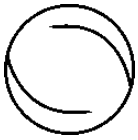
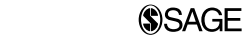
Special Themed Section on Marxist Studies of Organization



Organization Studies
2015, Vol. 36(4) 445–471

DOI: 10.1177/0170840614561566
www.egosnet.org/os



Community and Innovation: From Tönnies to Marx

Paul S. Adler
University of Southern California, USA

Abstract

The idea of community has lurked in various forms in organization studies since the field's inception, but its recent prominence as a critical precondition for innovation makes urgent the resolution of two theoretical puzzles. Both puzzles can be stated in the terms suggested by Tönnies' classic contrast of *Gemeinschaft* and *Gesellschaft*, community and association. First, it is difficult to reconcile the idea that community is critical to innovation with the traditionalistic character of *Gemeinschaft*. Second, it is difficult to reconcile any idea of community in work organizations with the conflictual character of the capitalist employment relation and the instrumental *Gesellschaft* character of the economic sphere. I argue that the resolution of the second puzzle via Marxist theory leads us to a resolution of the first. My thesis, in summary, is that community is a critical component of the capitalist labour-process, and that where this labour-process is oriented toward innovation, community is taking an historically new form. This new form represents a dialectical synthesis of *Gemeinschaft* and *Gesellschaft*, a form we can call *Genossenschaft*, or collaborative. The argument is essentially theoretical; I illustrate some key features of this emergent collaborative form with case data from a software services firm. In conclusion I suggest that this new form represents communism developing in the heart of capitalism.

Keywords

innovation, knowledge work, organizational forms, topics

Introduction

Although the word itself appears only occasionally in organization studies, community has been an enduring preoccupation in our field. Community – which we can define provisionally as a social collectivity bound by a common identity, values, and norms – was already implicated (on a small scale) in Frederick Taylor's arguments about work-teams' tendency to 'soldier' (Taylor, 1972) and (on a larger scale) in the Hawthorne study's analysis of social networks among shop-floor workers (Roethlisberger & Dickson, 1939). Since then, community has been, I submit, the common core underlying concepts such as informal organization, intra- and inter-organizational networks, organic organizational form, organizational culture, social capital, clans, and industrial districts.

Corresponding author:
Paul S. Adler, Marshall School of Business, University of Southern California, Bridge Hall 306, Los Angeles, CA 90089-0808, USA.
Email: padler@usc.edu

In recent years, the theme of community has become increasingly prominent, in particular in discussions of innovation in industry, and most notably in the literature on communities of practice, technical communities, open source communities, and supplier communities (O'Mahony & Lakhani, 2011). In many accounts, the growing importance of community within and beyond firms is due to community's critical role in stimulating innovation and to the increasingly important role of innovation in firms' competitiveness (Adler, 2001; Benkler, 2006; Brown & Duguid, 2001; Grant & Baden-Fuller, 2000; Powell, 1989). Enthusiasts claim that the proliferation of these forms of community is dramatically reshaping the industrial landscape (e.g. Castells, 2011).

Community's recent prominence in studies of innovation makes urgent the resolution of two interrelated theoretical puzzles. Both can be stated in the terms suggested by Tönnies' classic contrast of *Gemeinschaft* and *Gesellschaft* (Tönnies, 1957). First, it is difficult to reconcile the purported role of community as an antecedent to innovation with the standard accounts of community. The paradigmatic form of community – Tönnies' *Gemeinschaft*, familiar to organization studies as Ouchi's 'clan' type of organization (Ouchi, 1980) – is essentially traditionalistic, notoriously insular, conservative, and status bound, and as such it is a context that is hardly conducive to dynamic innovation. Not surprisingly, therefore, many discussions of community's role in innovation simply ignore the need to characterize community theoretically, using the term simply to designate any collectivity, without asking what kind of social order is being invoked (as noted by West & Lakhani, 2008).

Second, it is difficult to reconcile this enthusiasm for community with the body of sceptical writing arguing that the basic structure of capitalism – its fundamentally competitive and exploitative nature, its predominantly instrumental and contractual *Gesellschaft* character – makes any idea of community in industry a fantasy (e.g. Alvesson & Thompson, 2006). More generally, the field of organization studies is somewhat polarized between those who see community as a primordial feature of persistent human collectivities, including businesses (e.g. d'Iribarne, 2003), and on the other side those who see power asymmetries as a fundamental feature of social structures (e.g. Reed, 2001) and who therefore critique as essentially obfuscatory any affirmation of bonds of community within industry. Indeed, debates on community and related concepts reflect the long-standing tension in sociology between theories of regulation and order versus theories of change and conflict (Burrell & Morgan, 1979; Contu & Willmott, 2003). Marxist theory – at least in its conventional form – provides one of the more compelling conflict-based critiques of the community thesis.

To respond to this pair of puzzles, this paper develops an alternative reading of Marx on community. This Marxist resolution of the second puzzle, I argue, also offers a resolution of the first. As regards the second puzzle, my reading of Marx's theory of the capitalist production process (building on Adler, 2007) suggests that community – in the form of what Marx calls the 'collective worker' – is an essential feature of the labour process, even under antagonistic capitalist employment conditions. As regards the first puzzle, I argue that the capitalist system based on exploitation and competition drives firms to higher levels of productivity and innovation, and in order to respond to these pressures, firms are led to reconfigure the labour-process community in a distinctive new form. This new form represents a dialectical synthesis of traditionalistic *Gemeinschaft* and contractual *Gesellschaft*, a synthesis that we might call collaborative *Genossenschaft* (German for partnership, cooperative, comradeship). This tendency, I argue, is in constant tension with the capitalist firm's profitability imperative, which both encourages the emergence of this new form of community and simultaneously undermines and thwarts it.

In developing this argument, the following sections first discuss the concept of community and define the two puzzles in more detail. Second, I review the role played by this concept in Marx's theory, and argue that capitalist development drives the emergence of this new form of community

within the capitalist enterprise along four dimensions: norms, values, authority, and capabilities. I then illustrate this transformation with some materials from a study of a software services company. A conclusion draws some implications, suggesting in particular that the emergence of a collaborative form of community can be understood as communism developing in the heart of capitalism.

Community and Innovation: Two Puzzles

In sociology, the term community is usually taken to signify a social group characterized by common geography, face-to-face interaction, emotional bonds of loyalty, and homogeneous values and norms (Calhoun, 1998; Delanty, 2003; Hillery, 1955; Williams, 1985). In this literature, Tönnies (1957) is often the founding reference, contrasting traditional (pre-capitalist) and modern (capitalist) society by their respective anchoring in community – *Gemeinschaft* – versus association – *Gesellschaft. Gemeinschaft* is characterized by traditionalism, shared values and norms, mutual commitment, ascribed status, limited division of labour, and simple social structures; *Gesellschaft* is characterized by voluntary, anonymous, arm's length, contractual relations, and instrumental values (Brint, 2001; Calhoun, 1998). *Gemeinschaft* is implicit or explicit in many discussions of community among scholars of organization studies: Ouchi's 'clan', for example, is explicitly grounded in Tönnies' concept (Ouchi & Barney, 2004).

As Gläser (2001) points out, recent decades have seen a shift in usage: scholars and laypeople alike have increasingly applied the term community in a more minimalist fashion, to collectivities that are neither normatively dense nor geographically bounded, without face-to-face interaction or affective ties. Indeed, on Gläser's account, community as such is any collectivity in which participants have 'a perception of having something in common with others' (p. 7). Djelic and Quack (2010) follow a similar path. In the language proposed by Ren et al. (2007), this is a community based on mere categorical identity rather than on interpersonal bonds.

It is this minimalist understanding of community that seems to be at work in many studies of innovation, such as those listed by O'Mahony and Lakhani (2011) on open source communities as well as those on scientific communities, occupational communities, communities of practice, technical communities, and online communities. To this list we could add: epistemic communities (Haas, 2009), supplier communities (Doel, 1999; Sheth & Sharma, 1997); brand communities (McAlexander, Schouten, & Koenig, 2002; Muniz Jr & O'Guinn, 2001), consumer communities (Kruckeberg & Starck, 2004; Shang, Chen, & Liao, 2006), and user communities (Morrison, Roberts, & Von Hippel, 2000; Von Hippel, 2005).

Puzzle #1: How can community support innovation?

Neither *Gemeinschaft* nor minimalist community provides a context conducive to innovation. Take first *Gemeinschaft*: notwithstanding the popularity of Ouchi's clan concept in innovation research, *Gemeinschaft* creates a context that is clearly inhospitable to innovation. *Gemeinschaft* is characterized by strong loyalty to insiders, high barriers to outsiders, and low tolerance for diversity, whereas an impressive body of research has shown that innovation thrives on difference and diversity (DiTomaso, Post, & Parks-Yancy, 2007; Gulley & Lakhani, 2010; Jehn, Greer, Levine, & Szulanski, 2008; Page, 2008; Van Knippenberg, De Dreu, & Homan, 2004) and on a dispersed network of weak ties (Granovetter, 1982; Ruef, 2002). Scholars have expressed concern that for precisely this reason, the clan-like cohesion that characterizes some communities of practice (Brown & Duguid, 2001, p. 202) may support exploitation of existing capabilities but not the innovative creation of new capabilities (Cohendet & Simon, 2008; Lindkvist, 2005; Nooteboom, 2008).

This misfit between *Gemeinschaft* and innovation surely helps explain the retreat to the minimalist concept. In reality, however, the weaker, minimal forms of community have weak and variable effects on key outcomes, including innovation (Brint, 2001). Categorical identity is perhaps a necessary condition for innovation, but it is certainly not sufficient: categories can just as easily be formed around collectivities quite opposed to or uninterested in innovation. And on the other hand, many studies find innovation propelled by unusually strong shared norms and values, rather than merely minimal community (e.g. Chatman & Flynn, 2001; O'Reilly, Chatman, & Caldwell, 1991).

Puzzle #2: How can community function under capitalist conditions?

Marxist theory offers an apparently compelling argument for doubting that – even if strong community were conducive to innovation – such community could survive in a capitalist society or enterprise. Take first the societal level. As compared to pre-capitalist societies, in capitalist societies the individual is emancipated – and alienated – from community. Community is replaced by mere instrumental association and class exploitation. As a result, in a class-divided, capitalist society, the state cannot represent the will of a community: it can only be a state for one class in its rule over others. Community here assumes great *ideological* prominence, precisely to compensate for its disappearance as society's *material* foundation. In *The German Ideology* we read:

> Only in community [with others has each] individual the means of cultivating his gifts in all directions; only in the community, therefore, is personal freedom possible. In the previous substitutes for the community, in the State, etc. personal freedom has existed only for the individuals who developed within the relationships of the ruling class, and only insofar as they were individuals of this class. The illusory community, in which individuals have up till now combined, always took on an independent existence in relation to them, and was at the same time, since it was the combination of one class over against another, not only a completely illusory community, but a new fetter as well. In a real community the individuals obtain their freedom in and through their association. (Marx & Engels, 1970, p. 83)

Civil society under capitalism, Marx argues, can only mean bourgeois society – community of and for the propertied class, and mere 'illusory community' for society as a whole.

Let us turn now from the societal to the enterprise level. Inspired by Marx's critique, many Marxists are dismissive of any suggestion that community is operative in advanced capitalist societies. Indeed, so rigorous was this rejection that this dimension of social analysis was long absent from Marxist theorizing, lost in an indeterminate space between the economic base and the superstructure of state and ideology. Gramsci and Polanyi stood as rare exceptions (Burawoy, 2003).[1]

Marx's analysis of the capitalist production process – at least in its conventional reading – buttresses this sceptical view of community. Marx sees the capitalist production process as a contradictory unity of a labour process and a valorization process. In the valorization process, exchange-values in the form of monetary wages, materials costs, capital investment, and sales income are combined to create money profit. In the labour process, use-values in the form of workers' skills and effort, tools, and materials are combined to create new use-values in the form of products or services (Marx, 1977, Appendix). Marx says of the relationship between the two:

> If capitalist direction [of work] is thus twofold in content, owing to the twofold nature of the process of production which has to be directed – on the one hand a social labor process for the creation of a product, and on the other hand capital's process of valorization – in form it is purely despotic. (Marx, 1977, p. 450)

The despotism of the capitalist production process seems to make a mockery of any claims of workplace community.

Community in Marx's Historical Theory

The thesis of the present article is that we can read Marx differently and more fruitfully, finding in his work an account of community that resolves both these puzzles – providing us with a way of understanding both the critical role played by community in the production process and how community comes to take a new form that better supports innovation.

Community in the labour process

Community is an important but little discussed thread running through much of Marx's work (Mahowald, 1973; Megill, 1970; Sayer, 1990; Springborg, 1986). Engeström (1987) makes a strong case for a reading of Marx in which community figures as a crucial element of the transhistorical labour process, mediating (along with material and symbolic tools) the subject's praxis and the object of this activity. Community, Marx writes, is the 'first presupposition' of human productive activity (Marx, 1973, p. 472). However, community plays different roles and assumes different forms in different phases of history.

First, looking back in time, the famous section of Marx's *Grundrisse* on 'Forms which precede capitalist production' (Marx, 1973, pp. 471–514) frames the entire history of humanity prior to capitalism in terms of evolving forms of *Gemeinschaft* community. In pre-capitalist society, Marx writes: 'individuals relate not as workers but as proprietors – and members of a community who at the same time work' (Marx, 1973, p. 471).

> The further back we trace the course of history, the more does the individual, and accordingly also the producing individual, appear to be dependent and to belong to a larger whole. At first, the individual in a still quite natural manner is part of the family and of the tribe which evolves from the family; later he is part of a community, of one of the different forms of the community which arise from the conflict and the merging of tribes. It is not until the eighteenth century that in bourgeois society the various forms of the social texture confront the individual as merely means towards his private ends, as external necessity. (Marx, 1971, Introduction)

Marx describes how the centrality of community in these pre-capitalist societies constrained the development of social productivity:

> All forms […] in which the community presupposes its subjects in a specific objective unity with their conditions of production, or in which a specific subjective mode of being presupposes the communities themselves as conditions of production, necessarily correspond to a development of the forces of production which is only limited, and indeed limited in principle. The development of the forces of production dissolves these forms, and their dissolution is itself a development of the human productive forces. (1973, p. 496)

Second, looking forward to a hoped-for future, Marx saw communism as a reassertion of community, now in a higher form and on a vastly expanded, indeed global, scale. Communism is the 'free association of producers' – the re-emergence of community as a condition of production and as society's overarching organizing principle, displacing market and state. As Marx says in his famous letter on the Russian peasant communes, communism will represent 'a superior form of the 'archaic' type of collective property and production' (Marx, 1989, p. 346).

Third, looking around at the capitalist form of society in which Marx lived and whose basic structure still prevails today, I argue that Marx's views on the 'illusory' character of societal community under capitalism and on the 'despotic' form of the production process are only part of

a more complex, dialectical constellation. Contrary to the conventional reading, I argue that alongside these absences, community is a positive presence in Marx's theory – disguised only very lightly as 'cooperation' within the labour process of the 'collective worker'.[2] The following paragraphs explicate this reading.

With the increased scale and complexity of the labour process wrought by capitalist development, the labour process – the production of use-values – becomes the task of a collectivity. Engels (1978, p. 702) characterized it in these terms:

> Before capitalist production, i.e. in the Middle Ages […] the instruments of labour – land, agricultural implements, the workshop, the tool – were the instruments of labour of single individuals, adapted for the use of one worker [… Capitalist development transformed these productive forces] from means of production of the individual into *social* means of production, workable only by a collectivity of men. The spinning-wheel, the hand-loom, the blacksmith's hammer were replaced by the spinning-machine, the power-loom, the steam-hammer; the individual workshop, by the factory, implying the cooperation of hundreds and thousands of workmen. In like manner, production itself changed from a series of individual into a series of social acts.

These 'social acts' are the acts not of the individual worker but of a 'collective worker' – the entire mass of more or less specialized workers as well as technical and managerial staff, cooperating to produce use-values (Gramsci, 1971, p. 201; Marx, 1977, pp. 464, 468–9, 483, 544, 644, 945).

This collective worker, I submit, must function as a community – a collectivity with shared identity, values, and norms – if it is to support the cooperation essential to the complex, interdependent labour process that characterizes modern capitalist industry. While workers' perfunctory compliance to management authority might suffice to ensure profitability in some settings with very simple production processes or in some locations with unusually low production costs, and while financial incentives can be designed to encourage individual effort and even individual creative exploration, when production is more innovation-oriented the labour process requires that workers engage actively in mutual adjustment and joint problem-solving, and these in turn require a bond of trust that only community – not authority nor incentives – can provide (De Dreu & West, 2001; Eisenberger, Stinglhamber, Vandenberghe, Sucharski, & Rhoades, 2002; Keller, 1997; Ng, Feldman, & Lam, 2010). And that is why the community theme recurs so frequently in the literature on innovation.

Under capitalist conditions, the stability and cohesion of this collective worker community is constantly challenged by the divisive and demotivating effects of the valorization process – the profit imperative that drives firms constantly to expand the value of capital invested. As noted above, the capitalist production process is a contradictory unity of the labour-process (producing use-values) and the valorization process (producing exchange-value and profit); but this contradiction is a 'real' contradiction, not merely a notional one: both poles are operative, and if the 'form' taken by management's direction of the labour process is 'purely despotic', this form is in a relation of real contradiction with the underlying 'content' of cooperation. Even as exploitation (in the valorization process) undermines the collective worker's cohesion, the imperatives of effective use-value creation (in the labour process) constantly impel firms to recreate that community. These two aspects of the production process co-exist in real tension. As a result of this contradictory character of the capitalist production process, the mutual adjustment and problem-solving required by modern production are typically precarious and realized only imperfectly: Delbridge calls the resulting configuration 'conflicted collaboration' (Delbridge, 2007).

This tension is overlain with a second, related one, insofar as some participants in the capitalist labour process occupy contradictory class locations, simultaneously members of the collective

worker and agents of exploitation (Meiksins, Smith, & Berner, 1996; Smith, 1987; Wright, 1985). Engineers and managers often occupy such locations, playing both a productive role as technical experts and an exploitative role as disciplinarians – and the latter often undermines the former, just as within the broader society class divisions often undermine civil society. Alongside the collective worker, management attempts to orchestrate a 'collective capitalist', the better to ensure the loyalty of these categories and thereby ensure valorization.

The socialization of community: From traditionalism to collaboration

The second puzzle can be resolved by recognizing that the capitalist production process embodies a real community in the form of the collective worker in the labour process; what then of the first puzzle? What type of labour-process community would support innovation?

The conventional reading of Marx sees him as arguing that that capitalism starts out by borrowing pre-capitalist, *Gemeinschaft* forms, but that this community is progressively destroyed as capitalism develops and *Gesellschaft* replaces *Gemeinschaft* in the workplace. In Marx's work, capitalist development brings a shift from 'formal' to 'real' subordination (or 'subsumption') of labour to capital (Marx, 1977, Parts 3, 4). At first, capitalist enterprises take over the labour process as they inherited it from feudalism, and work proceeds under the operational control of craft workers and their craft traditions. Only subsequently, when competition and legal limits on the length of the working day force capitalists to search for new sources of surplus value, do capitalists reengineer the labour process to wrest control from craft workers and assure the 'real' subordination of labour through technical and bureaucratic control over the labour process (Braverman, 1974; Edwards, 1979). This process of real subordination eliminates traditional, *Gemeinschaft* work relations and gives them a contractual *Gesellschaft* quality.

The previous section argued that this account misses the enduring importance of community to effective capitalist production: *Gesellschaft* is simply too thin and brittle a bond to support effective use-value production in much modern industry, particularly in innovation-oriented sectors. This leads to two qualifications of the conventional historical account – one minor, the other major.

The minor qualification: even as contractual, *Gesellschaft* relations of technical and bureaucratic control were expanding through capitalist industry during the 20th century; many organizations attempted to sustain cohesion in the collective worker by maintaining important elements of *Gemeinschaft*, in particular by creating paternalistic forms of bureaucracy (Heckscher, 1996; Jacoby & Taras, 1997; Rudolph & Rudolph, 1979). The 'informal' organization was often carefully nurtured, in the form of loyalty to the firm, to the sub-unit, and to the individual boss. The conventional account is surely correct that such traditionalistic bonds were useful to capitalists in suppressing class solidarity; and it is surely also correct that these efforts were doomed by the logic of capitalist development; but the persistence of these efforts also testifies to the importance of community for an effective labour process.

Notwithstanding these factors contributing to *Gemeinschaft's* persistence, in advanced capitalist societies the last decades of the 20th century seemed to bring us to a turning point, where that additive combination of paternalistic *Gemeinschaft* and bureaucratic *Gesellschaft* was no longer very viable. The intensification of global competition, the exhaustion of the Fordist accumulation regime, and the rise of neoliberal ideology drove many organizations to abandon their efforts to maintain worker loyalty (Thompson, 2003; Vidal, 2013). Ruthless hierarchical control and market-based logics destroyed whatever remained of work-place *Gemeinschaft*.

However, alongside the organizations that embraced the hard logic of *Gesellschaft* market relations, other organizations have taken a different path, attempting to create a new form of community that would better satisfy the exigencies of modern, innovation-oriented production:

this is my second, and more important, qualification to the conventional Marxist account. The real subordination of labour is a process in which the core contradictions of the capitalist production process are deepened, not eliminated in capital's favour. Alongside the destruction of *Gemeinschaft* community, capitalist development drives a constructive process that leads to the emergence of a qualitatively new type of community within the capitalist labour process.

The vector of this development is the process that Marx called 'socialization'. As explicated by Adler (2007), Marx's concept of socialization refers to the extent to which an activity embodies the capabilities of the larger society rather than only those that emerge in isolated, local contexts. Capitalist development, Marx argues, drives the progressive socialization of production, replacing reliance on tacit knowledge that is generated and disseminated locally with reliance on knowledge that is explicit, codified, and generated and disseminated globally. Craft and traditional forms of know-how are progressively replaced by science and engineering (Ingvaldsen, 2015). In this process, traditionalistic, parochial *Gemeinschaft* community is replaced, first, by contractual, universalistic *Gesellschaft*, and then, as the deficiencies of the latter prompt a search for a social order that can bring together in a novel synthesis the strengths of *Gemeinschaft* and *Gesellschaft*, we see the emergence of a new type of community – one we might call collaborative, or *Genossenschaft*. Whereas paternalistic bureaucracy buttressed the weaknesses of *Gesellschaft* with the conservative elements of *Gemeinschaft, Genossenschaft* represents a true dialectical supersession of the two earlier forms, incorporating the strengths of both. The emergence of *Genossenschaft* in the labour process is, on this reading, part of the socialization process, representing progress toward more rational, conscious planning and management of cooperation in large-scale, interdependent operations.

Adapting the account proposed by Cohen (1974), I posit that under *Gemeinschaft*, the individual is 'engulfed' by social structures that afford only 'undifferentiated unity'. Interpersonal relations are primarily conditioned by social status rather than individual choice. Under *Gesellschaft*, the individual emerges, but only in the 'alienated' form of 'differentiated disunity'. Social structures afford individual differentiation, but at the cost of the dissolution of traditional communities. The vector of dialectical transcendence of this contradiction would be toward a type of community that affords 'differentiated unity' – where individuals would play differentiated roles, but would bring their individualized consciousness to the cooperation required in the labour process.[3]

Viewed from this vantage point, many of the innovations in work organization under the banners of 'high commitment', 'corporate culture', 'teamwork', and 'soft skills' represent steps in the direction of this new type of community. Some theorists of post-Fordism have taken seriously these initiatives, but the more common Marxist position on this aspect of post-Fordism reflects the conventional Marxist scepticism about community and is therefore centred on a critique of what these initiatives do not achieve (Vidal, 2011). Without minimizing the validity of these critiques, I think that they miss the significance of these efforts in shaping this new type of community.

I characterize this new synthesis along several dimensions. In summary, collaborative, *Genossenshaft* community is …

* in its *norms*, a synthesis of universalism and particularism: formalized procedures are used to ensure universal diffusion of best practices, and systematic ways of tailoring those procedures facilitate their adjustment to particular circumstances.
* in its *values*, a synthesis of individualism and collectivism: the paramount value is the individual's ability to contribute creatively to the community's shared purpose.

The persistence of such norms and values requires buttressing by distinctive authority and capabilities. The synthesis in these dimensions makes *Genossenshaft* community …

* in its *authority* structure, a synthesis of hierarchy and participation: bureaucratic hierarchies are used to ensure organization-wide consistency, but management styles, policy-setting, decision-making, and staff functions are participative and afford workers real influence.
* in its capabilities, a synthesis of differentiation and integration: people have the skills and the incentives they need both to play specialized roles and to contribute actively to the integration of their specialized roles in pursuit of their shared purpose.

These features of collaborative, *Genossenschaft* community are not widespread in industry today, but they are visible in at least partial form in various innovation-oriented organizations (Adler, Kwon, & Heckscher, 2008; Heckscher, 2007; Heckscher & Adler, 2006). They are also consonant with Boltanski and Chiapello's (2005) portrait of the project *cité* and with Sennett's (2012) portrait of 'civility' as a form of sociability in contrast with *Gemeinschaft* 'solidarity'. Given the coexistence of socialization with valorization pressures, it is hardly surprising that these manifestations of collaborative community prove to be precarious. The Marxist thesis advanced here is not that this new form of community is sweeping across industry today, but rather this it is emerging as the new configuration of one pole of a real contradiction, of which the other pole is the persistence of the capitalist valorization imperative. The latter has not changed fundamentally, but the profound changes in the former mean that the basic contradiction of capitalist society is deepening and the form of the class struggle is evolving correspondingly.

Collaborative Community: An Illustration from Software Services

In this section, I illustrate these features of collaborative, *Genossenschaft* community with data from a case study of a firm in the software services industry. Large-scale software systems represent, of course, only one type of innovation; other industries may suggest different models; but my account seems to parallel the results from a range of industries presented in Heckscher and Adler (2006): future research will be needed to test the generalizability of the characterization presented here.

I focus on a key vector of organizational change in software development – the Capability Maturity Model (CMM). (In Europe, a very similar initiative goes under the acronym SPICE ISO/IEC 15504.) The CMM explicitly targets the craft model that predominated in the early years of software production. As software systems grew larger and more complex, the craft model faltered, and the proportion of projects that failed to meet their goals or failed entirely rose dramatically (Gibbs, 1994; Lieberman & Fry, 2001). In the 1980s, the U.S. Air Force studied 17 major software systems contracts and found that every one was late (by an average of 75%) and over budget (Humphrey, 2002). The 'chaos' in large-scale commercial sector projects was (and still is) in general even worse (Jones, 2002; Standish Group, 1994). In 1984, frustrated with such chaos, the U.S. Department of Defense (DoD) funded the Software Engineering Institute (SEI), based at Carnegie-Mellon University, to develop a model of a more reliable software development process. With the assistance of the MITRE Corporation, SEI developed the Capability Maturity Model, releasing a preliminary description in 1987 and the first official version (version 1.1) in 1991.

The development of the CMM illustrates the process of socialization in action: its development was an industry-wide learning process as a broad community of industry people helped shape it. Paulk (1995, p. 11) writes: 'Nearly 1000 external reviewers who were part of a "CMM Correspondence Group" had the opportunity to comment on the various drafts leading to CMM version 1.1. A CMM Advisory Board helped the SEI review and reconcile conflicting requests.' The software CMM was subsequently complemented by CMM tools for systems engineering, people management, and software acquisition. In 2000, several of these were integrated into a broader tool called CMM-Integration.

My research focused on the software CMM. This CMM distinguishes five successively more 'mature' levels of process capability, each characterized by mastery of a number of Key Process Areas (KPAs) – see Table 1. The CMM belongs to a class of improvement approaches that focuses on 'process' rather than 'people'. It does not recommend any particular approach to organizational and behavioural issues: it focuses on the 'whats' and not the 'hows', leaving CMM users to determine their own implementation approach. Level 1 represents an ad hoc, craft approach. Level 2 represents the rationalization of the management of individual projects. At Level 3, standard processes are defined and used for the organization's entire portfolio of projects. Level 4 pushes rationalization even further, specifying mechanisms for quantifying the performance of the development process. Level 5 specifies systems for assuring the continuous improvement of that process. The philosophy underlying this hierarchy was inspired by Crosby's (1980) five stages of TQM maturity (Humphrey, 2002).

Early CMM assessments revealed a startlingly 'immature' state of software process: 80.3% of the 132 organizations assessed during 1987–1991 were found to be at Level 1, 12.1% at Level 2, with only 6.8% at Level 3, 1.4% at Level 4 and 0.8% at Level 5. Over the subsequent years, however, there appears to have been a significant shift, although it is difficult to tell given the changing and unrepresentative nature of the sample composed of organizations that volunteer for evaluation. Of the 1124 organizations assessed between 1998 and August 2002, 19.3% were at Level 1, 43.2% at Level 2, 23.4% at Level 3, 7.3% at Level 4 and 6.8% at Level 5 (Software Engineering Institute, 2004).

Research methods

The empirical research used in the present article was conducted in a large, US-based professional services firm, which I will call GCC. GCC was one of the largest software services firms in the world. Major players in this industry include Accenture, IBM, EDS, and CSC. In 2000, GCC's total sales exceeded $9 billion. It employed around 60,000 people. GCC had experienced double-digit annual revenue growth over most of the prior decade.

GCC was an innovation-oriented business. Most of the effort in the units I studied was devoted to the creation of new systems for their customers rather than maintaining existing systems, and these systems were often 'leading edge' in their complexity and sophistication. Over recent years, customer requirements had become more complex; new technologies and languages had been introduced at an accelerating rate; and the programs were pushed to show ever-greater flexibility in responding to new customer needs.

With the support of senior management, I conducted interviews with personnel in four 'programs' – I will call them A, B, C, and D – in GCC's Government Systems group during the course of 2002. Programs at GCC were organizational units devoted to long-standing, multi-project, client engagements. At the time of my research, Program A with a staff of 450 was at CMM Level 5; Program A's sister, Program B, with a staff of 275, was Level 3; Program C with a staff of 450 was almost Level 5 (it was certified Level 5 shortly after my study); and Program C's sister, Program D, with a staff of 470, was at Level 3. I interviewed 68 people at various hierarchical levels and in various functions in these four programs. Table 2 summarizes their roles. In the interview excerpts, these interviewees are identified by role, program affiliation, and identifying number. Note that unless otherwise indicated, roles are non-managerial. Interviewees listed as 'developers' worked in either systems engineering (developing system requirements) or software engineering (developing code to meet those requirements): unlike some other organizations, the personnel in these two groups had similar status and backgrounds. 'Managers' were responsible for specific functional departments, specific projects, or whole programs, as indicated.

Table 1. The Capability Maturity Model.

Level	Focus and description	Key process areas	Distribution of appraised organizations in:	
			1987–1991 (132 organizations)	2000–2004 (1543 organizations)
Level 1: Initial	**Competent people and heroics:** The software process is ad hoc, occasionally even chaotic. Few processes are defined, and success depends on individual effort and heroics.		80.0%	9.6%
Level 2: Repeatable	**Program management processes:** Basic program management processes are established to track cost, schedule, and functionality. The necessary process discipline is in place to repeat earlier successes on programs with similar applications.	* software configuration management * software quality assurance * software subcontract management * software project tracking and oversight * software project planning * requirements management	12.3%	42.6%
Level 3: Defined	**Engineering processes and organizational support:** The software process for both management and engineering activities is documented, standardized, and integrated into a standard software process for the organization. All programs use an approved, tailored version of the organization's standard software process for developing and maintaining software.	* peer reviews * intergroup coordination * software product engineering * integrated software management * training program * organization process definition * organization process focus	6.9%	30.1%
Level 4: Managed	**Product and process quality:** Detailed measures of the software process and product quality are collected. Both the software process and products are quantitatively understood and controlled.	* software quality management * quantitative process management	0.0%	8.6%

(Continued)

Table 1. (Continued)

Level	Focus and description	Key process areas	Distribution of appraised organizations in:	
			1987–1991 (132 organizations)	2000–2004 (1543 organizations)
Level 5: Optimizing	**Continuous process improvement:** Improvement is enabled by quantitative feedback from the process and from piloting innovative ideas and technologies.	* process change management * technology change management * defect prevention	0.8%	9.6%

Source: adapted from Paulk, 1995; Software Engineering Institute, 2004.

Table 2. Interviewees.

Role:	Interviews in programs:			
	A	B	C	D
System developers/system engineers/testers	6	7	4	5
Process engineers	3	2	2	2
Department managers	1	2	2	6
Project managers	3	1	3	0
Program managers	1	0	1	2
QA/CM	1	3	2	2
Training	1	0	1	0
Other support staff	1	0	0	4
Total interviewees	17	15	15	21
Total employee count	460	275	450	470

Interviews lasted approximately one hour. They were guided by a short, open-ended interview protocol that sought information on the interviewees' personal background, their work experience at GCC and how it compared to prior organizations, and their assessments of the GCC work process and the CMM. The interviews were tape-recorded and interviewees were assured anonymity. The recordings were transcribed, and edited versions were sent back to interviewees for review, correction, and further comment. I also consulted voluminous internal documentation from each of these programs as well as documents from corporate entities supporting them.

To put the interviews into comparative context, I relied on two types of data. First, two book-length studies of programmers describe in detail organization and work conditions in the earlier life of the industry (Greenbaum, 1979; Kraft, 1984). Second, many of my interviewees had worked in Level 1 or 2 organizations before joining GCC, and some of my interviewees had been with GCC long enough to recall conditions in GCC at similarly low CMM Levels.

The following four sections marshal this material to provide some illustrations of the key features of collaborative, *Genossenschaft* as it emerged in this context. They address in turn norms,

values, authority and capabilities. I then discuss the contradiction between collaboration and valorization and the evolution of the class struggle in this setting.

Norms: Synthesizing universalism and particularism

As the collective worker community becomes more socialized, the norms that shape its work practices are no longer those of a craft – particularistic, tacit, naturally emergent, grounded only in local experience, and mysterious to outsiders. Through formalization and standardization, these norms become universalistic, publicly available and debatable, social instead of private. However, under pressure to accommodate the uncertainties and context-sensitivity of the innovation process, these universalistic norms must be somehow synthesized with an appropriate degree of particularism. The following paragraphs trace the development of this synthesis at GCC.

In the early history of software, traditionalistic craft norms prevailed. Several GCC interviewees described the particularistic norms and the ad hoc organization of work that they experienced in Level 1 organizations prior to joining GCC. Their comments were consistent with Kraft's (1977, p. 56) characterization of programming in the 1960s: 'Programmers (and analysts) followed a logic and procedures which were largely of their own making'. Developers learned from their more senior colleagues the 'tricks of the trade'. There was a degree of collegiality here, but the collective worker was severely limited in its extent and in the scale and complexity of tasks it could take on. A veteran programmer described work in that period thus:

> No one knew what was going on – certainly not the managers. But even the programmers and systems analysts were confused. There were no standards for doing anything – coding, testing, documenting – they were all done the way each person felt like it, or in fact, they were not done at all. […] Programmers never documented what it was their program was to do. It was the same with setting up testing procedures and test data. When the whole system was put together, we never knew if it really worked because nothing got written down. (Greenbaum, 1979, pp. 73–77)

Early efforts to bring this unruly process under control led at first to universalistic bureaucratic norms – ones that many developers experienced as alienating and coercive (the assessments on this score by Kraft, 1977, and Greenbaum, 1979, are echoed by Beirne, Ramsay, & Panteli, 1998; Friedman & Cornford, 1989; Prasad, 1998). This seems to have been the case at GCC. Discussing the Military Standards for software quality control that come into force at Program C in the mid-1980s – an early form of formalization – one GCC veteran said: '[Military Standard] 2167A was supposed to make coding a no-brainer' (Development manager, D-20). On the civilian side too, the initial experience with a highly formalized development process was top-down, oriented to conformance: 'most managers felt that it was just a matter of ensuring that people were implementing it' (Program manager, A-11). The results were not very satisfactory from a labour-process or valorization-process point of view: these early steps afforded little relief from escalating error rates and schedule and budget overruns.

However, by the time of my study a decade or more later, formalized procedures had not only become more elaborate but had also evolved toward a synthesis of universalism and particularism. Concerning universalism: developers at GCC were aware that their effectiveness was not only the result of their own individual effort and skill and of informally shared tricks of the trade, but also and increasingly the result of a social, rather than private, accumulation of working knowledge embodied in formalized, standardized processes. In the words of one developer:

> When I got here I was kind of shocked. Right off, it was 'Here are your instructions'. 'So what does this tell me?' 'It tells you how to do your job.' I thought I was bringing the know-how I'd need to do my job.

> But sure enough, you open up the Instructions, and they tell you how to do your job: how to lay the code out, where on the form to write a change request number, and so on. I was shocked. But I can see the need now. Now I'm just one of 30 or 40 other people who may need to work on this code … Now I can see that it makes things much easier in the long run. … By the time we see the Instructions, they've been through a lot of revision and refinement. So they're pretty much on target. (Developer, C-13)

Concerning particularism: formalization and standardization of work norms were extensive, but served primarily as tools to guide work and to be adjusted to fit the particular circumstances, rather than as universal rules used to control employees, assumed to be recalcitrant and unreliable. Through a formalized 'Tailoring Cycle', software development standards and procedures (known as 'S&Ps' in Program A) were modified for each project with the participation of the developers themselves (on a similar process at Motorola, see Fitzgerald, Russo, & O'Kane, 2003):

> People have to be a part of defining the process. We always say that 'People support what they help create'. That's why the Tailoring Cycle is so important. As a project manager, you're too far away from the technical work to define the S&Ps yourself, so you have to involve … your key people. … It's only by involving them that you can be confident you have good S&Ps that have credibility in the eyes of their peers. (Project manager, A-8)

Tailoring allowed the formalized procedures to serve an 'enabling' rather than 'coercive' function (Adler & Borys, 1996). They thus were able to support rather than stifle innovation. This synthesis allowed the organization to bring more discipline to the innovation process without impairing its creativity:

> You'll discover that even in very innovative projects, most of the tasks are ones you've done many times before. Then, for the tasks that are truly novel, you can still leverage your prior experience by identifying somewhat related tasks and defining appropriate guidelines based on those similarities. (Test, B-9, formerly with Program A)

Values: Synthesizing individualism and collectivism

In the earlier history of software as described by Greenbaum, Kraft, and other scholars, programming resembled a craft in the lack of formalized, standardized techniques and in the widely shared ethos celebrating individual autonomy (Carmel, 1997). Greenbaum quotes one programmer:

> After you've been doing it for a while, coding gets boring. Especially after they divide up the project into so many modules that you don't know what you're doing relative to the whole system. So partly it's to preserve our sanity – we do things our own way and don't document it. Anyway, documenting is the most boring part of all. (1979, p. 75)

As noted above, early efforts to bring software production under greater managerial control and improve the reliability of the software production process took a classically bureaucratic form, imposing universalistic standards, defined by experts, on programmers who were now merely hired staff. Kraft and Greenbaum document the difficulties experienced by software organizations in maintaining any loyalty among increasingly alienated programming staff.

However, by the time of my interviews, many developers at GCC expressed very different values, representing a synthesis of individualism and collectivism:

> Developers want above all to deliver a great product, and the process helps us do that. What I've learned coming here is the value of a well thought-out process, rigorously implemented, and continuously improved.

> It will really improve the quality of the product. In this business, you've got to be exact, and the process ensures that we are. You have to get out of hacker mode! (Developer, A-14)

What mattered to these GCC developers' identity was now not so much their individual autonomy as their ability to contribute effectively to a shared purpose – the use-value of their product. One interviewee expressed it this way:

> Think of bridge-building. Back in the eighteenth century, there were some very beautiful bridges built, but quite a few of them collapsed because they were designed by artists without any engineering understanding. Software is like bridge-building. Software developers think of software as something of an art, and yes, you need that artistry, but you better have the engineering too. Developers often don't like the constraining rules, but the rules are necessary if you want to build complex things that have to work together. If you have only two or three people, you don't need all these rules. But if you have hundreds of people, the way we have here, then you need a lot of rules and discipline to get anything done. (Training, C-15)

Indeed, my interviews revealed a striking difference in tone between less mature programs, where individualistic values predominated, and more mature ones, where I found a synthesis of collectivistic concern for discipline and individualistic concern for creative contribution. This was evidenced in the way that 'we' often replaced 'I' as the subject of work in the more mature programs. Indeed, the ratio of mentions of 'we' to mentions of 'I' in my interview notes was 1.83 in Program A and 1.95 in Program C (the two Level 5 programs), and 1.29 in Program B and 1.44 in Program D (the two Level 3 programs). Interviewee B-7, a developer, presented an assessment that was particularly probative because her experience of a relatively mature process was recent:

> A more mature process means you go from freedom to do things your own way to being critiqued. It means going from chaos to structure. It's a bit like streetball versus NBA basketball. Streetball is roughhousing, showing off. You play for yourself rather than the team, and you do it for the love of the game. In professional basketball, you're part of a team, and you practice a lot together, doing drills and playing practice games. You aren't doing it just for yourself or even just for your team: there are other people involved […]. You have to take responsibility for other people – your teammates – and for mentoring other players coming up. (Developer, B-7)

Most strikingly, developers at GCC were not resentful of the documentation burden they shouldered – typically one of the loudest complaints of developers in less mature organizations (e.g. Hodgson, 2004). Documentation was now seen as a natural part of one's job, since that job was interdependent with others for whom the documentation would be essential. Developers' sense of collective interdependence extended to an imagined relationship with previous and future developers and with other people who are working on the code. Numerous interviewees offered assessments similar to this one:

> I think that our process – and even the paperwork part of it – is basically a good thing. My documentation is going to help the next person working on this code, either for testing or maintenance. And vice versa when I'm on the receiving end. (Developer, C-11)

Authority: Synthesizing hierarchy and participation

When programming resembled a craft, hierarchical control was limited, as we saw above, and software organizations had few specialized staff functions. In response to the productivity and quality problems created by the craft model, especially in larger projects, software organizations

attempted to assert hierarchical management control over methods and policies, creating new staff functions to design and impose these control systems. But this shift from *Gemeinschaft* to *Gesellschaft* did not yield the desired performance improvements. As a result, management's approach evolved towards a synthesis of hierarchy and participation:

> The first phase, in the late 1980s, was conformance. We had developed our standard process – a big fat set of requirements and standards – and most managers felt that it was just a matter of ensuring that people were implementing it. The second phase, in the early 1990s, was enlightenment. This phase coincided with our big TQM push. We started getting working level people involved in improving things. The third phase, running between about 1994 and 1998, was empowerment. The word might sound trite to some people, but we had the process framework, and we had the involvement, so we were really ready to delegate more autonomy down to the projects and the tasks. (Program manager, A-11)

This new, collaborative synthesis of hierarchy and participation was visible at GCC in four aspects of the authority structure: management style, policy-setting, decision-making, and staff roles.

Management style. Managers at GCC were acutely aware that autocratic styles of management would cripple the collaboration the labour-process needed:

> By and large, we haven't had too much difficulty bringing our managers around to this more collaborative approach. But we choose our project managers with an eye to their commitment to collaboration too. We did have a problem with one staff person. He had a very difficult relationship with the project people he was supposed to be helping. We got a lot of complaints that he was trying to force the projects to conform to his idea of how they should function. We tried to counsel him and get him to work in a more cooperative way. But he just wouldn't ease up. Eventually we just had to let him go. And we had quite a battle with one program manager when he wasn't picked to head a new project: we felt he just wasn't enough of a team manager. (Program manager, A-11)

Aware of this issue, management built systems to ensure that managers with coercive styles would be rapidly identified and their behaviour rectified:

> We didn't initially have any questions on the employee survey about your boss. Frankly, people were worried that managers might retaliate. But now we do, and we find the data very useful in surfacing management problems. The earlier rounds of the survey did show some big communications problems in some groups. Counselling often helped, and in some cases, we moved people out to other positions. (Program manager, A-11)

Policy-setting. All four GCC programs had process improvement teams that included rank-and-file developers. Participation in these teams was more widespread in the more mature (higher level) programs. In the Level 5 Program C, for example, over 19% of the total developer staff had been actively involved at some time in the course of the prior 12 months in either the Software Engineering Process Group (SEPG) or one of the various process improvement teams working in conjunction with it. As estimated by key managers I interviewed about this, the corresponding proportion for Program A, the other Level 5 program, was similar, whereas in the two Level 3 programs, the proportions were much lower – approximately half that of their more mature counterparts.

Decision-making. The socialization of the authority structure was also visible in the bottom-up influence in situational decision-making. Process provided superior-subordinate relations with objective points of reference outside the dyadic interpersonal relationship. Several interviewees argued that this gave the subordinate more participation and power. This excerpt illustrates:

> Before I came to GCC, I worked for one of the most autocratic managers you can find. It was always, 'And I want that report on my desk by 5 p.m. today', with no explanation or rationale. Compared to that kind of situation, an organization with a more mature process leaves a lot less room for a manager to arbitrarily dictate how you should work and when work is due. And a more mature process also means that there are more formal review points, so any arbitrary autocratic behaviour by a manager will become visible pretty quickly. (Program manager, D-5)

Staff functions. As GCC's process became more mature, new staff functions such as Configuration Management and Process Engineering emerged. Programs A and B had created specialized staff groups for Process Engineering (PE), while Programs C and D folded this activity into an enlarged Quality Assurance (QA) function. By collecting and analysing data from the line organization's project work, these staff units could identify best practices and package them into models and standards for use by the developers. The overall structural configuration was thus characterized by powerful, specialized staffs. However, these staffs worked in a largely supportive manner, rather than dictating requirements to the line organization as assumed in many accounts of bureaucracy.

QA illustrates the new staff/line relations. (For discussion of the impact of process maturity on the role on Configuration Management, see Butler, Standley, Sullivan, & Turner, 2001: many of the same conclusions emerge.) In the past, QA was often remote from the daily work of developers, arriving on the scene at the end of the work cycle to audit the output. Staff/line relations were notoriously antagonistic. QA's role evolved with process maturity to (a) a greater focus on process quality rather than only product quality, (b) greater responsibility for infusing process rather than only auditing it, and (c) a closer and more collaborative relation with the line departments. QA's role in the Tailoring Cycle is a good example:

> The process forces people out of their functional or module silos and into structured communication across those boundaries. QA, for example, gets a defined place in our reviews and our process improvement cycle. But QA is not a policeman! QA is there to help the project – help you identify the processes you need, tailor existing ones to your needs, learn that process, and do a check to see if you're using it. If I find a problem, it's my job to help the project work out how to address it and how I can help. (Quality assurance, B-5)

Capabilities: Synthesizing differentiation and integration

In the early, craft years of programming, task specialization and horizontal functional differentiation were very limited, and developers enjoyed high levels of autonomy, task variety, and task identity. Greenbaum (1979, pp. 64–65) quotes a veteran programmer thus:

> I remember that in the [nineteen-] fifties and early sixties I was a 'jack of all trades'. As a programmer I got to deal with the whole process. I would think through a problem, talk to the clients, write my own code, and operate the machine. I loved it – particularly the chance to see something through from beginning to end.

As software grew in scale and complexity, the division of labour became more complex, and coordination problems multiplied. Conflict was notoriously common in the relations between 'systems engineering' – the function responsible for analysing customer requirements – and 'software engineering' – responsible for translating those requirements into code. This conflict has often been construed by organizational researchers as reflecting Taylorism's split between conception and execution, since the former was often seen as more skilled, while the latter function was subject to efforts to simplify, deskill, and routinize (Greenbaum, 1979, pp. 68 ff.; Pettigrew, 1973).

In the subsequent years, as process became more mature, the status and skill levels of the two functions became more equal; there was considerable elaboration of various integrative mechanisms; and the coordination across groups became more rigorous and more collaborative. Mutual indifference or rivalry was replaced by active cooperation. The approach at Program C was typical of the intensity of coordination efforts and their collaborative form:

> We actively work this issue [cross-department coordination between systems engineering and design engineering] in a variety of ways. First with reviews: we now try to establish requirements peer reviews, which include at least one representative from software engineering, on our projects. Second, with some clarification mechanisms: in addition to the reviews, software engineers review the requirements as part of the receipt/estimation process. […] Third, with requirements detail: we've found that the level of specificity of requirements can vary significantly from author to author, even within the same larger systems engineering group. Not surprisingly, there are fewer communication/understanding issues when the requirements are more detailed and consistent. Fourth, with teamwork: we've actively tried to promote the enhanced communication through using Integrated Product Teams comprising system and software engineers and test engineers. This has worked well when the group leadership skills are strong, facilitating communication and resolution of issues. Fifth, by location: we've also used co-location of systems and software engineers. This has enabled quick and easy communication – chats over the cubicle wall. (Process engineer, D-14, describing Program C)

This transition – from reliance on the formal hierarchy for integrating differentiated subunits to a more active engagement of everyone in that integration task – was also visible in relations between programming and testing functions:

> Process means that people play more specialized, defined roles, but also that these specialists get involved earlier and longer as contributors to other people's tasks. If we analysed the way a coder uses their time, and compared it with comparable data from, say, 15 years ago, we'd find the coder doing less coding because of more automated tools. They'd be spending more time documenting their code, both as it was being built and afterwards in users' guides. They'd be spending more time in peer reviews. And they'd be spending more time in design meetings and test plan meetings. As for testers […] now the testers are more involved in system concept definition and requirement definition activities. (Quality assurance, A-3)

This synthesis of differentiation and integration was supported by performance measurement and incentive systems that were designed to encourage everyone to work toward the goals of *both* the organization as a whole and their subunit rather than *only* their subunit. Moreover, these goals were framed in *both* use-value and exchange-value terms rather than *only* in exchange-value terms. In less mature organizations, performance measures focused on subunit exchange-value variables such as cost and expected completion dates: these simplistic measures rewarded individual 'heroics' rather than collaboration and discipline. As the GCC program grew in maturity, performance measures (and the associated incentives and promotion opportunities) expanded to include a broader range of metrics designed to encourage alignment with best practices in software management and not only end-results, and rewarding performance at the individual, team, and organization levels (see also Chrissis, Konrad, & Shrum, 2011).

Collaboration versus valorization

The socialization of labour process community – driving it from traditionalistic *Gemeinschaft*, via contractual *Gesellschaft*, to collaborative *Genossenschaft* – is stimulated by valorization pressures characteristic of the prevailing capitalist relations of production, as shown in the previous section;

but valorization pressures at the same time limit and distort socialization and collaboration. The limiting and distorting effects reflect several factors: (a) the pursuit of economic profit sometimes conflicts with the pursuit of technical performance; (b) corporate interests sometimes undermine the cooperation required for the effective functioning of the collective worker; and (c) competitive rivalry between firms sometimes undermines their collaboration. I address these in turn.

First, GCC managers understood that process maturity required a high level of employee participation; however, the authority structure expressed not only a productive, coordination function but also the exploitative, wage relation. As a result, the authority structure was only partly aligned with the use-value requirements of software quality:

> As I see it, GCC is a corporation, and that means it's run for the benefit of the major stockholders. So top management is incentivized to maximize dollar profits. Quality is only a means to that end, and in practice, quality sometimes gets compromised. I used to be a technical person, so I know about quality. But now I'm a manager, and I'm under pressure to get the product out – come what may. I just don't have time to worry about the quality of the product. I have a manager of software development under me who's supposed to worry about that. (Development manager, D-20)

The contradiction between profit (exchange-value) and quality (use-value) was particularly visible to the interviewees in the form of missed opportunities for process improvement. Many expressed frustration with the gap between the discipline demanded by mature process and the funding that senior management made available for the support functions and IT tools required by such maturity:

> One key challenge [in pursuing process improvement] is maintaining buy-in at the top. Our top corporate management is under constant pressure from the stock market. The market is constantly looking at margins, but government business has slim margins. That doesn't leave much room for expenditures associated with process improvement – especially when these take two or three years to show any payoff. (Process engineer, C-14)

Second, the valorization imperative created a constant risk that managers would fall back on coercion. This aspect of the socialization/valorization contradiction was visible in the tension between, on the one hand, management's awareness of the importance of employee commitment and the firmness with which senior management treated instances of autocratic behaviour by managers, and on the other hand, the recurrent instances of coercion that escaped this control and called into play that firmness. The cases of a staff manager and a program manager were mentioned in an excerpt above; the case below is also instructive on the dilemmas facing management in this regard:

> We really can't afford an autocratic style of leadership. The risk of losing critical people is too high. […] We did have a pretty autocratic manager a while back in our software development organization. He had very strong technical skills and would often make decisions without consulting his staff. We heard a lot of complaints, and we saw some turnover too. But his technical skills made him very valuable to us, so we kept him on even after he offered to resign. We tried to get him to change his style, but he didn't, and eventually, after maybe two years of this, we just had to let him go. It was difficult. And he took a few loyal staff people with him too. (Program manager, D-5)

Third, the socialization of the labour process – particularly in this professional services sector – also involved a broadening of the collective worker across firms; but this productive exigency was in contradiction with the persistence of market competition at the core of the prevailing relations of production. This contradiction was particularly visible in Program B:

> The biggest problem here has been the customer and getting their buy-in. At Program A, our customer grew towards process maturity with us. Here [at Program B], we started with a less mature client. Some of the customer management even told us that they didn't want to hear about QA or our quality management system – they saw it as wasteful overhead. When you bid a project, you specify a budget for QA and so forth, but if they don't want to pay, you have a resource problem. […] On the Y2K project, the customer kept changing standards and deadlines. Basically, we were dealing with a pretty process-immature customer, and that made it difficult for us to build our process maturity. (Process engineer, B-13, formerly with Program A)

The process maturity effort within GCC also reflected this contradiction. On the one hand, pressures to conform to the CMM were sometimes helpful in prompting desired technical changes within GCC. On the other hand, part of the CMM effort was clearly 'ceremonial', and to that extent could lead to a decoupling between formal process and daily practice (as described by Meyer & Rowan, 1977). However, this contradiction, like the others just discussed, was not static – its form evolved over time as socialization progressed: while in the two Level 3 programs several interviewees commented that changing the process documentation did not always reflect changes in the way work was actually done, in the two Level 5 programs there were no such comments, even in response to probing in interviews, and despite the fact that the Level 5 requirements were far more comprehensive and detailed.

Program A illustrates how the structural features of capitalist relations of production undermined and limited collaboration. Due to unforeseen changes in their customer's priorities, Program A's volume of work had been reduced, and the valorization imperative forced GCC to cut its workforce from over 1600 to some 460. The result was a collapse of morale: the response rate to the annual employee attitude survey fell from an average of over 50% in prior years to 37% in 1997 and 15% in 1998–9. Even though these layoffs were managed under unusually compassionate policies, and even though these surveys revealed a very high rate of agreement (among the dwindling proportion of respondents) with the item 'I am treated fairly and understand why Program A is downsizing', engagement in process improvement activity declined. As one interviewee noted:

> It's hard to convince people that improving the process will help us get or keep business. We had a world-class process, and look what happened to us! Jobs in an organization like this depend a lot more on the vagaries of contracting than on our process excellence. (Department manager, A-6)

Deepening the contradiction, reshaping the class struggle

Viewed in longer perspective, the socialization/valorization contradiction evolved in form – deepening, rather than dissolving it.[4] In the earlier period of the software industry, valorization pressures stimulated efforts to replace craft with bureaucratic structures, as software organizations struggled to master the challenges of managing larger, more complex, software projects. However, these efforts were stymied by both technical and social factors. Technically, the industry lacked some key elements of the requisite technical infrastructure. They also lacked a viable management model because they saw rationalization in essentially coercive terms: as we saw in the excerpt quoted above, it 'was supposed to make coding a no-brainer'. Not surprisingly, software firms found their efforts blocked by programmers who saw this rationalization as a weapon against their autonomy. This is illustrated by Greenbaum's quote from a programmer in the 1970s:

> What kind of job security would we have if we wrote everything down the way they wanted us to? We didn't like it when things got too out of control, but on the other hand would you see to it that your job was so standardized that it could be done by a monkey? (Greenbaum, 1979, p. 75)

The result was that coercive bureaucracy largely failed to solve the intensifying software crisis.

In the more recent period, in contrast, the CMM was recast, and now functioned less as a weapon and more as a tool. As such, the collective software worker could embrace it as a way to master their collective task, and it appears from the excerpts I have quoted above that it was accepted in just this way by many developers at GCC. As a result, the nature of class conflict in the software production process shifted from those created by individualistic developers defending their autonomy to those created by the fundamental structure of the capitalist enterprise and its subordination to the profit imperative. As evidenced in several interview passages quoted above, developers were now increasingly conscious that the key issue facing them was not how to preserve their individual autonomy or craft control, but a deeper one – of how to deal with the fetters on productive advance created by the capitalist form of enterprise. This awareness was expressed from numerous interviewees, in comments such as this:

> We could do better at capturing and using lessons learned. We have all the vehicles for doing it – presentations, newsletters, databases. But it takes time. And there are so many competing priorities. In the end, it's all about profit and meeting schedules! (laughs) (Project manager, A-8)

Conclusion: Communism Developing in the Heart of Capitalism

This paper was motivated by the need to resolve two puzzles posed by the idea that community is important for innovation-oriented industry today. First, the traditionalism of *Gemeinschaft* community militates against innovation. Here, I argued that the form of community at work in at least some parts of innovation-oriented industry today is quite different from *Gemeinschaft*. Second, any idea of community seems to contradict what we know to be the conflictual character of the capitalist employment relation. Here, I argued that the contradiction was not a logical one, but a real one, evidenced in the tensions between an increasingly socialized labour process and the persistence of valorization pressures in the prevailing relations of production. My argument, in summary, has been that community is indeed developing in at least some sectors of industry, and that this community is taking an historically new form, one that represents a dialectical synthesis of *Gemeinschaft* and *Gesellschaft*, a form I proposed we might call *Genossenschaft*, or collaborative. I sketched the key features of this new form along four dimensions – norms, values, authority and capabilities – and illustrated these features with case data from a software services firm.

Obviously, some of these features are not entirely new: they can be seen in various cooperative endeavours that have historically been shielded from competitive and exploitative pressure, such as producer and consumer co-ops, artistic groups, independently funded research groups, and universities. My claim is that the collaborative type is now becoming more important within the heart of capitalist production, and that this is portentous.[5] Driven by the exigencies of capitalist production, collaborative community in the labour process has developed considerably compared with these other instances, embracing much larger and much more heterogeneous collectivities, and elaborating sophisticated systems for managing interdependent processes. And with its appearance within capitalist production, there is now a far more powerful force driving its diffusion, all the while limiting its development too.

I conclude by suggesting that this new form represents communism developing in the heart of capitalism. Insofar as the collective worker takes the collaborative, *Genossenschaft* form, it prefigures the communist ideal of a 'free association of producers'. This bald assertion may seem at first sight an overenthusiastic extrapolation; however, it is difficult to see how the communist ideal would differ in any substantial way from the norms and values whose contours I have just sketched. On the other hand, however, progress towards communist authority and capabilities is much

slower. As concerns authority under communism, managers – insofar as a specialized management function would be needed in larger, more complex organizations – would surely be elected or rotated: at GCC, managers needed some real endorsement from below, but that is still a long way from election or rotation. As concerns the capabilities dimension: Marx suggested that communism would be based on the principle of 'from each according to their abilities, to each according to their needs' (Marx & Engels, 1959), but the GCC case reflected no such commitment. Notwithstanding this and other gaps, collaborative community, as we see it emerging tentatively in industry today, can be said to embody a rough sketch of a future society, and moreover, this tentative emergence appears to help create the new interconnections and new understandings that could help workers in their efforts to create that future.

The emergence of collaborative community in the collective worker coexists in a real contradiction with the persistence of the valorization imperative characteristic of capitalist relations of production. We therefore rarely see it in pure form, and where we do see it, it is typically fragmentary and corrupted by valorization pressures, and always precarious because it is constantly being undone by exploitation, market competition, and the concomitant periodic crises. Indeed, in real social structures, whether at a societal or an organizational level, community today is often most salient by its painful absence. Nevertheless, as the labour process becomes more complex and interdependent – as it becomes more 'socialized' – community in the collective worker evolves and the collaborative form of community begins to take shape.

Community among workers is forged, of course, not only in the labour process, but also in struggles prompted by the conflicting interests at the heart of capitalist relations of production, as collective solidarity in opposition to exploitation. As Marx and Engels argue in the *Communist Manifesto*, capitalist development brings workers together in larger, more integrated work processes; given the exploitative nature of relations of production, they attempt to form unions and parties to defend their interests; and the advancing forces of production in communications technology facilitate this struggle. However, labour-process community and class-struggle community are intertwined: changes in the form of labour-process community encourage a similar shift in the form of class-struggle community. Progressives often lament the ongoing demise of unions in so many countries; I share this dismay, but I would also argue that these unions have often been based on *Gemeinschaft* bonds of loyalty. And they have often been, as *Gemeinschaft* usually is, bound by common location in firms, which makes them ill-suited to a more mobile and globalized labour process. Or they have become *Gesellschaft* associations, merely instrumental for individual protection, and even less effective as instruments of class solidarity. New forms of worker association are needed to meet today's challenges – forms that are more truly collaborative in nature: workers' experience of collaborative community in the labour-process might encourage new, collaborative forms of organization in the class struggle. 'Social movement' types of unionism might be understood as experiments of this kind (Seidman, 2011).

In both the labour-process and the class struggle, the collaborative community hypothesis seems consonant with Marx's assertion that:

> [I]f we did not find concealed in society as it is the material conditions of production and the corresponding relations of exchange prerequisite for a classless society, then all attempts to explode it would be quixotic. (Marx, 1973, p. 159)

It would require a separate paper to explore the paths by which these attempts to explode the capitalist form of society could lead to a more advanced form. Nothing in the argument presented here indicates whether this path would be short or long, gradual and peaceful or more abrupt and perhaps violent. But the emergence of collaborative community in the labour-process does appear to strengthen some of the enabling conditions for some such transition.

Acknowledgements

The arguments presented here were developed through a lengthy collaboration and dialogue with Charles Heckscher. Matt Vidal, Rick Delbridge, Paul Edwards, and other participants in the EGOS subtheme on Marxist organization studies offered valuable critiques along the way. I thank Iris Bosa for bringing the term Genossenchaft to my attention.

Funding

This research received no specific grant from any funding agency in the public, commercial, or not-for-profit sectors.

Notes

1. This dismissive view ignores at least five forms in which community operates in capitalist social formations: (a) pre-capitalist bonds of community that persist in some neighbourhoods, ethnic groups, and extended families; (b) new forms of community that emerge around national and religious identities; (c) community as forged in workers' and social movements' struggles against various forms of exploitation and domination; (d) the emergence of community in supportive and creative activities outside the direct control of capital; and (e) community within the capitalist labour process itself. More recently, writers with Marxist roots have addressed (a) and (b) (DeFilippis, Fisher, & Shragge, 2006); and a burgeoning literature in the 'post-workerist' spirit such as Hardt and Negri (2009) addresses (c) and (d); my argument focuses on (e). Note that Burawoy's article designates this missing object not as 'community' but 'society'; that latter term, however, is too broad, because society is often taken to mean the entire social system, inclusive of state, economy, civil society, and family. Burawoy uses 'civil society' as a synonym, but the latter too is confusing: for some writers, it includes the economic sphere, while for others it is distinct from both state and economy, and there is debate too over whether it includes the family or only refers to the sphere of public life. The term *community* may be unwieldy because of its manifold connotations and denotations, but it seems to be the best we have.
2. The more conventional reading interprets Marx's chapter in *Capital* on cooperation (1977, Ch. 13) as addressing a phase of industrial development that passes with the emergence of manufacture (Thompson, 1989, pp. 43–44). I read Marx as arguing that key features of this phase are enduring, structural features of the labour process: Marx writes that cooperation 'constitutes the starting point of capitalist production. This is true both historically and conceptually' (Marx, 1977, p. 439).
3. Marx writes: '[T]he communal relationship into which the individuals of a class entered, and which was determined by their common interests over against a third party, was always a community to which these individuals belonged only as average individuals, only insofar as they lived within the conditions of existence of their class – a relationship in which they participated not as individuals but as members of a class. With the community of revolutionary proletarians, on the other hand, who take their conditions of existence and those of all members of society under their control, it is just the reverse; it is as individuals that the individuals participate in it' (Marx & Engels, 1970. Pt. I, D).
4. I interpret Marx's discussion of the shift from formal to real subordination of labour as historically progressive, deepening rather than weakening the basic contradiction between forces and relations of production. In contrast, the conventional reading of Marx sees the basic contradiction not between forces and relations of production, but between classes, and sees the passage to real subordination as deepening that contradiction only in the sense that workers are even more ferociously opposed to their exploitation, now that this exploitation has stripped all dignity from their experience of work (Thompson, 1989, p. 52).
5. My argument is thus different from the 'post-workerist' argument that new forms of community are proliferating *outside* the realm of capitalist production (including the 'multitude' (Hardt & Negri, 2005), which displays some similarities with my collaborative form in its synthesis of individualism and collectivism). My argument also differs from theirs in that I follow Marx in assuming that surplus-value itself is only produced *within* the capitalist employment relation, even if this production relies – perhaps more than ever – on use-values (family care, freely-shared knowledge, land, air, water, etc.) appropriated from outside that production process (Smith, 2008).

References

Adler, P. S. (2001). Market, hierarchy, and trust: The knowledge economy and the future of capitalism. *Organization Science*, *12*(2), 215–234.

Adler, P. S. (2007). The future of critical management studies: A paleo-Marxist critique of labour process theory. *Organization Studies*, *28*(9), 1313–1345.

Adler, P. S., & Borys, B. (1996). Two types of bureaucracy: Enabling and coercive. *Administrative Science Quarterly*, *41*(1), 61–89.

Adler, P. S., Kwon, S., & Heckscher, C. (2008). Professional work: The emergence of collaborative community. *Organization Science*, *19*(2), 359–376.

Alvesson, M., & Thompson, P. (2006). Post-bureaucracy. In S. Ackroyd, R. Batt, P. Thompson, & P. S. Tolbert (Eds.), *The Oxford handbook of work and organization* (pp. 485–507). New York: Oxford University Press.

Beirne, M., Ramsay, H., & Panteli, A. (1998). Developments in computing work: Control and contradiction in the software labour process. In P. Thompson & C. Warhurst (Eds.), *Workplaces of the Future* (pp. 142–162). Houndsmill: Macmillan.

Benkler, Y. (2006). *The wealth of networks: How social production transforms markets and freedom*. New Haven: Yale University Press.

Boltanski, L., & Chiapello, E. (2005). *The new spirit of capitalism*. London: Verso.

Braverman, H. (1974). *Labor and monopoly capital: The degradation of work in the twentieth century*. New York: Monthly Review.

Brint, S. (2001). Gemeinschaft revisited: A critique and reconstruction of the community concept. *Sociological Theory*, *19*(1), 1–23.

Brown, J. S., & Duguid, P. (2001). Knowledge and organization: A social-practice perspective. *Organization Science*, *12*(2), 198–213.

Burawoy, M. (2003). For a sociological Marxism: The complementary convergence of Antonio Gramsci and Karl Polanyi. *Politics & Society*, *31*(2), 193–261.

Burrell, G., & Morgan, G. (1979). *Sociological paradigms and organizational analysis*. London: Tavistock.

Butler, T., Standley, V., Sullivan, E., & Turner, F. (2001). Software configuration management: A discipline with added value. Available at: https://http://www.stormingmedia.us/74/7486/A748684.html (accessed 17 July 2006).

Calhoun, C. (1998). Community without propinquity revisited: Communications technology and the transformation of the urban public sphere. *Sociological Inquiry*, *68*(3), 373–397.

Carmel, E. (1997). American hegemony in packaged software trade and the 'culture of software'. *The Information Society*, *13*(1), 125–142.

Castells, M. (2011). *The rise of the network society: The information age: Economy, society, and culture* (Vol. 1). London: Wiley-Blackwell.

Chatman, J. A., & Flynn, F. J. (2001). The influence of demographic heterogeneity on the emergence and consequences of cooperative norms in work teams. *Academy of Management Journal*, *44*(5), 956–974.

Chrissis, M. B., Konrad, M., & Shrum, S. (2011). *CMMI for Development: Guidelines for Process Integration and Product Improvement*. Boston, MA: Pearson.

Cohen, G. A. (1974). Marx's dialectic of labor. *Philosophy and Public Affairs*, *3*(3), 235–261.

Cohendet, P., & Simon, L. (2008). Knowledge intensive firms, communities and creative cities. In A. Amin & J. Roberts (Eds.), *Community, economic creativity and organisation* (pp. 227–253). New York: Oxford University Press.

Contu, A., & Willmott, H. (2003). Re-embedding situatedness: The importance of power relations in learning theory. *Organization Science*, *14*(3), 283–296.

Crosby, P. (1980). *Quality is free*. New York: McGraw-Hill.

d'Iribarne, P. (2003). The combination of strategic games and moral community in the functioning of firms. *Organization Studies*, *24*(8), 1283–1307.

De Dreu, C.K. W., & West, M. A. (2001). Minority dissent and team innovation: the importance of participation in decision making. *Journal of Applied Psychology*, *86*(6), 1191.

DeFilippis, J., Fisher, R., & Shragge, E. (2006). Neither romance nor regulation: Re-evaluating community. *International Journal of Urban and Regional Research*, *30*(3), 673–689.

Delanty, G. (2003). *Community: Key ideas*. London: Routledge.
Delbridge, R. (2007). Explaining conflicted collaboration: A critical realist approach to hegemony. *Organization Studies*, *28*(9), 1347–1357.
DiTomaso, N., Post, C., & Parks-Yancy, R. (2007). Workforce diversity and inequality: Power, status, and numbers. *Annual Review of Sociology*, *33*, 473–501.
Djelic, M. L., & Quack, S. (2010). Transnational communities and governance. In M. L. Djelic & S. Quack (Eds.), *Transnational communities: Shaping global economic governance* (pp. 3–36). Cambridge: Cambridge University Press.
Doel, C. (1999). Towards a supply-chain community? Insights from governance processes in the food industry. *Environment and Planning A*, *31*(1), 69–85.
Edwards, R. C. (1979). *Contested terrain: The transformation of the workplace in the twentieth century*. New York: Basic.
Eisenberger, R., Stinglhamber, F., Vandenberghe, C., Sucharski, I. L., & Rhoades, L. (2002). Perceived supervisor support: Contributions to perceived organizational support and employee retention. *Journal of Applied Psychology*, *87*(3), 565.
Engels, F. (1978). Socialism: Utopian and scientific. In R. C. Tucker (Ed.), *The Marx-Engels reader* (pp. 683–717). New York: Norton.
Engeström, Y. (1987). *Learning by expanding: An activity-theoretical approach to development research*. Helskinki: Orienta-Konsultit.
Fitzgerald, B., Russo, N., & O'Kane, T. (2003). Software development method tailoring at Motorola. *Communications of the ACM*, *46*(4), 64–70.
Friedman, A., & Cornford, D. (1989). *Computer systems development: History, organization and implementation*: New York: John Wiley & Sons.
Gibbs, W. (1994). Software's chronic crisis. *Scientific American*, *271*(3), 72–81.
Gläser, J. (2001). Producing communities as a theoretical challenge. *Proceedings of the Australian Sociological Association*, 1–11.
Gramsci, A. (1971). *Selections from the prison notebooks of Antonio Gramsci*. London: Lawrence and Wishart.
Granovetter, M. (1982). The strength of weak ties: A network theory revisited. In P. V. Marsden & N. Lin (Eds.), *Social structure and network analysis* (pp. 105–130). Beverly Hills: SAGE.
Grant, R. M., & Baden-Fuller, C. (2000). Knowledge and economic organization: An application to the analysis of interfirm collaboration. In G. von Krogh, I. Nonaka, & T. Nishiguchi (Eds.), *Knowledge creation: A source of value* (pp. 113–150). Basingstoke: Macmillan.
Greenbaum, J. M. (1979). *In the name of efficiency*. Philadelphia, PA: Temple University Press.
Gulley, N., & Lakhani, K. (2010). The determinants of individual performance and collective value in private-collective software innovation. *HBS Working Paper 10–065*.
Haas, P. (2009). Introduction: Epistemic communities and international policy coordination. *International Organization*, *46*(01), 1–35.
Hardt, M., & Negri, A. (2005). *Multitude: War and democracy in the age of empire*: New York: Penguin.
Hardt, M., & Negri, A. (2009). *Commonwealth*. Cambridge, MA: Harvard University Press.
Heckscher, C. C. (1996). *White-collar blues: Management loyalties in an age of corporate restructuring*. New York: Basic Books.
Heckscher, C. (2007). *The collaborative rnterprise*. New Haven, CT: Yale University Press.
Heckscher, C., & Adler, P. (2006). *The firm as a collaborative community: Reconstructing trust in the knowledge economy*. New York: Oxford University Press.
Hillery, G. A. (1955). Definitions of community: Areas of agreement. *Rural Sociology*, *20*(2), 111–123.
Hodgson, D. (2004). Project work: The legacy of bureaucratic control in the post-bureaucratic organization. *Organization*, *11*(1), 81–100.
Humphrey, W. (2002). Three process perspectives: Organizations, teams, and people. *Annals of Software Engineering*, *14*(1), 39–72.
Ingvaldsen, J. (2015). Organizational learning: Bringing the forces of production back in. *Organizaiton Studies*.
Jacoby, S. M., & Taras, D. G. (1997). *Modern manors: Welfare capitalism since the New Deal*: London: Wiley Online Library.

Jehn, K., Greer, L., Levine, S., & Szulanski, G. (2008). The effects of conflict types, dimensions, and emergent states on group outcomes. *Group Decision and Negotiation*, *17*(6), 465–495.
Jones, C. (2002). Defense software development in evolution. *Crosstalk*, 26–29.
Keller, R. T. (1997). Job involvement and organizational commitment as longitudinal predictors of job performance: A study of scientists and engineers. *Journal of Applied Psychology*, *82*(4), 539–545.
Kraft, P. (1984). *Programmers and managers: The routinization of computer programming in the United States*. New York: Springer-Verlag.
Kruckeberg, D., & Starck, K. (2004). The role and ethics of community building for consumer products and services. *Journal of Promotion Management, 10*(1–2), 133–146.
Lieberman, H., & Fry, C. (2001). Will software ever work? *Communications of the ACM*, *44*(3), 122–124.
Lindkvist, L. (2005). Knowledge communities and knowledge collectivities: A typology of knowledge work in groups. *Journal of Management Studies*, *42*(6), 1189–1210. DOI: 10.1111/j.1467–6486.2005.00538.x.
Mahowald, M. B. (1973). Marx's Gemeinschaft: Another interpretation. *Philosophy and Phenomenological Research*, *33*(4), 472–488.
Marx, K. (1971). *A contribution to the critique of political economy*. London: Lawrence and Wishart.
Marx, K. (1973). *Grundrisse: Foundations of the critique of political economy* (M. Nicolaus, Trans.). Harmondsworth: Penguin.
Marx, K. (1977). *Capital, Vol. 1*. New York: Vintage.
Marx, K. (1989). First draft of letter to Vera Zasulich (written March 1881). In *Marx/Engels collected works* (Vol. 24, p. 346). New York: Progress Publishers.
Marx, K., & Engels, F. (1959). *Critique of the Gotha programme*. Moscow: Foreign Languages Publishing House.
Marx, K., & Engels, F. (1970). *The German ideology, Vol. 1*. Moscow: International Publishers Co.
McAlexander, J. H., Schouten, J. W., & Koenig, H. F. (2002). Building brand community. *The Journal of Marketing*, *66*(1), 38–54.
Megill, K. A. (1970). The community in Marx's philosophy. *Philosophy and Phenomenological Research*, *30*, 382–393.
Meiksins, P., Smith, C., & Berner, B. (1996). *Engineering labour: Technical workers in comparative perspective*. London: Verso Books.
Meyer, J. W., & Rowan, B. (1977). Institutionalized organizations: Formal structure as myth and ceremony. *American Journal of Sociology*, *83*, 340–363.
Morrison, P. D., Roberts, J. H., & Von Hippel, E. (2000). Determinants of user innovation and innovation sharing in a local market. *Management Science*, *46*(12), 1513–1527.
Muniz Jr, A. M., & O'Guinn, T. C. (2001). Brand community. *Journal of Consumer Research*, *27*(4), 412–432.
Ng, T. W. H., Feldman, D. C., & Lam, S. S. K. (2010). Psychological contract breaches, organizational commitment, and innovation-related behaviors: A latent growth modeling approach. *Journal of Applied Psychology*, *95*(4), 744.
Nooteboom, B. (2008). Cognitive distance in and between COPs and firms: Where do exploitation and exploration take place, and how are they connected? In A. Amin & J. Roberts (Eds.), *Community, economic creativity, and organization* (pp. 123–147). New York: Oxford University Press.
O'Mahony, S., & Lakhani, K. R. (2011). Organizations in the shadow of communities. *Research in the Sociology of Organizations*, *33*, 3–35.
O'Reilly, C. A., III, Chatman, J., & Caldwell, D. F. (1991). People and organizational culture: A profile comparison approach to assessing person-organization fit. *Academy of Management Journal*, *34*(3), 487–516.
Ouchi, W. G. (1980). Markets, bureaucracies, and clans. *Administrative Science Quarterly*, *25*(1), 129–141.
Ouchi, W., & Barney, J. B. (2004). An interview with William Ouchi. *The Academy of Management Executive (1993–2005)*, 108–116.
Page, S. (2008). *The difference: How the power of diversity creates better groups, firms, schools, and societies*. Princeton, NJ: Princeton University Press.
Paulk, M. C. (1995). The evolution of the SEI's capability maturity model for software, software process: Improvement and practice. *Software Process: Improvement and Practice*, *1*, 3–15.

Pettigrew, A. (1973). Occupational specialization as an emergent process. *Sociological Review*, *21*(2), 255–278.
Powell, W. W. (1989). Neither market nor hierarchy: Network forms of organization. In B. Staw & L. Cummings (Eds.), *Research in organizational behavior* (Vol. 12). Greenwich, CT: JAI Press.
Prasad, M. (1998). International capital on 'silicon plateau': Work and control in India's computer industry. *Social Forces*, *77*(2), 429–452.
Reed, M. I. (2001). Organization, trust and control: a realist analysis. *Organization Studies*, *22*(2), 201–228.
Ren, Y., Kraut, R., & Kiesler, S. (2007). Applying common identity and bond theory to design of online communities. *Organization Studies*, *28*(3), 377–408.
Roethlisberger, F. J., & Dickson, W. J. (1939). *Management and the worker*. Cambridge, MA: Harvard University Press.
Rudolph, L. I., & Rudolph, S. H. (1979). Authority and power in bureaucratic and patrimonial administration: A revisionist interpretation of Weber on bureaucracy. *World Politics*, *31*(2), 195–227.
Ruef, M. (2002). Strong ties, weak ties and islands: Structural and cultural predictors of organizational innovation. *Industrial and Corporate Change*, *11*(3), 427–449.
Sayer, D. (1990). *Capitalism and modernity: An excursus on Marx and Weber*: London: Routledge.
Seidman, G. (2011). Social movement unionism: From description to exhortation. *South African Review of Sociology*, *42*(3), 94–102.
Sennett, R. (2012). *Together: The rituals, pleasures and politics of cooperation*. New Haven, CT: Yale University Press.
Shang, R. A., Chen, Y. C., & Liao, H. J. (2006). The value of participation in virtual consumer communities on brand loyalty. *Internet Research*, *16*(4), 398–418.
Sheth, J. N., & Sharma, A. (1997). Supplier relationships: emerging issues and challenges. *Industrial Marketing Management*, *26*(2), 91–100.
Smith, C. (1987). *Technical workers: Class, labour and trade unionism*. Basingstoke: Macmillan Education.
Smith, T. (2008). *The 'general intellect' in the Grundrisse and beyond*. Paper presented at the 'Reading the Grundrisse' conference, University of Bergamo, Italy, 14–18 July.
Software Engineering Institute. (2004). *Process maturity profile of the software community*, 2004 mid-year update.
Springborg, P. (1986). Politics, primordialism, and orientalism: Marx, Aristotle, and the myth of the Gemeinschaft. *The American Political Science Review*, *80*(1), 185–211.
Standish Group. (1994). Chaos study report. Available at: http://www.standishgroup.com
Taylor, F. W. (1972). *Scientific management: Comprising shop management, the principles of scientific management [and] testimony before the Special House Committee*. Westport, CT: Greenwood Press.
Thompson, P. (1989). *The Nature of work, 2nd edn*. London: Macmillan.
Thompson, P. (2003). Disconnected capitalism: Or why employers can't keep their side of the bargain. *Work Employment Society*, *17*(2), 359–378. DOI: 10.1177/0950017003017002007.
Tönnies, F. (1957). *Community and society*. New York: Harper & Row.
Van Knippenberg, D., De Dreu, C., & Homan, A. (2004). Work group diversity and group performance: An integrative model and research agenda. *Journal of Applied Psychology*, *89*(6), 1008–1022.
Vidal, M. (2011). Reworking postfordism: Labor process versus employment relations. *Sociology Compass*, *5*(4), 273–286.
Vidal, M. (2013). Low-autonomy work and bad jobs in postfordist capitalism. *Human Relations*, *66*(4), 587–612.
Von Hippel, E. (2005). *Democratizing innovation*. Cambridge, MA: MIT Press.
West, J., & Lakhani, K. R. (2008). Getting clear about communities in open innovation. *Industry and Innovation*, *15*(2), 223–231.
Williams, R. (1985). *Keywords: A vocabulary of culture and society*. New York: Oxford University Press.
Wright, E. O. (1985). *Classes*. London: Verso.

Author biography

Paul S. Adler is currently Harold Quinton Chair in Business Policy at the Marshall School of Business, University of Southern California. His research focuses on organization theory and business/government/society interactions.