardization was clearly meant to affect the tiny details of activities, and in quite a trivial way.

Mintzberg (1979, 1989) greatly expanded the variety of standardization by tracing a distinction between standardization of *work processes*, *outputs*, *skills*, and *norms*. Standardization of work processes is epitomized by Taylorism but characterizes mass production in general, just like standardization of outputs. While these two concepts are not equivalent, they are clearly linked to one another, and both of them were implied in the above quotations by March and Simon (1958) and Thompson (1967). Henceforth we shall refer to the *standardization of processes and outputs* (eventually shortened as *standardization of*

processes) as one single sort of standardization, keeping in mind that this was the only sort of standardization that both March and Simon (1958) and Thompson (1967) had in mind.

By contrast, the standardization of skills – possibly better known as standardization of competencies today – is a substantial departure from it. Standardization of skills or competencies characterizes professional bureaucracies, it is carried out to a large extent outside the organization where professionals are employed, and is somewhat looser than the standardization of processes and outputs. Without denying the difference between skills and competences but mindful that this concept applies to both of them, we shall use the term *standardization of skills / competencies* henceforth.

Finally, standardization of norms is eventually induced by ideologies that make organization members display homogeneous behavior. According to Mintzberg (1979), ideologies play an increasingly prominent role as a necessary glue of otherwise loosely coupled adhocracies.

Figure 1 illustrates the taxonomy and characteristics of standardization as they have been codified between the 1950s and the 1980s. This is the received wisdom that has appeared in countless textbooks and informed scholarly discourse for decades.

However, a broader understanding of standardization was occasionally achieved by authors who did not explicitly mention it as a coordination mode. Notably, Gouldner (1954) remarked that in modern bureaucracies impersonal rules are often used to replace explicit supervision. In particular, Gouldner observed that reminding of rules is generally preferable to issuing orders that can be resented, and covertly opposed. Consequently, managers should actively engage in substituting rules for orders as a less obtrusive, more effective technique to get things done. By mapping "rules" into coordination by standardization and "orders" into coordination by supervision, we may remark that Gouldner was pointing to an interesting trade-off between coordination by supervision and coordination by standardization. Mintzberg (1979) took on Gouldner's insight, but did not build on it.

By contrast, Blau (1970) envisaged a trade-off between standardization and flexibility in the sense that greater standardization (with this wording) can allow for greater coordination by feed-back / mutual adjustment (with a different wording). However, this trade-off appeared to him as a marginal issue that he briefly mentioned in a book chapter and never resumed again.

Many years later Eisenhardt and Sull (2001) made quite similar a statement, claiming that flexible organizations can be obtained if management restrains from control while focusing on issuing simple rules. They listed several sorts of rules, including "boundary rules" as well as "how-to rules," "timing rules," "priority rules" and exit rules, making clear that rules should neither be too broad, nor too vague, nor implicit in order to be effective. In the next section we shall subsume these "rules" under the concept of standardizing boundary conditions.

Also studies on modularity independently arrived at quite similar a point of view (Sanchez and Mahoney, 1996). In particular, Grote *et al.* (2004, 2009) remarked that organizational routines can turn organizations into loosely-coupled systems that combine predictability with flexibility. They pointed to risk management as a domain where the flexibility allowed by rules that generate modularity was quickly discovered (Rasmussen, 1997; Hale and Swuste, 1998), and we shall see in the next section that the same applies to modular programming structures.

$$Standardization\ of \begin{cases} Processes\ and\ outputs & \leftrightarrow & Rigidity \\ Skills\ /\ Competencies & \leftrightarrow & Professionalism \\ Norms & \leftrightarrow & Ideology \end{cases}$$

Figure 1.
According to the received wisdom, standardization can either concern processes and outputs, or skills / competencies, or norms

In management studies, several empirical investigations found specific instances of standardization enabling flexibility. In particular, Argyres (1999) analyzed the development of a military airplane that required thousands of designers from four contractors to interact with one another, highlighting that IT communication helped developing a "standardized technical grammar" that enabled them to make decisions based on accurate expectations about each other's plans. Likewise, Valentine and Edmondson (2015) analyzed a hospital emergency department finding that pre-definition of roles improved team effectiveness by providing what they called a "team scaffold" wherein doctors and nurses could display their capabilities.

Finally, Harris, Hevner and Webb Collins is the one single piece of management research suggesting that providing boundaries to human interactions is key to enable creative, adaptive and flexible human interactions (Harris *et al.*, 2009a, b) [6]. Just like us, they have been inspired by the software industry where flexible management techniques have been widely discussed and analyzed. On the whole, all we added to their findings is a connection to the standard taxonomy of coordination modes and the pretense that it is actually a general pattern that applies to all industries.

The taxonomy of coordination modes has progressed substantially since March and Simon (1958), Thompson (1967) and Mintzberg (1979). In particular, Coordination Theory (CT) embarked in a comprehensive collection of coordination rules at a much more detailed level than those authors did, drawing insights from disciplines that include but are not limited to organization science (Malone and Crowston, 1994; Crowston, 1997; Crowston *et al.*, 2006). CT provides a map of variants of each coordination mode which, particularly for coordination by standardization and feed-back/mutual adjustment, makes it less of an art and more of a management technique. We stay at a more aggregate level, but perfectly compatible with their findings.

The organization of coding

Thompson (1967) prescribed that organizations should be designed to confine most coordination by mutual adjustment within its lowest-level units, which in their turn would be included in middle-level organizational units that a hierarchy of managers would coordinate by plan / supervision and standardization. Interestingly, writing computer code is an extremely tough test-bed for these prescriptions. The first reason is that writing computer code requires feed-back/mutual adjustment to such an extent that it is extremely difficult to arrange programmers into semi-independent groups (Staudenmayer, 1997). Secondly, writing computer code is a creative problem-solving activity that can be hardly reconciled with executing detailed directives. Observing, understanding and re-directing any single choice would require so much supervisory effort to make micromanagement impossible even if it were sought. Finally, writing computer code is a professional activity that is – at least to some extent – standardized by the community of programmers. Just as it happens with all professions, management has little say on this.

In spite of all difficulties, large efforts have been made at modularizing the process of writing computer code. In particular, the following practices have made it easier to arrange programmers into groups that are self-contained, at least to some extent:

(1) Sharp separation between those programmers who write the core code and those who are concerned with ancillary activities, such as software documentation, portability, testing, and interfaces (Baker, 1972);

(2) Re-usage of libraries and other pieces of software that had been developed for other or similar purposes (Boehm, 1987);

(3) Modularity of computer code made possible by Object-Oriented Programming (OOP), where self-contained pieces of code – called "objects" – interact in an artificial space (Yourdon, 1993). Recently, the modularity of OOP has been further enhanced by Aspect-Oriented Programming (AOP) (Filman *et al.*, 2005).

All these practices are sensible and useful, but of limited effectiveness. In spite of the greater modularity allowed by OOP and AOP, the so-called *Brooks' Law* (Brooks, 1975) is still acknowledged to hold by most software engineers (McCain and Salvucci, 2006): "Adding manpower to a late software project makes it later."

Brooks' Law is due to the fact that feed-back/mutual adjustment is so demanding that the time it takes to explain to the newly added workforce what others did and what tasks they may take charge of, is very likely to exceed the time savings that newcomers contribute once they are finished with learning what has been done hitherto (Brooks, 1975; Hsia *et al.*, 1999). Feed-backs and the need for mutual adjustment are so pervasive that in spite of all efforts the time to complete a software project roughly squares with the number of programmers (Brooks, 1975) and, quite remarkably, this figure did not improve substantially over decades (Berkling *et al.*, 2009).

At first sight, organizational design appears to face a hopeless task. Modularization has been actively pursued, with important but limited results (Thomas and Murphy, 2011). Consequently, the classical prescriptions of organizational design are extremely difficult to follow. So how does it happen that large software projects do exist, involving hundreds or even thousands of programmers?

The answer is that software engineers have learned to separate software development in two sequentially distinct phases. First, the *architecture* of the software is specified. Subsequently, computer code is written to fit that architecture. Software architecture includes specification of what modules a software must be made of, how these modules must interact, what information they must pass to one another as well as to human operators and peripheral devices such as printers, screens or beamers, what information they receive from human operators, datasets or devices, and so on. Experience suggested that allocating substantial time to proper architectural design ensures that fewer bugs will be reported, that fixing them will not cause other bugs to appear elsewhere, and that the assignment is more likely to be completed on time (Brooks, 1975; Boehm, 1979; De Marco, 1997). Drawing a similarity with gothic cathedrals whose construction could extend for centuries, involving generations of artists without ever changing the original design, this has been called the *cathedral model* of software development (Brooks, 1975).

The cathedral model is based on standardization. However, software architects do not standardize the work processes of their fellow programmers, but rather provide a scaffolding structure for them to express their creativity and coordinate with one another by means of feed-backs and mutual adjustments. They standardize the boundary conditions of human behavior, not behavior itself. This idea, which originated in the 1970s, constitutes the base of all subsequent developments in software engineering in spite of all tremendous changes that happened in this industry.

A first wave of structural changes was triggered by Free / Open Source Software (FOSS). Right in the 1970s software was becoming a commodity, and therefore had to be covered by copyright. In order to protect intellectual property, software houses would sell the executable files of their programs – those that are installed on computers – without making the *source code* available, which is what programmers actually write. However, this deprived expert users of the possibility to improve the code, a possibility that is of substantial importance for cutting-edge applications. According to Richard Stallman, author of the foundational GNU [7]. Manifesto (Stallman, 1985), the ensuing sense of frustration was the initial reason for FOSS (Stallman, 1985–2015).

FOSS was enabled by a new legal framework. It was the GNU Public License (Stallman and Moglen, 1989), which allowed programmers to distribute source code while retaining intellectual property. Since the 1980s the FOSS community is surprising the world with extremely efficient software – including, among else, *Linux*, *Apache*, *R* and *Mozilla* – which is being made available for free. The main reason for its efficiency is that with public access to the source code many programmers can make themselves available to add features and fix bugs (Ghosh, 1998). Notably, in order to exploit this sort of crowd contributions FOSS versions are generally released very often, even at very early development stages (Raymond, 1999).

With this practice FOSS abandoned the idea of delivering software when it is ready for usage, a circumstance which is quite uncommon for most services and products. It had the possibly unwanted, but extremely valuable consequence of allowing the software requirements to change and eventually adapt to users' needs. We shall see later on that this feature of FOSS eventually inspired a further milestone in the organization of programmers' work.

FOSS projects are generally conceived by one or a few individuals who launch a proposal on dedicated web sites such as *Savannah*, *SourceForge*, *GitHub* or *GitLab*. Others may join, creating communities that form and dissolve around specific features or modules of the software (Gonzales-Barahona *et al.*, 2004; Crowston and Howison, 2005). Whenever divergent opinions emerge, the project *forks* into two or more projects pursuing different avenues, which in some cases reconcile at a later point in time though generally they do not (Robles and Gonzalez Barahona, 2012; Viseur, 2012).

This fluid, bubbling community looks opposite to the carefully planned work that is carried out in software houses. The constructive chaos of a bazaar has been contrasted to the cathedral model, purporting that chaotic bazaars can do better than the careful planners of cathedrals (Raymond, 1999).

However, this statement needs qualification. While it is indisputable that FOSS can achieve superior quality, this does not imply that no coordination is at work whatsoever.

Raymond based his statements on his own experience as the initiator and manager of a successful FOSS project. One crucial feature of his experience was that he did not start from scratch but rather from a piece of existing software. Most importantly, when he had to make a choice he selected what he would call the most "solid" and "professional" code over one that had more advanced features but less orderly a structure (Raymond, 1999, p. 26). This was quite an important decision, which he motivated with the fact that the more orderly structure would ease re-writing and re-using the code (Raymond, 1999, p. 25).

We embrace the observation that FOSS needs an architecture just like proprietary software does, the difference being that proprietary software builds its own architecture in-house whereas FOSS makes use of someone else's architecture (Valloppillil, 1998; Bezroukov, 1999; Massey, 2002; Narduzzo and Rossi, 2005; Langlois and Garzarelli, 2008). This amounts to maintain that just like proprietary software, also FOSS manages coordination by feed-back/mutual adjustment by standardizing the boundaries wherein it takes place.

Most FOSS developers are volunteers. They are far too few with respect to the number of projects that they launch, so the vast majority of projects do not attract any follower at all and are discontinued after some time (Krishnamurthy, 2002; Capiluppi *et al.*, 2003). In the end FOSS provided useful competitors to inefficient proprietary software, but it did not make proprietary software go extinct.

While FOSS remained a niche phenomenon, proprietary software strove to reap the benefits of a FOSS practice such as frequently releasing provisional versions. In particular, since the 1990s *Agile Programming* (AP) is changing the organization of coding towards increasingly flexible arrangements and greater adherence to customers' changing needs (Dybå and Dingsøyr, 2008; Anwer *et al.*, 2017). Just like other "agile" management streams in manufacturing and elsewhere, AP aims at organizing the production of software for less

predictable, more turbulent business environments (Nerur and Balijepally, 2007; Conboy, 2009).

In its early days, AP rejected the cathedral model altogether [8]. The *Agile Manifesto* claimed that the best architectures and designs emerge from self-organizing teams (Beedle *et al.*, 2001), apparently reverting a trend that had been established through decades.

AP quickly spread among software houses because they increasingly had to deal with customers who required software whose specifications could change while it was being developed. However, with AP meeting both successes and failures it became clear that architecture could be considered irrelevant only in those settings where one was already there in the implicit form of domain-specific design patterns, re-usable libraries or other pieces of existing software, and/or specifications on interfaces imposed by standards or compatibility needs (Turk *et al.*, 2005; Booch, 2007; Bellomo *et al.*, 2014). In a sense AP discovered, just like FOSS, that it could do without an architecture only if one was already there.

The conflictual relation between AP and software architecture generated considerable concern because AP allowed customers to define and re-define requirements while the code was being developed, forcing developers to accept changes that could jeopardize the architectural base on which the code was being written (Breivold *et al.*, 2010). Software architects made huge efforts to meet AP's needs (Poupko, 2015, 2016). Several schemes to devise modular, flexible architectures were devised, requiring that architects and programmers meet several times while code is being developed in order to adjust to one other's needs (Nord and Tomayko, 2006; Hadar and Silberman, 2008; Sharifloo *et al.*, 2008; Bachmann *et al.*, 2012; Bellomo *et al.*, 2014).

We submit the following interpretation of this uneasy relationship of AP with architectures. Architectures, in our view, standardize the boundary conditions within which coordination by feed-back/mutual adjustment can take place with a minimum of planning and supervision. However, these boundaries must be loose and wide enough not to constrain programmers' work into rigid schemes. Increasingly volatile customers' requirements called for the adaptive, flexible boundaries which AP in the end provided.

Also in this respect, the case-study of software engineering offers interesting insights. It shows that managing flexible organizations is not just about setting boundaries and let people mutually adapt to one another, but it often implies continuous revision and update of those boundaries in order to avoid that the very facilitators of freedom turn into strait jackets.

Notably, this aspect of AP closely mirrors a concept that eventually developed in evolutionary organization studies. In evolutionary theory, phenotypes exhibit a certain degree of *plasticity* in spite of genotypes being fixed (Miner *et al.*, 2005). By extension, evolutionary organization science makes use of the term "plasticity" in order to denote adaptations that can be made in spite of certain organizational routines, culture and norms being fixed (Levinthal and Marino, 2015; Herath and Secchi, 2021). Now, consider an organization standardizing boundaries in order to enable creative exploration, while at a slower time scale upgrading those very boundaries from time to time. This organization is enabling flexibility but, if it ever reaches the point where its boundary conditions reach certain routines, culture and norms that it cannot change, then the sort of flexibility that it is enabling should be rather called plasticity. In other words, plasticity defines the maximum flexibility that an organization can attain.

We shall briefly review other instances henceforth, but none of them reaches the depth of questioning the rigidity of boundary conditions, as software engineering did. *A fortiori*, it will not be possible to assess whether flexibility reached its upper limit, plasticity.

A general pattern

The analysis of the experiences accumulated in software engineering suggests a possible extension of a basic building block of organization science, namely coordination by

standardization. More precisely, we define *standardization of boundary conditions* as any form of standardization that sets the framework where interactions take place without specifying the details of those interactions. Figure 2 illustrates our extension with respect to previous taxonomies as they appeared in Figure 1.

We purport that standardization of boundary conditions can be eventually employed to make organizations more flexible, up to reaching plasticity. Specifically, we suggest that greater coordination by standardization of boundary conditions allows to increase coordination by feed-back/mutual adjustment while coordination by plan/supervision and standardization of processes and outputs can be correspondingly reduced. Figure 3 depicts one possible shift of the balance of coordination modes. For the sake of simplicity, standardization of skills/competencies and standardization of norms have been assumed not to change.

If our theoretical construct is useful, then it must be able to shed light on a wide variety of flexible organizational arrangements. With this claim we do not maintain that standardizing boundary conditions is all what is needed in order to make organizations more flexible, but we submit that this is one key ingredient. It is not a sufficient condition to achieve greater flexibility, but certainly a necessary one.

Henceforth we shall review a few instances of organizations that have been often cited as paradigmatic of the trend towards greater flexibility. In particular, the work carried out by the Tavistock Institute in the 1950s has been a forerunner of all attempts to reach greater flexibility, lean manufacturing heralded flexibility in the 1980s whereas self-managed organizations are leading the heyday since the 2000s. We shall not discuss all features of these organizations but rather focus on those aspects that can be usefully interpreted with the sort of balance shift illustrated in Figure 3.

The marrows in the British coal mines

The Tavistock Institute initiated the paradigm of *sociotechnical systems* with an inquiry on work organization in British coal mines in the 1950s (Trist *et al.*, 1963) which, with hindsight, has been considered a forerunner of the increasingly flexible organizational arrangements that would eventually emerge a few decades later (Manz and Stewart, 1997). Let us first of all summarize their findings.

The traditional organization of work in British coal mines was based on *marrows* (teams) of typically six miners. Each marrow was assigned a branch of a mine, and each marrow would cover all 24 h work by means of three shifts of two miners each. Each of the two miners in a shift was able to do all the jobs that were required to extract coal, namely, hewing and loading the coal into tubs that brought it to the surface, extending the roof support, clearing stones and extending rails. A marrow was composed by self-selected workers who shared a piecemeal wage based on the number of tubs that were delivered to the surface in 24 h.

The two miners in a shift coordinated by feed-back/mutual adaptation all of their work processes, from hewing the coal to extending the rails. In their turn, the three shifts coordinated with one another pairwise, in the sense that it was fine for a shift to spend substantial effort in extending the roof and the rails instead of shipping tubs to the surface because payment was shared across all shifts anyway.

$$\textit{Standardization of} \begin{cases} \textit{Processes and outputs} & \leftrightarrow \textit{Rigidity} \\ \textit{Skills / Competencies} & \leftrightarrow \textit{Professionalism} \\ \textit{Norms} & \leftrightarrow \textit{Ideology} \\ \textit{Boundary Conditions} & \leftrightarrow \textit{Flexibility} - \textit{Plasticity} \end{cases}$$

Figure 2.
With our extension, standardization of boundary conditions adds to the standardization of processes and outputs, of skills/competencies, and norms

In the first decades of the XX century this traditional organization gave place to a fordist arrangement where teams of miners specialized in one single process would either load coal on a conveyor belt, or clear the mine from stones, or extend the roof under the supervision of a foreman. This organization was made possible by technical innovations that allowed for much larger tunnels and was expected to generate greater efficiency. Surprisingly, it did not. One reason was that specialized miners waited for orders from their foreman instead of coordinating with one another by feed-back/mutual adjustment. Furthermore, each shift had an incentive to leave burdens to the next one, such as stones to clean. Finally, a foreman had to be there, down at the coal face.

Trist and his colleagues proposed a solution where the traditional arrangement was applied to larger marrows – each marrow would consist of ca 40 workers spread over three shifts of 12–15 workers each – in order to exploit the greater productivity of modern equipment. Since all the members of a marrow would share piecemeal remuneration, one management layer – the foremen at the coal face – could be eliminated.

In our interpretation, this solution amounts to decrease coordination by plan/supervision (elimination of the foreman) and decrease standardization of work processes while increasing coordination by feed-back/mutual adjustment (all miners could take on all roles and arrange work by themselves). To achieve this aim, it was crucial that the boundary conditions be standardized, where coordination by feed-back/mutual adjustment would unfold. Rules for piecemeal payment, rules for equal payment of all members of a marrow, rules for prospective miners to self-select in a marrow are means to achieve coordination by standardizing boundary conditions. Just like the software architecture for programmers, these rules permitted spontaneous coordination by feed-back/mutual adjustment to thrive.

Autonomation in lean manufacturing

Originally developed by Toyota in the 1960s and '70s, since the 1980s lean manufacturing has been sweeping around the world contrasting workers' *autonomation* to the extreme standardization of work processes in fordist plants. Lean manufacturing is based on quite many elements, among which the following are possibly the most important ones (Sugimori *et al.*, 1977):

(1) Workers are not assigned one single task but rather carry out one out of a set of several tasks, depending on the vagaries of demand for each particular product;

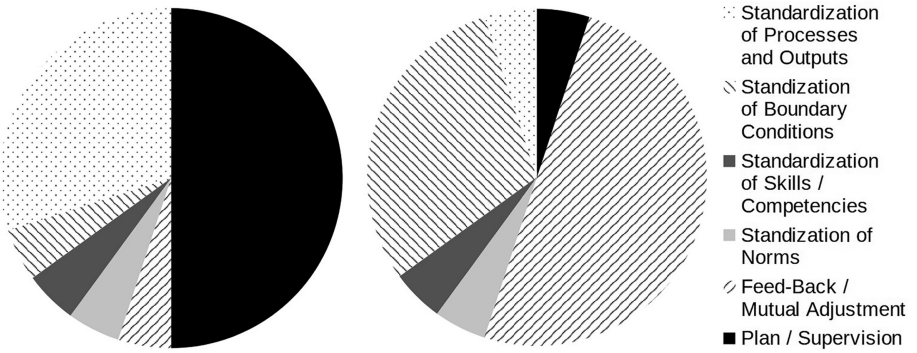

Figure 3.
Left, the balance of coordination modes for a rather rigid organization mostly relying on coordination by plan / supervision and standardization of processes and outputs; Right, the balance of coordination modes in a more flexible arrangement where standardization of boundary conditions has been used to allow greater coordination by feed-back / mutual adjustment while decreasing the previously dominating coordination modes

(2) Instead of detailed supervision, workers communicate with one another through *kanbans* (signals) as soon as they are finished with their current activity;

(3) Workers are allowed and even encouraged to stop production whenever they cannot keep up with its pace, a circumstance which appears on the *andon* public board;

(4) Inventories between production islands, work teams or otherwise defined organizational units, as well as with customers and suppliers, are discouraged or eliminated (*just-in-time*).

Points (1) and (2) amount to decrease coordination by plan/supervision while increasing coordination by feed-back/mutual adjustment. The interesting question is how this is achieved.

According to Parker and Slaughter (1990), the above point (3) is key. The managers of lean production plants can afford to exert less supervision because they overload workers, and they know that they reached the right amount of overload out of the information displayed on the *andon* board. The goal is not to run production smoothly with the *andon* showing no sign of distress, but rather that for each working team its lights occasionally flash, signaling that all of them receive a sufficiently high workload from time to time. Within this logic, inventories must be eliminated because they would allow teams or other organizational units to hide their real burden, hence point (4) above. Parker and Slaughter (1990) labeled this arrangement of production *management by stress*.

In this understanding of lean manufacturing, (3) and (4) set the boundaries for adaptation by feed-back/mutual adjustment to take place. Just like architecture for programmers or piecemeal payment to the whole marrow of coal miners, they ensure that mutual adjustment displays itself along the desired paths and domains.

Incidentally, we would like to remark that in both examples eliminating middle managerial layers called for some form of indirect control. In lean manufacturing this was achieved by means of the *andon*, whereas in the British coal mines piecemeal payment of the coal tubs that were being shipped to surface was essential for managers to ensure that work was being actually carried out.

Self-managed organizations

Self-managed organizations encourage their members to take initiatives rather than executing orders, providing them with the means to carry out their visions if they find a sufficient critical mass of colleagues who are willing to join them (Bernstein *et al.*, 2016). Quite often, organizations of this sort have been presented as ideal places to work, hyped by the popular press as the organization of the future and labeled with curious names ranging from "organic" to "spaghetti" or "teal" organizations, or "holacracies," but there is really no magic about them. Henceforth we shall illustrate some of their features, highlighting that providing boundaries to their members' behavior is essential to make them work.

Self-managed organizations are eventually labeled as "leaderless organizations" or, more accurately, "less hierarchical" or "flat" organizations. In reality, hierarchy does exist in these organizations, and it is even quite prominent (Denning, 2014), but of a different sort from the ones we are generally accustomed to. In the usual terms of a hierarchy being exemplified by a pyramidal structure, self-managed organizations can be flat to the point of exhibiting two hierarchical levels only. However, they continuously create, destroy and re-create nested structures where someone sets the goals and checks the attainments of someone else, with the additional complication that one and the same person generally takes multiple roles, which may be characterized by differential rights towards others. It is a magma of nested, partially overlapping and often short-lived hierarchies, rather than anarchy (Bernstein *et al.*, 2016). Salaries are generally higher than average, but much less differentiated. Differences of

remuneration rather arise because of *ad hoc* compensation for accomplishing specific goals (Hamel, 2011).

Self-managed organizations are not the one single organizational form that will characterize our future, though it will be one of them. It is appropriate for organizations that must quickly adapt to market changes bypassing the inefficiencies inherent in forwarding information upwards and waiting for (not always appropriate) decisions percolating downwards; however, enabling all organization members to pursue their own initiatives comes at a cost that may not be offset by savings on managerial layers. The story of Oticon, a Danish hearing aids producer, is instructive in this respect. In this case, the self-managed organization was allowed to last for only a few years, until a sufficient number of innovations had been generated. Once the aim had been accomplished, generating novel ideas that could not be pursued was a waste of resources, hence this otherwise successful experience was terminated (Fioretti, 2012).

Self-managed organizations are characterized by a huge number of norms, rules of conduct, even "constitutions" in some cases (Lee and Edmondson, 2017). We found testimonies by their practitioners claiming that precisely these rules allow their members to take initiatives, such as "The structure allows you freedom" and similar claims (Robertson *et al.*, 2021). We argue that this is a clear instance of coordination by feed-back and mutual adjustment being enabled by standardization of the boundaries of acceptable behavior.

Finally, we would like to discuss a problem that has been repeatedly reported with self-managed organizations, namely that such a huge amount of norms may make organization members watch one another instead of tracking customer needs (Denning, 2014). The response has been that, indeed, top management must make clear that the ultimate purpose is satisfying customers (Compagne, 2014; Ward, 2017). In this response we detect an affinity with the "management by stress" so essential for lean manufacturing. The *ad hoc*, nested and intertwined hierarchies of the self-managed organization must be made busy with goals that are triggered by the one portion of the organization that is absolutely stable – the strategic apex – which, if those goals are challenging enough, does not need to check the details of the operations being carried out.

Conclusions

While adhocracies and network organizations are not novel theoretical constructs, we extended the gamut of coordination modes by adding a variant to the list with the aim of gaining a deeper understanding on those organizational forms. We relied heavily on concepts that have been developed and tested by software engineers without awareness of organization principles, but with a rare taste for theorizing tacit knowledge into codified practices and empirically observable outcomes.

We showed that, contrary to commonsensical expectations, standardization of boundary conditions is key to make people express themselves in active and creative manners, conducive to exploration of novel possibilities and capable of achieving organizational flexibility. However, the very analysis of software engineering that highlighted the creative power unleashed by standardizing boundaries hinted at the possibility that standardization of boundary conditions turns into a cage that impairs the very flexibility it was designed to foster. Flexibility cannot be taken as achieved once and for all, but is rather a moving target that at any time can easily eschew. Moreover, core identity behaviors set the plasticity upper bound to flexibility.

One obvious limitation of our analysis is that although we did mention the existence of standardization of skills or competences, as well as standardization of norms, we did not use these concepts at all. And yet, both sorts of standardization play key roles in the move towards flexibility. Standardization of skills/competences, once the province of professional

bureaucracies, is likely to expand in knowledge-based economies where employees carry on into their organizations the mental schemes that they have learned during their formal education. Likewise, standardization of norms is likely to be paramount for organizations that unite their members – among else – by means of ideologies, such as those emanating from being self-managed or the possibility for any worker to stop production in lean manufacturing plants.

Another limitation of our scheme is that it does not make it explicit that whenever middle managerial layers are eliminated, some indirect form of control must be put in place. Management by stress is likely to be an extreme case, but milder forms of indirect control are always in place. Although we did mention it in the three examples of the previous section, this fundamental aspect of flexible organizations does not appear in the balance of coordination modes.

Finally, we ignored several key aspects of the move towards flexibility such as the cognitive effort that is required by team members, the hybridization between organizational forms, or the very doubt whether such extremely flexible organizations can be still called bureaucracies. Nevertheless, we hope to have to some extent advanced our understanding of what adhocracies and network organizations are, and on which operating principles they are based.

Notes

1. Miles and Snow (1986) actually employed the term *dynamic networks* in the text of their paper, but the expression *Network Organization* appeared in the title. This is the term that subsequent scholars eventually picked up.
2. March and Simon (1958,p. 160) pointed to the importance of standardization but did not label it a coordination mode. They identified two coordination modes instead: *coordination by feedback* and *coordination by plan*. Thompson (1967) identified the three forms of *coordination by mutual adjustment* (which corresponds to March and Simon's *coordination by feedback*), *coordination by plan* and *coordination by standardization*, respectively. Finally, Mintzberg (1979) preferred the expression *coordination by direct supervision* to *coordination by plan*, which eventually entered common usage with the shorter version of *coordination by supervision*.
3. Thompson (1967,p. 56) rejected the expression "coordination by feed-back" claiming that "[. . .] the term "feedback" has gathered a connotation of super/subordination which unduly restricts it for our purposes." With hindsight, his point is totally unjustified.
4. Puranam *et al.* (2012) correctly observed that Thompson (1967) had defined coordination by mutual adjustment in a way that ignored the need to predict and adjust to one other's behavior, which is implied by March and Simon's coordination by feed-back instead (March and Simon, 1958). We rather chose to keep these two expression aside, for two reasons. The first one is that it is practically impossible to eradicate an expression that has settled in scientific discourse. The second one is that, in subsequent usage, coordination by mutual adjustment has acquired those cognitive connotations whose importance Puranam *et al.* (2012) correctly stressed. Consider also that two of the above examples – doctors and patients, and airplane crews – stem from Thompson (1967) himself, and yet they do require cognitive and interpretive abilities.
5. Mintzberg (1979) employed the expression *Coordination by direct supervision*. The adjective "direct" was eventually dropped by subsequent authors.
6. Harris *et al.* (2009a) and Harris *et al.* (2009b) coined the concept of *scope boundaries*, which is equivalent to our usage of the term *boundaries* except that they did not relate it to standardization.
7. With a recursive definition, the acronym GNU stands for "Gnu is Not Unix." The creation of an alternative to the proprietary operative system Unix marked a major achievement of the free software community.

8. For the sake of precision, AP rejected the "waterfall model" but never mentioned the cathedral model. The waterfall model implies that software production must undergo a fixed sequence of steps where defining the architecture necessarily precedes coding. For our purposes, waterfall model and cathedral model can be taken as synonyms.

References

Anwer, F., Aftab, S., Waheed, U. and Muhammad, S.S. (2017), "Agile software development models TDD, FDD, DSDM, and crystal methods: a survey", *International Journal of Multidisciplinary Sciences and Engineering*, Vol. 8, pp. 1-10.

Argyres, N.S. (1999), "The Impact of information technology on coordination: evidence from the B2 'Stealth' bomber", *Organization Science*, Vol. 10, pp. 162-180.

Bachmann, F., Nord, R.L. and Ozkaya, I. (2012), "Architectural tactics to support rapid and agile stability", *CrossTalk*, Vol. 25, pp. 20-25.

Baker, F.T. (1972), "Chief programmer team management of production programming", *IBM Systems Journal*, Vol. 11, pp. 56-73.

Beedle, M., Van Bennekum, A., Cockburn, A., Cunningham, W., Fowler, M., Highsmith, J., Hunt, A., Jeffries, R., Kern, J., Marick, B., Martin, R.C., Schwaber, K., Sutherland, J. and Thomas, D. (2001), "The agile Manifesto", available at: https://agilemanifesto.org (accessed 4 March 2020).

Bellomo, S., Kruchten, P., Nord, R.L. and Ozkaya, I. (2014), "How to agilely architect an agile architecture", *Cutter IT Journal*, Vol. 27, pp. 10-15.

Berkling, K., Kiragiannis, G., Zundel, A. and Datta, S. (2009), "Timeline prediction framework for iterative software engineering projects with changes", in Berkling, K., Joseph, M., Meyer, B. and Nordio, M. (Eds), *Software Engineering Approaches for Offshore and Outsourced Development*, Springer, Berlin-Heidelberg, pp. 15-32.

Bernstein, E., Bunch, J., Canner, N. and Lee, M. (2016), "Beyond the holacracy hype", *Harvard Business Review*, Vol. 7, pp. 38-49, July-August.

Bezroukov, N. (1999), "A second look at the cathedral and the bazaar", *First Monday*, 4, available at: http://firstmonday.org/article/view/708/618.

Blau, P.M. (1970), "Decentralization in bureaucracies", in Zald, M.N. (Ed.), *Power in Organizations*, Vanderbilt University Press, Nashville, TN, pp. 150-174.

Boehm, B.W. (1979), "Software engineering - as it is", *Proceedings of the 4th International Conference on Software Engineering*, Munich, pp. 11-21.

Boehm, B.W. (1987), "Improving software productivity", *Computer*, Vol. 20, pp. 43-57.

Booch, G. (2007), "The economics of architecture-first", *IEEE Software*, Vol. 24 No. 5, pp. 18-20.

Breivold, H.P., Sundmark, D., Wallin, P. and Larsson, S. (2010), "What does research say about agile and architecture?", in Hall, J., Kaindl, H., Lavazza, L., Buchgeher, G. and Takaki, O. (Eds), *Fifth International Conference on Software Engineering Advances*, Los Alamitos, CA, IEEE Computer Society, pp. 32-37.

Brooks, F.P. (1975), *The Mythical Man-Month: Essays on Software Engineering*, Addison-Wesley, Reading, MA.

Capiluppi, A., Lago, P. and Morisio, M. (2003), "Evidences in the evolution of OS projects through changelog analyses", in Feller, P., Fitzgerald, B., Hissam, B. and Lakhani, K. (Eds), *Taking Stock of the Bazaar: Proceedings of the 3rd Workshop on Open Source Software Engineering*, Washington, DC, IEEE Computer Society, pp. 19-24.

Compagne, O. (2014), "Holacracy is not what you think", *Holacracy*, available at: https://blog.holacracy.org/holacracy-is-not-what-you-think-67144c3adf8 (accessed 4 January 2022).

Conboy, K. (2009), "Agility from first principles: reconstructing the concept of agility in information systems development", *Information Systems Research*, Vol. 20, pp. 329-354.

Crowston, K. (1997), "A coordination theory approach to organizational process design", *Organization Science*, Vol. 8, pp. 157-175.

Crowston, K. and Howison, J. (2005), "The social structure of free and open source software development", *First Monday*, 10, available at: http://firstmonday.org/ojs/index.php/fm/article/view/1207/1127.

Crowston, K., Rubleske, J. and Howison, J. (2006), "Coordination Theory: a ten-year retrospective", in Zhang, P. and Galletta, D. (Eds), *Human-Computer Interaction in Management Information Systems*, M.E. Sharpe, New York, NY, pp. 120-138.

De Marco, T. (1997), *The Deadline: A Novel about Project Management*, Dorset House Publishing, NewYork, NY.

Denning, S. (2014), *Making Sense of Zappos and Holacracy*, Forbes, January, available at: https://www.forbes.com/sites/stevedenning/2014/01/15/making-sense-of-zappos-and-holacracy/?sh=5c52efee3207.

Dybå, T. and Dingsøyr, T. (2008), "Empirical studies of agile software development: a systematic review", *Information and Software Technology*, Vol. 50, pp. 833-859.

Eisenhardt, K.M. and Sull, D.N. (2001), "Strategy as simple rules", *Harvard Business Review*, pp. 107-116, January.

Filman, R., Elrad, T., Clarke, S. and Aksit, M. (2005), *Aspect-Oriented Software Development*, Addison-Wesley, Boston, MA.

Fioretti, G. (2012), "Two measures of organizational flexibility", *Journal of Evolutionary Economics*, Vol. 22, pp. 957-979.

Ghosh, R.A. (1998), "FM interview with Linus Thorvalds: what motivates free software developers?", *First Monday*, Vol. 3, March, available at: https://firstmonday.org/ojs/index.php/fm/article/view/583/504doi: 10.5210/fm.v3i2.583.

Gonzales-Barahona, J.M., Lopez, L. and Robles, G. (2004), "Community structure of modules in the Apache project", in Feller, J., Fitzgerald, B., Hissam, Scott and Lakhani, K. (Eds), *Collaboration, Conflict and Control: Proceedings of the 4th Workshop on Open Source Software Engineering*, Washington, DC, IEEE Computer Society, pp. 43-48.

Gouldner, A.W. (1954), *Patterns of Industrial Bureaucracy*, Free Press, Glencoe, pp. 157-180.

Grote, G., Zala-Mezö, E. and Grommes, P. (2004), "The effects of different forms of Co-ordination on coping with workload", in Dietrich, R. and Childress, T.M. (Eds), *Group Interaction in High Risk Environments*, Routledge, London, pp. 39-54.

Grote, G., Weichbrodt, J.C., Günter, H., Zala-Mezö, E. and Künzle, B. (2009), "Coordination in high-risk organizations: the need for flexible routines", *Cognition, Technology and Work*, Vol. 11, pp. 17-27.

Hadar, E. and Silberman, G.M. (2008), Agile architecture methodology: long term strategy interleaved with short term tactics, *Companion to the 23rd ACM SIGPLAN Conference on Object-Oriented Programming Systems Languages and Applications*, New York, NY, Association for Computing Machinery, pp. 641-651.

Hale, A.R. and Swuste, P. (1998), "Safety rules: procedural freedom or action constraint?", *Safety Science*, Vol. 29, pp. 163-177.

Hamel, G. (2011), *First, Let's Fire all the Managers*, Harvard Business Review, December, available at: https://hbr.org/2011/12/first-lets-fire-all-the-managers.

Harris, M.L., Hevner, A.R. and Webb Collins, R. (2009a), "Controls in flexible software development", *Communications of the Association for Information Systems*, Vol. 24, article 43.

Harris, M.L., Webb Collins, R. and Hevner, A.R. (2009b), "Control of flexible software development under uncertainty", *Information Systems Research*, Vol. 20, pp. 400-419.

Herath, D.K. and Secchi, D. (2021), "Editorial", *Evidence-Based HRM*, Vol. 9, pp. 121-125.

Hsia, P., Hsu, C.T. and Kung, D.C. (1999), Brooks' Law revisited: a system dynamics approach, *Proceedings. Twenty-Third Annual International Computer Software Conference and Applications*, Los Alamitos, CA, IEEE Computer Society, pp. 370-375.

Krishnamurthy, S. (2002), "Cave or community? an empirical examination of 100 mature open source projects", *First Monday*, pp. 6-3, available at: http://firstmonday.org/ojs/index.php/fm/article/view/960.

Langlois, R.N. and Garzarelli, G. (2008), "Of hackers and hairdressers: modularity and the organizational economics of open-source collaboration", *Industry and Innovation*, Vol. 15, pp. 125-143.

Lee, M.Y. and Edmondson, A.C. (2017), "Self-Managing Organizations: exploring the limits of less-hierarchical organizing", *Research in Organizational Behavior*, Vol. 37, pp. 35-58.

Levinthal, D.A. and Marino, A. (2015), "Three facets of organizational adaptation", *Organization Science*, Vol. 26, pp. 743-755.

Malone, T.W. and Crowston, K. (1994), "The interdisciplinary study of coordination", *ACM Computing Surveys*, Vol. 26, pp. 87-119.

Manz, C.C. and Stewart, G.L. (1997), "Attaining flexible stability by integrating total quality management and socio-technical systems theory", *Organization Science*, Vol. 8, pp. 59-70.

March, J.G. (1991), "Exploration and exploitation in organizational learning", *Organization Science*, Vol. 2, pp. 71-87.

March, J.G. and Simon, H.A. (1958), *Organizations*, John Wiley & Sons, New York, NY.

Massey, B. (2002), "Where do open source requirements come from (and what should we do about it)?", *ICSE 2nd Workshop on Open Source Software Engineering*, available at: http://flosshub.org/system/files/Massey.pdf.

McCain, K.W. and Salvucci, L.J. (2006), "How influential is Brooks' Law? a longitudinal citation context analysis of Frederick Brooks' the mythical man-month", *Journal of Information Science*, Vol. 32, pp. 277-295.

Miles, R.E. and Snow, C.C. (1986), "Network Organizations: new concepts for new forms", *California Management Review*, Vol. 28, pp. 62-73.

Miner, B.G., Sultan, S.E., Morgan, S.G., Padilla, D.K. and Relyea, R.K. (2005), "Ecological consequences of phenotypic plasticity", *Trends in Ecology and Evolution*, Vol. 20, pp. 685-692.

Mintzberg, H. (1979), *The Structuring of Organizations*, Prentice-Hall, Englewood Cliffs, NJ.

Mintzberg, H. (1989), "The structuring of organizations", in Asch, D. and Bowman, C. (Eds), *Readings in Strategic Management*, MacMillan, London, pp. 322-352.

Narduzzo, A. and Rossi, A. (2005), "The role of modularity in free/open source software development", in Koch, S. (Ed.), *Free/Open Source Software Development*, Idea Group Publishing, Hershey, PA, pp. 84-102.

Nerur, S. and Balijepally, V.G. (2007), "Theoretical reflections on agile development methodologies", *Communications of the ACM*, Vol. 50, pp. 79-83.

Nord, R.L. and Tomayko, J.E. (2006), "Software architecture-centric methods and agile development", *IEEE Software*, Vol. 23, pp. 47-53.

Parker, M. and Slaughter, J. (1990), "Management by stress: the team concept in the US auto industry", *Science As Culture*, Vol. 1, pp. 27-58.

Poupko, A. (2015), "It has been a long journey, and it is not over yet", in Lassenius, C., Dingsøyr, T. and Paasivaara, M. (Eds), *Agile Processes in Software Engineering and Extreme Programming. Proceedings of the 16th International Conference (XP 2015)*, pp. 270-278.

Poupko, A. (2016), "The journey continues: discovering my role as an architect in an agile environment", in Sharp, H. and Hall, T. (Eds), *Agile Processes in Software Engineering and Extreme Programming. Proceedings of the 17th International Conference (XP 2016)*, pp. 226-234.

Puranam, P., Raveendran, M. and Knudsen, T. (2012), "Organization design: the epistemic interdependence perspective", *Academy of Management Review*, Vol. 37, pp. 419-440.

Rasmussen, J. (1997), "Risk management in a dynamic society: a modelling problem", *Safety Science*, Vol. 27, pp. 183-213.

Raymond, E.S. (1999), "The cathedral and the bazaar", *Knowledge, Technology and Policy*, Vol. 12, pp. 23-49.

Robertson, B., Lee, M., Mays, K., Janse, D., Allen, D., Brover, R., Timmerman, R., Krupa, G. and Everhart, T. (2021), "Rules of the game", *Holacracy*, video, available at: https://www.holacracy.org/explore or https://www.youtube.com/watch?v=TZSlwK4di_M (accessed 4 January 2022).

Robles, G. and González Barahona, J.M. (2012), "A comprehensive study of software forks: dates, reasons and outcomes", in Hammouda, I., Lundell, B., Mikkonen, T. and Scacchi, W. (Eds), *Open Source Systems: Long-Term Sustainability*, Springer-Verlag, Berlin-Heidelberg, pp. 1-14.

Sanchez, R. and Mahoney, J.T. (1996), "Modularity, flexibility, and knowledge management in product and organization design", *Strategic Management Journal*, Vol. 17, pp. 63-76.

Sharifloo, A.A., Saffarian, A.S. and Shams, F. (2008), "Embedding architectural practices into extreme programming", in Hussain, F.K. and Chang, E. (Eds), *Proceedings of the 19th Australian Software Engineering Conference*, Los Alamitos, CA, IEEE Computer Society, pp. 310-319.

Stallman, R.M. (1985), "The GNU Manifesto", *Dr. Dobb's Journal of Software Tools*, Vol. 10, pp. 30-35.

Stallman, R.M. (1985-2015), "The GNU Manifesto", available at: https://www.digitalmanifesto.net/manifestos/171.

Stallman, R.M. and Moglen, E. (1989), "GNU general public license, version 1", available at: https://www.gnu.org/licenses/old-licenses/gpl-1.0.html.

Staudenmayer, N. (1997), "*Managing multiple interdependencies in large scale software development projects*", PhD Thesis, MIT Alfred P. Sloan School of Management, Massachussets Institute of Technology (MIT), Cambridge, MA.

Sugimori, Y., Kusunoki, K., Cho, F. and Uchikawa, S. (1977), "Toyota production system and kanban system materialization of just-in-time and respect-for-human system", *The International Journal of Production Research*, Vol. 15, pp. 553-564.

Thomas, N. and Murphy, G. (2011), "How effective is modularization?", in Oram, A. and Wilson, G. (Eds), *Making Software: What Really Works, and Why We Believe it*, O'Reilly Media, Sebastopol, CA, pp. 373-391.

Thompson, J.D. (1967), *Organizations in Action: Social Science Bases of Administrative Theory*, McGraw-Hill, New York, NY.

Trist, E.L., Higgin, G.W., Murray, H. and Pollock, A.B. (1963), *Organizational Choice: Capabilities of Groups at the Coal Face under Changing Technologies*, Tavistock Publications, London.

Turk, D., France, R. and Rumpe, B. (2005), "Assumptions underlying agile software-development processes", *Journal of Database Management*, Vol. 16, pp. 62-87.

Valentine, M.A. and Edmondson, A.C. (2015), "Team Scaffolds: how mesolevel structures enable role-based coordination in temporary groups", *Organization Science*, Vol. 26, pp. 405-422.

Valloppillil, V. (1998), "Halloween document I", available at: http://www.catb.org/~esr/halloween/halloween1.html.

Viseur, R. (2012), "Forks impacts and motivations in free and open source projects", *International Journal of Advanced Computer Science and Applications*, Vol. 3, article 21.

Ward, C. (2017), "The Zappos Story: is holacracy a proven structure for improving customer experience?", *Mycustomer*, available at: https://www.mycustomer.com/service/management/the-zappos-story-is-holacracy-a-proven-structure-for-improving-customer (accessed 4 Jan 2022).

Yourdon, E. (1993), *Object-Oriented Systems Design: an Integrated Approach*, Prentice-Hall, Englewood Cliffs, NJ.

Corresponding author
Guido Fioretti can be contacted at: guido.fioretti@unibo.it