Reconfiguring Educational Expertise: MOOCs, Computer Scientists, and the Study of Learning

Abstract

In recent years, computer scientists have drawn on the rise of the Internet to turn their attention and technical expertise to activities usually imagined as strictly "social" by designing software infrastructures for brokering services (e.g. AirBnB, Uber), the distribution of cultural products (e.g. Netflix, Hulu), and new forms of paid labor (e.g. Amazon Mechanical Turk); they also seek to become social science knowledge producers and have called in recent years for a new kind of "computational social science" that draws on "big data". Building on two years of ethnographic fieldwork with computer scientists studying learning using the infrastructures built for Massive Open Online Courses (MOOCs), I describe how computer scientists apply their (historically specific) technical ideas to particular domains (here: learning and education) and how, through their technical practices and social imaginaries, they negotiate with other actors in that domain—a process that accounts for their rising expert authority and has significant political implications. Technically, computer scientists turn a domain into one that they are already familiar with by deploying the culturally flexible categories of "software" and "the user." They configure existing actors in the domain as either *domain experts*, to whom they defer to, or *users*, to be empowered, and often, both. Computer scientists aim to encode some of the knowledge of these domain experts into the software they build but their historically specific conception of human-machine interaction—of humans and programs operating in synchrony with programs being supervised by humans—means that these programs are supervised by the users of the system; their function is to empower these users (rather than replace them) with the users performing the significant work of "customizing" these programs. This process—to be understood as a kind of "boundary work" (Gieryn 1995)—allows computer scientists to enter new domains without specifying the nature of their own expertise, as simply neutral system-builders. By doing so, they are able to expand their jurisdiction (Abbott 1988), increase their social authority, make knowledge claims, and also arguably effect a change in long-standing domains and cultural categories (like "learning").

Introduction

In the summer of 2011, Juho Kim, then in the first year of his computer science PhD at MIT, went off to do an internship at Adobe's Creative Technology Labs in Silicon Valley and experienced something like a revelation. Juho had come to MIT with an interest in "supporting people's creative tasks with better-designed tools and processes"; for him, creative tasks meant "writing, programming, designing, even doing research." His special interest was in building "crowdsourcing" workflows that aided people in performing these creative tasks.[1] He expected that his time at Adobe would be spent

[1] Crowdsourcing is a form of organization where "microtasks" performed by anonymous individuals are aggregated by intelligent programs to produce useful outputs, and has been used to perform tasks like transcribing audio, object recognition in images, etc.

designing a new feature for Photoshop, Adobe's signature tool for image processing, used extensively by artists, graphic designers and other design professionals.[2] To his surprise, when he reached Adobe, he found that building new features was not a top priority; Photoshop already had hundreds of features and even Photoshop experts used less than half of them, while still managing to accomplish all sorts of intricate tasks. No, Adobe was interested in making Photoshop *easier to use,* or as Juho put it, "more learnable." As a research project, Juho built a chat-room for Adobe, enabled with voice and screensharing, where novice Photoshop users could ask questions of expert Photoshop users (conceived as a crowd)—"connecting people to give peer(-to-peer) help in using Photoshop," as he put it (Kim et al. 2012). The project satisfied all of his interests: helping creative people by building tools that deployed crowdsourcing methods innovatively. But it also led to something like a personal research revelation. He had found a "missing space" where computer science methods could intervene: "how can we make novices pick up those skills that experts already have in a more, you know, seamless manner?" Juho saw novice Photoshop users as "learners," expert Photoshop users as instructors, and the process of connecting the two roles—to help novices pick up expert skills—as a crowdsourcing problem. He had found a new domain of intervention for his research that combined all his interests: learning.[3]

That same summer Juho Kim was experiencing his revelation, other, more tumultuous, related events were happening nearby at Stanford University. In July 2011, Stanford announced that it would be putting three of its classes taught by their top computer science faculty online: on Artificial Intelligence, Machine Learning and Databases. While seemingly a sudden development, the decision to open up these classes came out of a long experimentation by these faculty to enhance their own teaching for Stanford undergraduates by building new kinds of software. The Stanford faculty were very inspired by new Internet platforms with vague to explicit educational agendas: YouTube, Khan Academy, lynda.com, StackOverflow, among others (see Ng and Widom 2014 for a narrative). Thousands signed up for these

[2] The extent of Photoshop's market dominance can be seen from the fact that the Merriam-Webster dictionary defines the verb "photoshop" as "to alter (a digital image) with Photoshop software or other image-editing software especially in a way that distorts reality (as for deliberately deceptive purposes)."

[3] Interview with Juho Kim,16 May 2014.

classes which consisted of lecture videos interlaced with interactive assessments; instructors guided the learners through these classes, often interacting with them in the student forums, although the grading itself was fully automated. Around 43,000 certificates were given out for learners who finished a class. These classes ended up being called MOOCs or Massive Open Online Courses. Inspired by Stanford, MIT started its own MOOC on Circuits and Electronics in the spring of 2012. In the summer of 2012, three new start-up companies to offer MOOCs existed: Udacity, Coursera, and edX, all of them led by computer scientists from MIT and Stanford, prompting the New York Times to declare 2012 to be the "Year of the MOOC" (Pappano 2012).

These two events above are related, most obviously, because they involve elite computer scientists. But the connections go deeper than that. In the first story, a computer scientist-in-training, inspired by new ways of connecting peers found on Internet platforms, used it to connect novice and expert Photoshop uses, and thereby found a new domain for his research: the study of learning—a "missing space" where innovative computer science methods like crowdsourcing could be applied. In the second, a group of already established computer scientists with close ties to Silicon Valley were inspired to take their existing *formal* classes and transform them into entities that resembled, in look, feel, and working, certain nominally educational sites of *informal* learning like YouTube and StackOverflow. The infrastructures created by these established computer scientists (i.e. MOOCs) became sites of research for young researchers like Juho Kim who were inclined to apply new forms of interaction on the Web (such as crowdsourcing and video) to both formal and non-formal learning situations. The result is the emergence of a new form of expertise in the study of learning.

In this paper, I examine the social imaginaries (Taylor 2003) and work practices of researchers like Juho Kim: computer scientists who design learning infrastructures and conduct research on learning, often through MOOCs. Building on two years of ethnographic fieldwork such researchers, I describe how computer scientists apply their (historically specific) technical ideas to particular domains (here: learning and education) and how, through their technical practices and social imaginaries, they negotiate with other actors in that domain—a process that accounts for their rising expert authority and has

significant political implications. Technically, computer scientists turn a domain into one that they are already familiar with by deploying the culturally flexible categories of "software" and "the user." They configure existing actors in the domain as either *domain experts*, to whom they defer to, or *users*, to be empowered, and often, both. Computer scientists aim to encode some of the knowledge of these domain experts into the software they build but their historically specific conception of human-machine interaction—of humans and programs operating in synchrony with programs being supervised by humans—means that these programs are supervised by the users of the system; their function is to empower these users (rather than replace them) with the users performing the significant work of "customizing" these programs. This process—to be understood as a kind of "boundary work" (Gieryn 1995)—allows computer scientists to enter new domains without specifying the nature of their own expertise, as simply neutral system-builders. By doing so, they are able to expand their jurisdiction (Abbott 1988), increase their social authority, make knowledge claims, and also arguably effect a change in long-standing domains and cultural categories (like "learning").

This paper therefore takes part in ongoing discussions about the nature of expertise (Boyer 2005; Carr 2010; Collins 2004; Collins and Evans 2008) in anthropology, sociology and science and technology studies (STS). In particular, my findings align with Gil Eyal's (2013) recent call to complement, if not replace, the sociology of professions with a sociology of expertise. Eyal argues for shifting attention away from experts to expertise itself; he defines expertise as the confluence of technical and institutional arrangements that form the conditions of possibility for routine expert utterances and actions; arrangements that may not necessarily be *just* the result of jurisdictional contests (Abbott 1988) between different experts as the sociology of professions has presupposed but also the contingent product of the stabilization of actor-networks (Latour 1987) mobilized by certain involved actors, experts and non-experts alike. This paper, for instance, demonstrates that the organizing principles of infrastructures—and who gets to shape them—are crucial to understanding which experts have the necessary skills and access to intervene in these infrastructures. In the particular case-study documented in this paper, it is computer scientists themselves who have invented and built the infrastructures that allow them to intervene in

multiple domains by configuring existing experts—in this case, a group known as "learning scientists"—as domain experts, and existing tasks (e.g. grading, peer assessment) as just examples of other tasks that they control (e.g. crowdsourcing); however these framings, as I show, draw resistance and redefinition from existing experts. However, the infrastructures alone are not enough; computer scientists also draw on two broader cultural changes: first, what one might call the leveling of learning, as a phenomenon that is a ubiquitous and routine part of everyday life, rather than something that happens just in schools, ironically a hard-won accomplishment in the learning sciences, which computer scientists use to frame their own interventions. Second, the popularity of the nudge model of governance lets computer scientists recruit existing experts on learning into their own work as domain experts, rather than framing their work in opposition to these experts, thereby enabling the formation and stabilization of new actor-networks.

This paper draws on two years of ethnographic fieldwork with learning researchers working with MOOCs, conducted as part of a larger study of the edX platform. I spent 5-8 months each with three groups working on learning research, one at MIT and two at Stanford, attending weekly group meetings and interviewing most group members. Of these groups, only one—Stanford's Lytics Lab—involved both computer scientists working with incumbent researchers on learning. In addition, I attended eight conferences and a summer school where these researchers met to present and discuss findings. At conferences, seminars, and talks, I paid particular attention to disagreements between these actors that brought to the fore fundamental differences in their outlooks and methods. I also analyzed the content of this community's substantial textual output, both published papers, and circulating documents within groups.

First, I describe the role that the category of "software" has within the imaginary of computer scientists and how they deploy it in learning research. I argue that the importance of software as a construct flows from the practices of computer scientists that privilege practical system-building and iterative design. Next, I describe how, in their design practices, computer scientists turn both instructors and learners into *users*—another salient cultural category used by computer scientists. I compare this to

the practices of existing educational researchers—particularly those known as "learning scientists"— who see instructors as collaborators in the design of learning environments. Finally, in the section on "The Politics of Expertise," I describe how computer scientists negotiate with these learning scientists, turning them into domain experts to be deferred to, while still critiquing them. This negotiation happens, at conferences, in front of funding agencies, and often within dissertation committees. I end by showing how MOOCs and the intervention of computer scientists lead to significant epistemological questions within the study of learning.

Software as a Medium of Governance

The historian Ellen Lagemann (2002) argues that from its inception in the early 20th century to today, American education research has been shot through with two key tensions: first, about the *methods* through which the research is to be conducted, and second, about how this research is to be *translated* into educational practice (see also chapter 4 of Labaree 2006). Both questions go to the heart of the enterprise. Is educational research to be conducted in laboratories or schools? With qualitative or quantitative methods? By enlisting teachers or leaving them out? These tensions have been re-configured and re-interpreted by new actors who seek to intervene in education research. For instance, Edward Thorndike and other behaviorist psychologists offered very definite answers to these questions in the early 20th century: education was to be studied using measurements, in laboratories, and by psychologists alone without any teacher involvement. This solution was widely accepted in the light of the emerging professionalization and specialization of research within the academy, the growing prestige of psychology as a discipline, and the low status of largely female teachers. Other battle took place during the Cold War over physics (Rudolph 2002) and mathematics (Phillips 2014) curricula, pitting newly empowered disciplinary experts against educationalists from Education schools.

Figure 1: A computer scientist's answer to the question of how the field can contribute to the study of learning and the design of educational technologies.

Not surprisingly, in the light of their own high-status as high-flying innovators, and the architects of software infrastructures used by large populations within the US, computer scientists have their own responses to these tensions, which they use to get their foot into the door of learning research. Their answer to both questions is: *software*. On the first question, of how educational research is best done, they stress building practical software systems that *work*. They argue that it is only by building systems and trying them out to see if and how they work, and then rebuilding them to fix errors, that (learning) research can truly be done. One of them characterized it as the "build a little, get feedback, correct, build a little more" approach.[4] On the second question of how educational research should inform practice, computer scientists argue that if research results can simply be inscribed into software, then this will structure the practices of learners and instructors; this, they suggest, is simpler than traditional institutional processes. The high-profile computer scientist Scott Klemmer referred to this as "baking pedagogy into software" (see Figure 1). Another computer scientist referred to it as "packaging"—a unit of software and a term intimately familiar to software engineers. As he put it:

> *I think the reaching across the aisle challenge [i.e. integrating learning research into practice], the artifact version of it, is how many of the best practices, from the education literature, that are successfully adaptable in this context, or maybe can even be improved in this context, can be*

[4] Email between the group members of the Lytics Lab, Stanford dated Feb 23, 2015.

packaged into software so that people who do not necessarily know that research still benefit from them.[5]

The answer to the first tension—build software—arises out of practical considerations and the existing practices of computer scientists. Computer scientists, except those who work in highly mathematical areas like the theory of algorithms, are expected to build practical working systems. Speaking about the world of Artificial Intelligence (AI) at MIT that he was an integral part of in the late 1980s, Philip Agre (1997) writes: "Building things was truly the end purpose of the hacker's work, and everything about the methods and language and value system of the AI world was organized around the design and implementation of working systems. This is sometimes called the 'work ethic': it has to work." Computer scientists may have frameworks, ideas, approaches or theories but none of these matter unless they are also embodied in working systems.

The answer to the second tension—"baking pedagogy into software"—is explicitly ideological in that it specifies a preferred channel of *governance*. The assertion can be traced to the use of "nudging" techniques (Thaler and Sunstein 2009) extensively deployed in the world of Internet platforms like Facebook, Twitter, Amazon, and YouTube. These platforms have heralded a transformation of the World Wide Web from a relatively open, non-commercial space, to one that is dominated by a few behemoths, obligatory passage points that curate the Internet experiences of many Web users. Scholars have argued that the one factor that contributes to the reach of these platforms is their reliance on "public relevance" algorithms (Gillespie 2014) and "default" features (van Dijck 2013) to shape the activities of their users. These algorithms (e.g. those behind Facebook's Newsfeed or Amazon's recommendations) are rhetorically and technically constructed to keep users engaged and to nudge them towards more interactions that are desired by designers. Computer scientists studying learning with MOOCs are intimately familiar with the nudging techniques of social psychology and behavioral economics used in platforms like Facebook, and popularized by policy-advisers like Sunstein and journalists like Charles Duhigg (2014). A survey of

[5] Armando Fox of UC-Berkeley, speaking at the conclusion of the Learning with MOOCs conference, Cambridge, MA, 2014.

MOOC research published in Science points out that "experiments [such as incentivizing learners to participate in discussion forums with the promise of a digital badge to improve overall learning outcomes], often inspired by psychology or behavioral economics, are widely under way in the field" (Reich 2015).

There is a second reason why "baking pedagogy into software" is an ideological assertion. It allows computer scientists to intervene in a domain—the study of learning and building learning infrastructures—without specifying the nature of their own expertise and deferring to established experts whom they configure as "domain experts." The assertion implies that computer scientists are not experts on pedagogy, yet they can take the lead on building systems while remaining free to pick and choose pedagogical insights to bake into the software, all the while remaining neutral system-builders. I explore the implications of this in the penultimate section on the politics of expertise.

Were computer scientists always this instrumental and deferential towards other experts? The history of computer science, and its most relevant subfield of Artificial Intelligence (AI), shows otherwise: the field's stance on other kinds of expertise in the beginning was aggressively adversarial but has shifted to a more deferential view in the light of a key shift in the field's conception of intelligence. Computer science arose from what is called the post-War "cognitive revolution": a shift wherein the digital computer and the concept of "information" became central organizing metaphors for a whole range of human sciences like psychology, anthropology, philosophy, and linguistics, while also producing new fields like computer science and cybernetics (Gardner 1987). Early AI practitioners were hybrid psychologists and computer scientists—or polymaths like Herbert Simon—and fearless in their forays into other disciplines; the name AI itself is an indication of their intellectual ambitions. Philip Agre again on the MIT AI lab: "I have often heard AI people portray philosophy, for example, as a failed project, and describe the social sciences as intellectually sterile. In each case their diagnosis is the same: lacking the precise and expressive methods of AI, these fields are inherently imprecise, woolly, and vague" (Agre 1997).

The field of AI had a certain approach to building systems through the 1960s to the 1980s, focusing on building *autonomous* intelligent agents that played chess, understood simple natural language, or proved mathematical theorems. Dubbed by the philosopher John Haugeland (1985) as Good Old Fashioned AI and sociologist Mikel Olazaran (1996) as "symbolic AI", AI practitioners understood intelligent human activity as the execution of plans or rules. In this approach, intelligence was seen to reside in the combination of algorithms, knowledge representation and the rules of symbolic logic, that were all written into the software code of the autonomous agent by AI practitioners, sometimes by consulting "domain experts" (Forsythe 2002) While the success of these early autonomous agents was clearly limited, they functioned, at least rhetorically, as replacements to existing experts rather than aids for them. This style of AI most produced brittle systems that did not work beyond toy microworlds and the field went into a crisis in the 1980s.

The style of AI that emerged through the 1990s and beyond was different in nature. Rather than autonomous agents, the field now focused on building intelligent systems that consisted of both humans and programs working in concert, with humans often supervising these programs. Today, a technique called "machine learning" is extensively used; a machine learning algorithm can be "trained"—using "training data"—to perform specific kinds of classificatory tasks; e.g. recognizing tigers in images. A sufficient number of such algorithms, performing really particular tasks, can be connected together, supervised by humans, to produce intelligent outputs. As machine learning has become easier and easier to use, computer scientists no longer aim to produce autonomous intelligent programs; rather they aim to create assemblages of humans and programs acting together. The culmination of this trend is crowdsourcing which is both a source of data for machine learning algorithms as well as part of a complex machine learning architecture to accomplish tasks using both humans and machines (Irani 2012).

It is against this background of human-machine collaboration, the dream of autonomy largely forgotten (except perhaps in robotics), that computer scientists approach new domains like learning. They concentrate on building software iteratively, but the software itself functions as a kind of modulating medium; the software is supervised by humans but it also coordinates the actions of these humans who

work with and through it. Ekbia and Nardi (2014) have described this social organization as "heteromation," a regime of governance where humans, construed as both consumers and workers, are "fashioned as computational components" for tasks that are critical to computer programs. In this arrangement, software designers need to negotiate with existing actors within a domain—instructors, learning scientists, and other educational researchers in the case of learning—while also configuring these same actors as users of the intelligent (though not autonomous) programs that they build. It is to these developments that I now turn to.

Configuring and Empowering Users

When MOOCs began, they were often engineering classes where assessments could be graded automatically through the software. However, this approach—using multiple-choice questions—would not work for more open-ended assessments in the humanities, social and design sciences. In 2012, then Stanford professor Scott Klemmer and his graduate student Chinmay Kulkarni built in collaboration with Coursera a peer review system (later called PeerStudio) that imagined Coursera learners as crowds. Every learner was asked to assess three to five of his peers' work; a grade for each assignment was then calculated as a weighted average of the peer assessments that it received. The arrangement was based on reciprocity; learners received a grade only if they had assessed a certain number of peers. Kulkarni and Klemmer were able to prove that peer grades corresponded roughly with instructor grades (Kulkarni et al. 2013). Coursera could now safely plow ahead with its plan to host a variety of classes of various stripes; peer assessment allowed Coursera to expand its course portfolio.

From that initial beginning, Kulkarni and Klemmer went on to do a wide range of experiments with PeerStudio. They built a machine learning program to assess and grade short-answer questions (Kulkarni et al. 2014). The catch was that the output of this program would not be used for actual grading; rather, the output of this program—the grade and the certainty with which it was computed—would be used to decide the *number* of peers needed to assess it. The more uncertainty the program expressed in its result, the more the number of peer graders required. Note that it was *instructors* who

were supposed to train this machine learning program; they were its end-users. Kulkarni and Klemmer showed that this arrangement reduced the overall amount of peer assessment required, which was important for MOOCs because it is a time-consuming activity for learners. In a subsequent paper (Kulkarni, Bernstein, and Klemmer 2015), they modified their system so that learners could ask peers to assess *drafts* of their assignments. A similar reciprocal system was designed: a learner was allowed to get feedback on his drafts if only he offered feedback on drafts of other learners. The system yielded two benefits: first, the scale of MOOCs meant that most learners got some form of feedback on their drafts within a few hours. Second, and consequently, these learners ended up with better learning outcomes, in this case, better grades. It should be clear from this description that PeerStudio is a software infrastructure that embodies the principle of heteromation. Programs are not set up as autonomous entities to replace humans; rather programs do coordination work between humans while humans, as users, are fashioned as inputs to these programs.

PeerStudio offers another insight into how computer scientists intervene in new domains: by deferring to existing experts within the domain and by fashioning some of these experts as the *users* of the software they produce. From education researchers, Kulkarni and Klemmer borrow certain principles: that the quality of the assessment depends on the quality of the rubric, that rapid feedback (on drafts) leads to better learning outcomes, and so on. Learners, of course, are the primary users of PeerStudio, but so are instructors. Instructors are the key to making PeerStudio work because they are the ones responsible for the rubric that guides peer assessment. Making PeerStudio *work* as a learning tool deployed in MOOCs thus requires careful nudging of both instructors and learners. In another paper, the designers of PeerStudio (Kotturi et al. 2015) describe a series of heuristics to overcome what they call "adoption challenges for global-scale peer learning." In plain terms, they are setting forth a series of rules ("sociotechnical remedies") on how best to get both instructors and learners use PeerStudio for maximal impact. First, instructors should be encouraged to make activities like peer assessment part of the course itself, perhaps by assigning credit. Second, instructors should make learners aware of their responsibility towards their fellow-learners; designers should therefore build into their software tools for automated, yet

instructor-supervised, tools for emailing learners e.g. the software alerts learners through an email when they leave a peer assessment incomplete. In the most interesting example, they suggest "humanizing" these software-based emails for e.g. by having them include names (of instructors, fellow-learners in the group discussion). They conclude with the suggestion that software developers of peer learning systems must try to "teach the teachers" by providing them with successful examples of software usage by other teachers. In other words, both learners and instructors are users of PeerStudio whose actions have to be carefully regulated through the design of the system itself.

The concept of the user has a long history in computer systems design and in STS (Oudshoorn and Pinch 2005; Kline 2000). In a seminal early analysis, Steve Woolgar (1991) argued from his ethnographic study of microcomputer design that the process of (hardware) design is the process of "configuring" the user by determining the right and wrong channels of interaction; the machine is literally a text that the user interprets, and the designer gets to decide what the right and wrong interpretations of that text are. In Woolgar's analysis, the boundary between the designer and the user corresponds to the boundary of the organization that produces the hardware, which in turn corresponds to the boundary between the inside and outside of the machine; users have no access to the inside of the machine while designers do.

Woolgar's definition of the user as someone who has limited access to the inside of the technology than designers is still mostly right, but it has been complicated somewhat by other analysts, especially when it came to the design of software. Hardware often remains closed off to users, but software can be rewritten and customized for particular sets of users; in fact, most of the jobs involved with software rarely involve writing code as much as they involve customizing and gathering (user) requirements. Software users are also not always outside the organization. In the case of enterprise software, they are often clients, and have a great deal of power to shape design. These clients, or their representatives, often sit in on design meetings and can even overrule designers over particular features. In addition, special kinds of users exist whose job is to bridge the gap between the context of production and the context of use; these special users write documentation, customize the software, and provide

services that make the software "work." The hard line between the context of production and the context of use that Woolgar observed in the design of hardware is considerably softer when it comes to software (Mackay et al. 2000). In fact, as the difficulties of creating large-scale software programs mounted over the years (Ensmenger 2010; Haigh 2001), the user came to have a central place in the methodologies of software design in the US, primarily out of market considerations.

Mike Hales (1994) has described three kinds of users based on different styles of design activity: users as clients who are meant to be satisfied, users as actor-constructors who are meant to be empowered, and users as co-designers who are meant to reflect on the artifact. In the imaginary of the computer scientists, both instructors and learners fall squarely into the second category; these are actors meant to be empowered through the software. "Teaching the teachers" is an important part of this philosophy. They play a role in the design process but always as users, never as co-designers.

The novelty of configuring instructors—if not learners—as users of learning software will become clear if we compare computer scientists to learning scientists, their closest collaborator-competitors in the study of learning and ask how they viewed instructors. The community that calls itself the learning sciences[6] dates from the early 1980s and was formally established in 1991. This community designs what it calls "learning environments." The term is mainly used for particular kinds of educational software that they design, but it is also used broadly to refer to classrooms and workplaces where learning is also seen to occur. The learning sciences are an interdisciplinary community with the constituent disciplines being "cognitive science, educational psychology, computer science, anthropology, sociology, information sciences, neurosciences, education, design studies, instructional design and other fields" (Sawyer 2006, xi). The goal is to understand the "cognitive and social processes that result in the most effective learning" (2006, xi) and then incorporate these into learning environments. In practice, educational psychologists and cognitive scientists are the dominant group both in terms of numbers and in

[6] The question of whether "learning sciences" should be followed by an "is" or an "are" was a topic of a small discussion at a meeting of the Stanford Lytics Lab.

their success[7]; between 2003 and 2006, the learning sciences secured more than 100 million dollars in NSF funding.

The learning sciences pride themselves on their "learner-centered design" principles; by this, they "refer to environments that pay careful attention to the knowledge, skills, attitudes, and beliefs that learners bring to the educational setting" (Bransford et al. 2000, 133). Learner-centered is used primarily in opposition to "knowledge-centered" and "assessment-centered" models of education and learning. Practitioners of the learning sciences argue that older models of the study of learning, led by behaviorist psychologists and disciplinary experts, had put too much emphasis on the knowledge content and on testing. They successfully urged a shift to a more student-centered perspective. The learning sciences also popularized the term "learner" rather than student, a term that was adopted partially in response to studies originating at the Institute for Research on Learning that forcefully made the point that learning did not just happen in schools (Lave and Wenger 1991; Wenger 1999; Brown, Collins, and Duguid 1989).

Learning scientists, since the field's inception, have offered their own answers to the question of how educational research must be conducted. On the question of method, while they focus on building learning environments, they primarily see themselves as scientists, rather than technologists. They aver that the best way to do educational research is to use (cognitive) learning theory as a driver for formulating research questions. As a learning scientist put it in an introductory talk that I attended: "much of student learning is driven by a small set of basic mechanisms." Therefore "an effective instructional resource creates the conditions for these learning mechanisms to support the learning goals instructors have set for students."

On the question of how educational research relates to the practice of teaching, learning scientists treat instructors as collaborators or content-experts. They assert that the development of learning environments requires three kinds of experts: (a) Subject-matter experts who know the domain (e.g. high-school algebra). This role could be filled by a curriculum expert and/or a teacher. (b) Learning experts

[7] Learning scientists told me that their discipline has three approaches to learning: behaviorist, cognitive and situated. One joked that if representatives of each group were asked to come and argue with each other, a "blood bath" would ensue. But she said she herself saw the three approaches as complementary.

who understand the process of "learning" (i.e. learning scientists themselves) which is seen as a cognitive process that is independent of the particular subject-matter (physics or mathematics). (c) Technology experts who develop and build the system. Note that the same person might occupy multiple roles in an actual project, but this is the often explicitly stated model of the division of research labor (e.g. Corbett and Koedinger 1998). Not every project requires all three experts but the best and most innovative projects require all three. All three expertises together—content, learning and technology—are more than just a sum of their individual parts.

Three points must be noted about this approach to building learning environments. First, that the role of instructors in the design process is as collaborators and subject-matter experts rather than users. Second, that there is a clear delineation of the kinds of expertise required, with content and learning expertise being privileged over technology design. Third, the process is long and costly, both in terms of time and resources put in; what is more, learning scientists pride themselves on this aspect. Learning scientists working on a particular class of learning environments called "intelligent tutoring systems" report that it takes "200 hours" of labor "to deliver a full hour of instruction" (Corbett and Koedinger 1998). Others have described the process as a "*huge* investment of time," (italics in the original) especially for the "faculty content expert" (Walsh 2010).

It is in this context that the work practices of computer scientists building learning environments start to come into high relief. By modeling both instructors and learners as users of the software they build, computer scientists are able to reconcile their own commitments both to user-centered design and rapid design processes. They are also able to enter a new domain—the design of learning environments—by modeling it to other domains that they are already involved in. A final problem still remains though. As we saw, learning scientists have divided the space of expertise needed to design learning environments: learning, subject-matter, and technology. In this trio, technology occupies the lowest ladder, a matter of implementation rather than design. How do computer scientists designing MOOC infrastructures become more than mere implementers? I argue that they do this by deferring to existing experts, configuring them as domain-experts and users, often both, but rarely ever specifying the

nature of their *own* expertise other than saying for e.g. that they "bake pedagogy into software." I focus on this point in the next section.

The Politics of Expertise

In 2013, "inspired by the emergence of Massive Open Online Courses (MOOCs) and the accompanying huge shift in thinking about education," a group of computer scientists came together to create a new conference called Learning at Scale (henceforth L@S). The goal of this new conference was to promote "interdisciplinary research at the intersection of the learning sciences and computer science." The primary impetus for L@S came from computer science's premier professional body, the Association of Computing Machinery (ACM). The first conference was held in March 2014 in Atlanta and deemed successful by the organizers; the conference is now in its third year and going strong. The creation of an ACM-sponsored and affiliated conference is an important event in the world of computer science because conferences occupy a somewhat unique place in the field in a way that they do not in other disciplines. By founding a high-status conference to conduct research on teaching and learning, computer scientists were signaling to each other—and to other educational technology researchers including learning scientists—of their interest in the topic. The computer scientists mentioned in the previous sections—Juho Kim, Scott Klemmer, Chinmay Kulkarni—have been key participants in this conference, serving on the program committee, and presenting highly well-received papers there.[8]

The organizers of L@S saw themselves as faced with three major challenges.[9] First, they wanted the conference to be not just about MOOCs but about something broader, but something that could be legitimately seen as a topic of computer science. Second, they wanted to "plant a flag" and bring together all the various sub-disciplines of computer science who worked on learning infrastructures in one place since these sub-disciplines usually published in separate conferences. Third, they wanted a "high-quality" forum of participants from the both the learning sciences and computer science.

[8] Most of the papers on PeerStudio that I cited before have been published at L@S.

[9] I interviewed three organizers involved in establishing L@S in the spring of 2015. Most of the quotes come from these interviews.

The most important of these challenges was the first. The organizers saw MOOCs as exciting but also perhaps too "trendy" and ephemeral. They solved this with the concept of "scale," hence "Learning at Scale." Engineers use "scale" in a different way from the rest of us: scale means when something is both large and small at the same time and elegantly so (Kelty 2000). The concept is used for everything: from software to organizations to business models. By naming their conference with both "learning" and "scale," computer scientists were claiming at least a slice of the available domain.

Conferences where computer scientists already participate and study something that may be arguably called "learning" do exist. The most widely attended is the ACM's annual Symposium on Computer Science Education (SIGCSE) where many computer scientists get together to discuss the best ways to *teach* computer science. But most computer scientists consider SIGCSE as a second-tier *research* conference arguing that the research presented there is just not rigorous enough; also, SIGCSE is just about teaching and learning computer science. Two other conferences exist: the AIED (Artificial Intelligence in Education) and the ITS (Intelligent Tutoring Systems) conferences. These conferences, organized by the AIED society, alternate with each other every year and date back to the early 1980s. These conferences attract a combination of computer and learning scientists. One organizer told me that the problem with these conferences was that "not many first-rate CS people go to them. (no disrespect to the ones who do)." Another conference called Education Data Mining (EDM) had emerged from the AIED conferences in 2008 reflecting the popularity of machine learning; but the L@S organizers saw this conference as far too occupied with applying certain mathematical models to educational data and not occupied enough with building "end-to-end" systems. Finally, there was work on *informal* learning (e.g. Juho Kim's work with chat-rooms for peer-to-peer help in Photoshop) but this was usually presented in the prestigious conferences on human-computer interaction (HCI) like CHI and CSCW. A final conference called Learning Analytics and Knowledge (LAK) had also recently started in 2012, and its goals were similar to L@S; however, LAK is organized by the e-learning community (which is what online learning was called pre-MOOCs) and work at LAK skews as much towards policy as it does towards research. To summarize, L@S would be mostly concerned with *formal* learning, led by first-rate,

world-class computer scientists, but also a forum where computer scientists and learning scientists could collaborate.

In general, the tone of the L@S conference "Call for Papers" (CFP) has been conciliatory and deferential towards existing experts in the learning sciences. The goal of the conference is always stated as encouraging the building of practical systems (with learning science knowledge encoded into them), and rarely, if ever, is there any mention of theory. The CFP for L@S 2016 mentions "theory" but immediately adds the phrase "at scale" to it: "The conference is at the intersection of computer science and the learning sciences, seeking to improve practice and theories of learning at scale." When I interviewed computer scientists who took an active interest in the conference, submitting papers there, they admitted upfront that they did not consider themselves experts on learning, or learning scientists. They stressed that they wanted to collaborate and bring in learning science findings into their systems. Juho Kim, for instance, told me that he did not see himself as a "[learning] theorist" or an "educational researcher" but his vision and his expertise lay in "building new things that enable new ways to learn." His contribution, as he saw it, was to take classroom theories of learning and see if they held up in the wild (i.e. on the Internet).

Just as they configured instructors into users rather than collaborators, computer scientists turn learning scientists into *domain experts* whose theories they test and implement in particular settings. Domain experts are deferred to when it comes to theory, but not necessarily about questions of what systems to build or which theories to test; that choice is for the system-builders, i.e. computer scientists, to enact. Even as domain experts, learning scientists do not get full authority: for instance, computer scientists argue that it is an open question of whether these theories would "scale." In his keynote at the LAK conference in 2014, Scott Klemmer complained that the education community, like the design community, often falls back too often on the answer "it depends." He argued that education research is really difficult to understand, and especially to separate the wheat from the chaff. Domain experts could help computer scientists, he argued, by contributing, for instance, the ten most important things that educators should know.

Tensions between the two communities often came out at conference question-and-answer sessions, and in private. At the first L@S conference in 2014, there were at least three points when learning scientists (or education researchers) walked up to the microphone to express their frustration that the discussions were not incorporating more existing work in the learning sciences. Lori Breslow, the director of MIT's Teaching and Learning Lab used the most striking anthropological imagery about "tribes" and "cultures" to refer to the two groups:

> *I'm on the educational side of the house […]. I'm working with a lot of computer scientists so over the last year and a half, sometimes I feel like I'm doing cross-cultural communication [laughter from audience], so I wonder if as we begin the next couple of days, if you [referring to the keynote speaker] could give us some advice about how these two different tribes can best work together.*

In private, criticisms were more forthcoming. One computer scientist told me that he thought learning scientists were too scientific, that they wanted things to be too perfect but sometimes it was better to wade in quickly and accomplish something that was quick and dirty. Software could never be perfect but it was important to get in there and build it. Computer scientists also characterized learning scientists as too closed off; too interested only in thinking about technologies that helped them modify their theories a tiny bit, and closed off to other technologies that wouldn't fit into their theories.

The learning scientists, for their part, were equally critical of the experiment-till-you-get-it-right ethos of the computer scientists. As one learning scientist put it sarcastically, referring to the tendency of the computer scientists to be unacquainted with prior work done in the learning sciences: why spend an hour in the library when you can spend twenty hours in the lab? Another told me that the L@S conference was better thought of as "teaching at scale" because as he saw it, papers at this conference from the computer scientists were not really about learning; instead they were about *supporting* learning. He understood learning as a cognitive process, while the papers he referred to were about were about creating interfaces between teachers and learners.

At Stanford's Lytics lab where a group of PhD students in the computer and learning sciences met together regularly, tensions sometimes arose between the two sides, focusing often on the point whether researchers in learning should see themselves as tinkerers or scientists. Typically, it would begin after a talk and demonstration of a more computer science-inspired project. The learning scientists in the room would ask what research question the project was solving. The computer scientists would suggest that the project was fulfilling a need, or a projected need, but that once the system was built, they would actually look for research questions that it could answer. Typically, this led to more discussions on how research questions and hypotheses were to be integrated into system-building.

After one such exchange, when the learning scientists in the room were skeptical of whether the system under discussion was even addressing a research question, a computer scientist sent around an email where he humorously and self-consciously caricatured the two sides and how they think about each other. Titled "On the Politics and Practice of Two Project Design Approaches," he argued that there were two approaches to system design within the Lytics Lab, the "top-down" and "bottom-up" approaches and he had observed a preference for one approach, if not outright disdain for the other, amongst lab members, "simmering in our discussions; a latent variable in our midst." Using the term "social scientist" to refer to the learning scientists, and "technologist" to refer to the computer scientists, he argued that while discussions within the lab had always remained cordial, "wars had been fought over less" and the two sides were in danger sometimes of caricaturing each other. The caricatures, he said, went like this

> *Social scientist [on technologists]: These technologists sit around building stuff they think is cool. Then they parade that around, looking for an application. Their naiveté about what humans want, or can work with, and about what's important is bottomless. Also: they exhibit a glaring inability to speak in terms others can understand.*
>
> *Technologist [on social scientists]: Social scientists just ridicule what we build; I have yet to hear solid proposals for what might work better. They sit around for months, arguing about obscure theories. If they ever do produce an outcome, they exhibit a glaring inability to speak in*

terms others can understand. Their naiveté about what systems can, and cannot do is bottomless. They think all it takes is to hire a few code monkeys who will design and implement software that works, is extensible, and can be maintained. And then they don't even have money to hire those monkeys.

In the caricature, the learning scientists find "cool stuff" system-building pointless unless there is a research question; on the other hand, the computer scientists find the learning scientists too immersed in their abstruse theories and utterly ignorant of the time and effort it took to build and maintain complex systems. He went on to argue that both top-down and bottom-up approaches to system design had their advantages. In particular, as a computer scientist, and with a partiality towards bottom-up approaches, he found that bottom-up approaches that used rapid iterative design were tremendously generative, raising new and new questions that a top-down theory-driven model of inquiry would not be able to attain. In particular, if learning scientists and computer scientists were to collaborate, he argued, the research "must excite both parties. A setup of 'we decide what is important, you implement' is as unfortunate as 'I built this thing; it's done, eat it'."

When pressed, computer scientists do make a more substantive theoretical criticism of the learning sciences. They argue that the model of intelligent tutoring systems—the most successful application of the learning sciences, commercially and practically—relies on the model of expertise as achievable through "deliberate practice" (Ericsson, Krampe, and Tesch-Römer 1993) and therefore works only for certain domains like middle-school mathematics and introductory physics.[10] It cannot begin to capture the complexity of open-ended tasks and creative tasks, jobs they argue are crucial to conceptions of skilled work in the twenty-first century. Here, for instance, is the opening of Chinmay Kulkarni's job

[10] Learning scientists themselves disagree about whether the "deliberate practice" model works for domains other than school-level introductory courses on the sciences and mathematics. Some will argue that it has never been tried. Others say that the approach can only apply to disciplines where experts can specify the nature of their expertise, and this is possible for, say, introductory physics, and not to history or sociology (see debates in Walsh 2010).

talk[11] at the University of Washington where he presents his work on peer assessment as a model of learning that not only extends but goes beyond the models of learning used by learning scientists:

> *One way of looking at my work is to take this view of learning as something that transforms novices into experts. And you've seen traditionally how it's done with deliberate practice. Students revise, they get feedback, they keep improving, and the traditional way of doing it is with the help of a coach. Over the last decade or two, we've seen intelligent tutoring systems which have enabled more students to do deliberate practice in domains like mathematics. What this talk shows is how we can broaden the domains for deliberate practice to more creative work and more open-ended work*[12].

This question of what the best model of collaboration should be between the new experts, the computer scientists, and the incumbents, the learning scientists, depends on another factor: funding. The learning sciences have been tremendously successful in being funded by federal agencies like the NSF. In fact, one of my Stanford interlocutors argued to me once that the establishment of the learning sciences itself as a field was driven by NSF interest in teaching better science and mathematics to middle-school students. At L@S 2014, one of the invited talks was by Janet Kolodner, a founder of the learning sciences and then the Program Manager of NSF's program on "Cyberlearning: Transforming Education," gave a talk on funding opportunities at the NSF. Kolodner was very clear that grant applications would not be funded unless they cited the relevant literature or included the relevant learning science expertise.

[11] Kulkarni is now an Assistant Professor at Carnegie Mellon University in its Human-Computer Interaction Institute (HCII). HCII is the bastion of both CMU"s formidable HCI-oriented computer science and learning science programs. CMU is arguably the center for learning science research.

[12] The talk can be viewed here in its entirety: https://www.youtube.com/watch?v=2lGNzX1Ap7E.

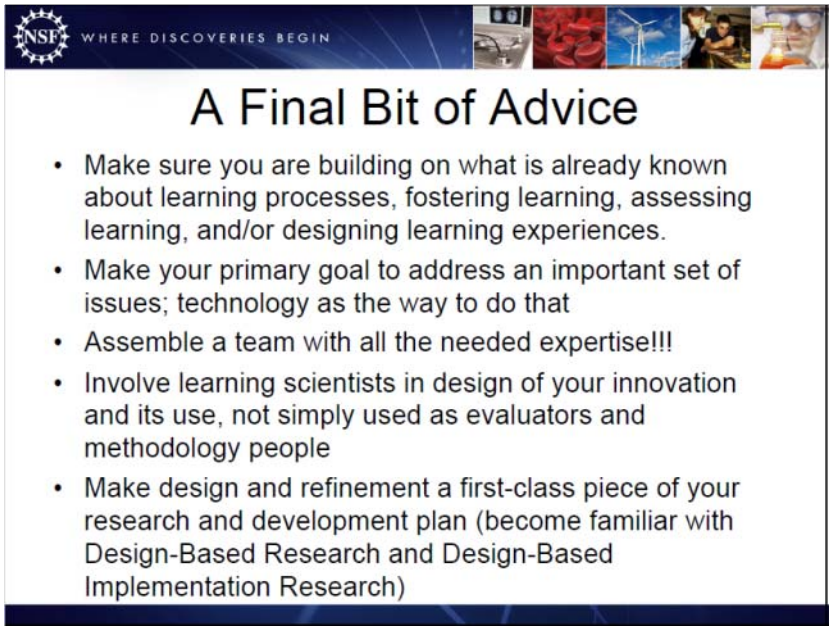

Figure 2: The Program Manager of NSF's CyberLearning Initiative advises computer scientists on the best approaches to securing NSF funding.

Finally, a little bit of advice. Based on some of the things I've heard here. I think that it's really really important for any of these programs to make sure you're building on what is already known about learning processes: fostering learning, assessing learning, or designing for learners; all of these programs that I talked about require that and it's really important not to be technology-centric. Well, if you're dealing with infrastructure, you're technology-centric, I understand that, but otherwise, make sure that your primary goal is to address an important set of issues with the technology as a way to do that. Even a platform, an infrastructure, that's the way to do it. Make sure you assemble a team with all the needed expertise, you don't have to know all those journals I mentioned earlier, if you have people on your team who know those journals.

While computer scientists have taken the lead in building learning infrastructures, they are still negotiating the framing of their involvement with the study of learning. The L@S community has been building on its key asset: the presence of world-class computer scientists from elite institutions. Yet, it has had to negotiate three main issues: (1) how to integrate existing learning science work while still

maintaining a distinct identity, and indeed, taking the lead in the field, (2) how to collaborate with learning scientists, rather than just drawing on the literature, and (3) how to negotiate with funding agencies whose officers are likely to be learning scientists themselves and who are likely to demand such collaboration. The future of the L@S conference and its community of practitioners have much to tell STS about negotiations of expertise, and the rise of computer science as a regime of expert knowledge.

Conclusion: Jurisdictional Negotiations

In January 2015, Harvard educational researcher Justin Reich, co-author of 2 full papers at L@S 2015 but someone who identifies more with the education research community (rather than computer science) wrote an article for Science that took stock of research done around MOOCs. Reich argued that newer studies would have to go beyond mere data analysis to actually find out under what conditions learners learned best; the way to do this was to have courses with sophisticated assessments[13] that drew on education research, and to conduct causal experiments to figure out which teaching materials delivered best course outcomes. Yet, he cautioned that these experiments had to go beyond ""domain-independent" plug-in experiments" by which he meant experiments that could be deployed in any course independent of content, for example, those that dealt with meta-cognitive aspects (such as learner motivation or incentives). While these experiments, he averred, had the advantage that "successful interventions [...] can be adapted to diverse settings," however, "[t]his universality is also a limitation: These studies cannot advance the science of disciplinary learning." To address this, he concludes there must be more experiments that look at the impact of specific learning materials and strategies, yet he worries that these are not happening because they require "more intentional nurturing" whereas "[p]lug-in experiments fit more easily in the siloed structures of academia." Reich's argument mirrors the argument made in this paper; he argues that computer scientists and other MOOC researchers are concentrating on "baking pedagogy into software" but not engaging with disciplinary learning (i.e. with physics or chemistry learning in particular). This paper argues that disciplinary-specific learning research requires

[13] By "assessments," he did not mean standards-based summative testing. In general, among MOOC researchers, whether from the learning sciences or computer science, testing is held in low regard.

collaboration with instructors and curriculum-experts which does not fit into the working practices of most computer scientists. It also requires making technology secondary to learning, turning computer science into just a "technical" discipline, a vision that leading computer scientists do not buy into.

A few days after this paper was published, Reich came to Stanford to give a talk. Reich's talk, hosted by Stanford's newly formed Vice Provost of Online Learning[14] (in charge of Stanford's MOOC program) at the newly redesigned Barnum Digital Learning Hub, was packed with people; the audience included many researchers at Stanford studying teaching and learning, both from computer science and the School of Education. The talk itself was largely a summary of his Science paper; it was time, he said, that MOOC researchers studied not just learner persistence or engagement but *learning itself*.

When the floor was opened to questions, the first question came from Chinmay Kulkarni, designer of PeerStudio, and a computer scientist rather than an educational researcher. He asked what he self-deprecatingly called his "stupid question of the day."

> *I guess this is my stupid question of the day.* ***We are assuming that everybody in these classes wants to learn in ways which are very similar to ways the university student learns.*** *Do we know what these guys want? For instance, when I looked at forums on classes, they're not saying: my achievement levels are up. Or they're not saying: I learnt this great thing about social psychology today. It's: this class told me something today that I saw in real life! And that's really fascinating!* ***Right, so, do we need to change our models of evaluations such that we support what students actually come to courses for, which may be different?*** *[my emphasis]*

The phrase "stupid question of the day" should alert us to the fact that the question poses fundamental issues in the study of learning. What Kulkarni was questioning was what counts as a valid educational goal, and who gets to decide this. Typically, experts on learning have decided what counts as an educational goal and what counts as a good assessment to evaluate those educational goals. MOOCs had changed this picture for many of my interlocutors, changing the definition of what "course" and "learning" meant. MOOC courses have ranged from introductory classes in computer science and

[14] In 2015, VPOL was renamed to Vice Provost of Teaching and Learning (VPTL).

physics to classes meant to raise awareness of poetry and global architecture to more interventionist classes on public health and women's welfare to classes on MongoDb and Microsoft Excel to professional education classes on "data science" to help corporate employees pursue professional advancement in Silicon Valley. These courses, with their blend of formal and informal matters of education, have arguably shifted what a "course" means and thereby what a "learning objective" for a course can be. Reich replied to Kulkarni's question by answering that it was best to leave the definition of a course objective to particular instructors themselves. But this answer itself is an indication of a shift: that instructors, and not say curriculum or learning experts, get to decide what learning objectives are.

In my two years of fieldwork around MOOC system builders, instructors and researchers, I observed these sorts of conversations numerous times: debates about what the objectives of learners are, about who gets to set the learning goals, and how best to measure and understand them, if they are not set by incumbent experts like learning scientists. Rather than seeing these conversations as simply a response to demographic and technological changes (as my interlocutors often did), this paper shows that they are also the result of an ongoing "boundary work" negotiation between expert communities. These questions are truly fundamental: is the learner best considered a user of software or a student of a higher education institution? Are instructors best understood as co-designers or just another class of users of these software systems? Finally, is a learning infrastructure best seen as an example of an institution or a "platform" like YouTube and Amazon?

These questions have become salient in the wake of the rise of computer scientists as experts who design some of our most ubiquitous software infrastructures. Computer science as an academic discipline has always had a porous boundary, and the field itself is full of what might seem to be paradoxes. It is a "science" but most of its practitioners make and maintain new kinds of software; they are technologists. They privilege systems rather than ideas; a system that implements certain ideas is considered better than the ideas themselves. It is ostensibly about "computers" but its founders, in their effort to differentiate themselves from electrical engineers, were very clear that it was about "computation" which was any activity that could be expressed in the form of rules and algorithms, i.e. in formally specifiable terms

(Mahoney 2011). These contradictions however have made computer science an extraordinarily fertile and porous field, capable of expanding or contracting its boundaries to incorporate new domains and activities for analysis.

While academic computer science dealt comfortably in abstractions (while also building systems), programmers in the workplace struggled to establish themselves as a profession. Jurisdictional contests dogged the software profession from its very beginning: in particular the conflict between programmers and mid-level managers over the role of computing in the workplace. Programmers, in their various incarnations as coders, systems analysts, and consultants, tried to assume—using as their bargaining chip, their technical skills and the increasing role of data processing in the day-to-day working of corporations—the jobs that mid-level managers did (Haigh 2001; Ensmenger 2010). Managers succeeded in this fight by virtue of being able to label programmers as mere technicians.

The status of both academic computer science and software engineers has arguably changed with the rise of the Internet and the ascension of Silicon Valley as the nation's greatest site of innovation. Software engineers and computer scientists have been successful at building software infrastructures that are now used ubiquitously, in workplaces and at home. This, in turn, has empowered new actors—"geeks" (Kelty 2008), "hackers" (Coleman 2012), and Silicon Valley entrepreneurs (Streeter 2010; Turner 2006)—to construct technical practices (e.g. open-source sharing, open licenses) and ethical representations (e.g. the theory of "open-source governance") that re-orient existing configurations of power and knowledge (Kelty 2008). These actors have turned their attention and technical expertise to activities usually imagined as strictly "social" by designing software infrastructures for brokering services (e.g. AirBnB, Uber), the distribution of cultural products (e.g. Netflix, Hulu), and new forms of paid labor (e.g. Amazon Mechanical Turk); computer scientists, in addition, seek to become social science knowledge producers and have called in recent years for a new kind of "computational social science" that draws on "big data" (e.g. Lazer et al. 2009; Lohr 2013).

This paper has demonstrated that the question of who builds new infrastructures—and by implication, who has technical and institutional access to them—is crucial to understanding how

computer scientists (and other experts) intervene in new domains. In the first instance, elite computer scientists at Stanford and MIT created MOOC infrastructures: a platform for formal classes that resembled in look, feel and working, the many sites on the Internet dedicated to informal learning. Early Coursera classes drew heavily on Stanford faculty; Scott Klemmer was recruited early on to teach his class on human-computer interaction. And it was to Scott Klemmer (and not say, David Boud, the learning scientist whose book Klemmer read in 2009 that inspired him to try out self-assessment in his Stanford classroom) that Coursera turned to when they became interested in the question of how open-ended assignments would be assessed and graded. Klemmer and his student, Chinmay Kulkarni, were able to conduct early experiments (sometimes through sheer ingenuity) to both validate the efficacy of peer assessment as well as how peer assessment might be improved in a massive online context. Designing rapidly, at start-up pace, drawing on the literature of the learning sciences (and social psychology and management theory), and using their own class as an experiment, and themselves as users, they were able to help Coursera architect a key component that helped sustain its online empire. Juho Kim's story is similar. Interested in transforming how-to videos into robust learning resources, MOOCs were an ideal vehicle for his research. His internship at edX in the summer of 2013 was as much a result of his proximity to edX as it was of his considerable talents; it is inconceivable to me that edX, an organization that is mostly software engineers, could have employed a learning scientist intern. Working at edX allowed Juho to build his LectureScape browser and solidified his claim that lecture videos and how-to videos were all on the same continuum of educational videos. If the integration of LectureScape into the edX platform is still far from happening, that is more a consequence of the fact that the edX platform has matured, and modifying its core features requires far more time, effort, and expense. On the other hand, edX and Coursera employees are regularly found at the Learning at Scale conference, and ideas demonstrated at this conference are far more likely to end up as features in the software of edX and Coursera.

This paper has also demonstrated that even as computer scientists are entering new domains, they remain tied to the idea that they are doing computer science rather than a different discipline, expressing

their findings in terms of abstract concepts like peer interaction, video analytics or even, learning at scale. These abstractions perform a powerful function, linking two different domains together: on the one side, the elements of software architecture and on the other, substantive topics like brokering, cultural distribution, or education. The links to the former are tight; the links to the latter are scruffier. The scruffier links allow computer scientists to intervene in more domains; the tight links help them connect back to software architects who design computing infrastructures (through institutional channels like conferences, internships, and perhaps even consulting). The result might be a leveling of social categories—like "learning," "brokering" etc.—through the fact that these activities are carried out in infrastructures that are designed to look very similar from a system architecture perspective.

The question of abstractions ties into another debate within STS studies of expertise. Eyal (2013) critiques Abbott's (1988) notion that the level of abstraction of professional knowledge is key to whether a profession is able to securely control its jurisdiction; a profession whose knowledge is too abstract fails to anything useful; a profession whose knowledge is too tied to particular domains risks losing its jurisdiction. In response, Eyal argues that the notion of an expert knowledge that needs to be at just the right level of abstraction, neither too abstract nor too concrete, is insightful but it only shifts the question a level down. For an analyst, it is not an easy task for an analyst to figure out the "optimality" of the abstract knowledge deployed by an expert community until after the jurisdictional negotiation is (mostly) complete. Eyal suggests instead that we "replace "abstraction" with the notion of "immutable and combinable mobile" and investigate the chain of transcriptions by which an expert statement or performance is conveyed along the network toward its "centers of calculation"" (2013, 874). This paper demonstrates the validity of this analysis: the abstractions produced by computer scientists are abstract enough to apply to a wide range of infrastructures (for learning, for brokerage), yet concrete enough that the engineers building these infrastructures can translate these abstractions into decisions for system design; there is a revolving door between the computer science abstractions and the centers of calculation where these infrastructures are built that is traversed frequently, through internships, employment, conferences, and consulting. What it also shows is that this link—"the chain of transcriptions"—is by no

means stable and finalized at this moment in time. While computer scientists are able to recruit domain experts into their own work by building domain findings into infrastructures, it is by no means clear why domain experts could not do these themselves. And indeed, in recent years, we see economists explicitly venturing into the computational design of markets. Social psychologists and behavioral economists have had a great deal of success getting policy-makers to pay attention their findings. This paper's analysis suggests that while computer scientists have a head-start in terms of having Silicon Valley's attention and the ability to actually build and code infrastructures, they will face competition from the domain experts they rely on, especially behavioral economists and social psychologists who may develop both: the ability to speak directly to Silicon Valley engineers as well as computational skills to actually build new kinds of infrastructures.

To summarize, this paper has documented the strategies through which computer scientists intervene in a new domain (here, the study of learning). They emphasize building software infrastructures that work, drawing on rapid cycles of iterative software development. They also turn existing actors and incumbent experts within a domain into *users*, who supervise the software, or as *domain experts*, to be deferred to, and whose findings are encoded into the infrastructures that these computer scientists build. Superficially, this resembles one of Abbott's jurisdictional settlements: one profession takes over the task of producing abstractions, the other the task of dealing with clients. But looked at closely, this is not the case. The deference that computer scientists extend towards domain experts does not extend all the way; it is computer scientists themselves who decide what domain knowledge to encode and what particular tools to build. These interventions shift the meanings of expertise within a domain as well as expand the computer scientists' own jurisdiction and increase their social authority, making them experts on multiple jurisdictions and domains. Their modesty about being mere system-builders is thus both true and profoundly transformative for the social domains they intervene in.

References

Abbott, Andrew. 1988. *The System of Professions: An Essay on the Division of Expert Labor*. Chicago, Ill. [u.a.: Univ. of Chicago Press.

Agre, Philip. 1997. "Toward a Critical Technical Practice: Lessons Learned in Trying to Reform AI." In *Bridging the Great Divide: Social Science, Technical Systems, and Cooperative Work*, 131–57. Mahwah, NJ: Erlbaum. http://polaris.gseis.ucla.edu/pagre/critical.html.

Boyer, Dominic. 2005. "The Corporeality of Expertise." *Ethnos* 70 (2): 243–66.

Bransford, John D., Ann L. Brown, Rodney R. Cocking, and others. 2000. *How People Learn*. Washington, DC: National Academy Press. http://csun.edu/~SB4310/How%20People%20Learn.pdf.

Brown, John Seely, Allan Collins, and Paul Duguid. 1989. "Situated Cognition and the Culture of Learning." *Educational Researcher* 18 (1): 32–42.

Carr, E. Summerson. 2010. "Enactments of Expertise." *Annual Review of Anthropology* 39 (1): 17–32. doi:10.1146/annurev.anthro.012809.104948.

Collins, Harry. 2004. "Interactional Expertise as a Third Kind of Knowledge." *Phenomenology and the Cognitive Sciences* 3 (2): 125–43. doi:10.1023/B:PHEN.0000040824.89221.1a.

Collins, Harry, and Robert Evans. 2008. *Rethinking Expertise*. University of Chicago Press. https://books.google.com/books?hl=en&lr=&id=gQpmyyYPc4IC&oi=fnd&pg=PP8&dq=collins+evans+expertise&ots=6GRBel5JlD&sig=492wJxpAv9XtV_2UlxokJTKkDiI.

Corbett, Albert, and Kenneth Koedinger. 1998. "Intelligent Tutoring Systems." In *Handbook of Human-Computer Interaction, Second Edition*, edited by M. G. Helander, T. K. Landauer, and P. V. Prabhu, 2 edition. Amsterdam ; New York: North Holland.

Duhigg, Charles. 2014. *The Power of Habit: Why We Do What We Do in Life and Business*. New York: Random House Trade Paperbacks.

Ekbia, Hamid, and Bonnie Nardi. 2014. "Heteromation and Its (dis)contents: The Invisible Division of Labor between Humans and Machines." *First Monday* 19 (6). http://firstmonday.org/ojs/index.php/fm/article/view/5331.

Ensmenger, Nathan. 2010. *The Computer Boys Take over Computers, Programmers, and the Politics of Technical Expertise*. Cambridge, Mass.: MIT Press. https://anulib.anu.edu.au/tools/generic_revproxy.html?url=http://search.ebscohost.com/login.aspx?direct=true&scope=site&db=nlebk&db=nlabk&AN=324687.

Ericsson, K. Anders, Ralf T. Krampe, and Clemens Tesch-Römer. 1993. "The Role of Deliberate Practice in the Acquisition of Expert Performance." *Psychological Review* 100 (3): 363.

Eyal, Gil. 2013. "For a Sociology of Expertise: The Social Origins of the Autism Epidemic1." *American Journal of Sociology* 118 (4): 863–907.

Eyal, Gil, and Larissa Buchholz. 2010. "From the Sociology of Intellectuals to the Sociology of Interventions." *Annual Review of Sociology* 36: 117–37.

Forsythe, Diana E. 2002. *Studying Those Who Study Us: An Anthropologist in the World of Artificial Intelligence*. 1st ed. Stanford University Press.

Gardner, Howard E. 1987. *The Mind's New Science: A History of the Cognitive Revolution*. Basic Books.

Gieryn, Thomas. 1995. "Boundaries of Science." In *Handbook of Science and Technology Studies*, edited by Sheila Jasanoff, Gerald Markle, James Petersen, and Trevor Pinch. SAGE.

Gillespie, Tarleton. 2014. "The Relevance of Algorithms." In *Media Technologies: Essays on Communication, Materiality, and Society*, edited by Tarleton Gillespie, Pablo J. Boczkowski, and Kirsten A. Foot. MIT Press.

Haigh, Thomas. 2001. "Inventing Information Systems: The Systems Men and the Computer, 1950-1968." *The Business History Review* 75 (1): 15–61. doi:10.2307/3116556.

Hales, Mike. 1994. "Where Are Designers? Styles of Design Practice, Objects of Design and Views of Users in CSCW." In *Design Issues in CSCW*, 151–77. Springer. http://link.springer.com/chapter/10.1007/978-1-4471-2029-2_8.

Haugeland, John. 1985. *Artificial Intelligence: The Very Idea*. Cambridge, MA: The MIT Press.

Irani, Lilly. 2012. "Microworking the Crowd." *Limn* 1 (2). http://limn.it/microworking-the-crowd/.

Kelty, Christopher. 2000. "Scale, Or The Fact Of." In *Proceedings of the Workshop on Traveling Facts*. Wissenschftskolleg, Berlin. http://kelty.org/or/papers/unpublishable/Kelty_Scale_2000.pdf.

Kim, Juho, Ben Malley, Mira Dontcheva, Diana Joseph, Krzysztof Z. Gajos, and Robert Miller. 2012. "Photoshop with Friends: A Synchronous Learning Community for Graphic Design." In *Interactive Demo*.

Kline, Ronald R. 2000. *Consumers in the Country: Technology and Social Change in Rural America*. Revised. The Johns Hopkins University Press.

Kotturi, Yasmine, Chinmay E. Kulkarni, Michael S. Bernstein, and Scott Klemmer. 2015. "Structure and Messaging Techniques for Online Peer Learning Systems That Increase Stickiness." In *Proceedings of the Second (2015) ACM Conference on Learning @ Scale*, 31–38. L@S '15. New York, NY, USA: ACM. doi:10.1145/2724660.2724676.

Kulkarni, Chinmay, Michael Bernstein, and Scott Klemmer. 2015. "PeerStudio: Rapid Peer Feedback Emphasizes Revision and Improves Performance." In *Proceedings of the Second ACM Conference on Learning @ Scale Conference*. L@S '15.

Kulkarni, Chinmay, Richard Socher, Michael Bernstein, and Scott Klemmer. 2014. "Scaling Short-Answer Grading by Combining Peer Assessment with Algorithmic Scoring." In *Proceedings of the First ACM Conference on Learning @ Scale Conference*, 99–108. L@S '14. New York, NY, USA: ACM. doi:10.1145/2556325.2566238.

Kulkarni, Chinmay, Koh Pang Wei, Huy Le, Daniel Chia, Kathryn Papadopoulos, Justin Cheng, Daphne Koller, and Scott Klemmer. 2013. "Peer and Self Assessment in Massive Online Classes." *ACM Trans. Comput.-Hum. Interact.* 20 (6): 33:1–33:31. doi:10.1145/2505057.

Labaree, David F. 2006. *The Trouble with Ed Schools*. 1 edition. New Haven: Yale University Press.

Lagemann, Ellen. 2002. *An Elusive Science: The Troubling History of Education Research*. 1 edition. Chicago: University Of Chicago Press.

Latour, Bruno. 1987. *Science in Action: How to Follow Scientists and Engineers Through Society*. Harvard University Press.

Lave, Jean, and Etienne Wenger. 1991. *Situated Learning: Legitimate Peripheral Participation*. 1st ed. Cambridge University Press.

Mackay, Hugh, Chris Carne, Paul Beynon-Davies, and Doug Tudhope. 2000. "Reconfiguring the User: Using Rapid Application Development." *Social Studies of Science* 30 (5): 737–57.

Ng, Andrew, and Jennifer Widom. 2014. "Origins of the Modern MOOC (xMOOC)." *Andrew Ng*. July 15. http://www.andrewng.org/?portfolio=origins-of-the-modern-mooc-xmooc.

Olazaran, Mikel. 1996. "A Sociological Study of the Official History of the Perceptrons Controversy." *Social Studies of Science* 26 (3): 611–59. doi:10.1177/030631296026003005.

Oudshoorn, Nelly, and Trevor Pinch, eds. 2005. *How Users Matter: The Co-Construction of Users and Technology*. The MIT Press.

Pappano, Laura. 2012. "Massive Open Online Courses Are Multiplying at a Rapid Pace." *The New York Times*, November 2, sec. Education / Education Life. http://www.nytimes.com/2012/11/04/education/edlife/massive-open-online-courses-are-multiplying-at-a-rapid-pace.html.

Pentland, Alex, and Tracy Heibeck. 2010. *Honest Signals: How They Shape Our World*. MIT press. https://books.google.com/books?hl=en&lr=&id=GmUXGwq8O9EC&oi=fnd&pg=PR5&dq=honest+signals&ots=VDbITNFzC9&sig=Oe7mUFesQUK69tY1691cIPwaseo.

Phillips, Christopher J. 2014. *The New Math: A Political History*. Chicago: University Of Chicago Press.

Reich, Justin. 2015. "Rebooting MOOC Research." *Science* 347 (6217): 34–35. doi:10.1126/science.1261627.

Rudolph, John L. 2002. *Scientists in the Classroom: The Cold War Reconstruction of American Science Education*. New York: Palgrave Macmillan.

Sawyer, Keith. 2006. "Introduction: The New Science of Learning." In *The Cambridge Handbook of the Learning Sciences*, edited by Keith Sawyer, 1 edition, 1–19. Cambridge ; New York: Cambridge University Press.

Taylor, Charles. 2003. *Modern Social Imaginaries*. Duke University Press Books.

Thaler, Richard H., and Cass R. Sunstein. 2009. *Nudge: Improving Decisions About Health, Wealth, and Happiness*. Revised & Expanded edition. New York: Penguin Books.

van Dijck, Jose. 2013. *The Culture of Connectivity: A Critical History of Social Media*. Oxford University Press, USA.

Walsh, Taylor. 2010. *Unlocking the Gates: How and Why Leading Universities Are Opening Up Access to Their Courses*. Princeton University Press.

Wenger, Etienne. 1999. *Communities of Practice: Learning, Meaning, and Identity*. 1st ed. Cambridge University Press.

Woolgar, Steve. 1991. "Configuring the User: The Case of Usability Trials." In *A Sociology of Monsters: Essays on Power, Technology and Domination*, edited by John Law. Routledge.