Users as Co-Designers of Software-Based Media: 
The Co-Construction of Internet Relay Chat

Guillaume Latzko-Toth
Université Laval

ABSTRACT While it has become commonplace to present users as co-creators or “produsers” of digital media, their participation is generally considered in terms of content production. The case of Internet Relay Chat (IRC) shows that users can be fully involved in the design process, a co-construction in the sense of Science and Technology Studies (STS): a collective, simultaneous, and mutual construction of actors and artifacts. A case study of the early development of two IRC networks sheds light on that process and shows that “ordinary users” managed to invite themselves as co-designers of the socio-technical device. The article concludes by suggesting that IRC openness to user agency is not an intrinsic property of software-based media and has more to do with its architecture and governance structure.

KEYWORDS Digital media; Communication technology; Co-construction; Design process; Ordinary user

Introduction

User influence on the contents of digital media has been widely acknowledged (Bruns, 2008; Jenkins, 2006; Millerand, Proulx, & Rueff, 2010; Schäfer, 2011). These contributions, in their various guises—from blog and wiki entries to tags, tweets, pictures, sounds, and videos—constitute a key activity associated with the use of social media platforms (Kaplan & Haenlein, 2010). Many media theorists equate the prolif-
eration of user-generated content with user empowerment (Pierson, Mante-Meijer, & Loos, 2011), and they interpret this phenomenon as challenging long-held assumptions regarding the asymmetry between users and designers of information and communication technologies (ICTs). Focusing on content, however, risks taking the technological infrastructure for granted. How interactive media is structured and configured greatly affects its social affordances, enabling and promoting certain usage patterns while hindering others.

Since the inception of Facebook, the relationship between this social network site and its users has been marked by ongoing contestations about design choices. Users often turn to the platform itself to organize and make their voices heard. For instance, in 2008 I joined a Facebook group called “Pour un Facebook en français.” This cause was one of many similar initiatives launched by non-English speakers demanding a multilingual Facebook interface. Our voices, and those of others, were heard. In March 2008, a French-language version of Facebook’s interface was released along with German and Spanish versions. Not only did the people behind Facebook listen to its users, they put them to work. The multilingual interface was the product of contributions from an army of volunteer translators. Since then, other controversies involving default privacy settings, profile categories (e.g., gender, conjugal status), the joining of groups, the exporting of personal data, automatic facial recognition, and requests for a “Dislike” button have manifest themselves in the Facebook environment.

Recurring disputes between social media platforms and discontented users can be understood as negotiations about the shape and uses of new media platforms. Users engage in these protests knowing that their complaints may be taken into consideration, potentially fostering incremental changes in the technical design. Although such mobilizations may be seen as constituting a form of user participation in the technology design process, if one focuses on technical agency the influence of such contributions appears to be more limited in scope, with the underlying information architecture and supported modalities of social interaction continuing to be determined by those who own and manage the platform. Seen through this lens, users’ ability to shape social media is channeled toward the content and configuration layers of networking platforms (see Kilker, 2003), displacing rather than redressing user/designer asymmetry.

The early history of Internet Relay Chat (IRC), a precursor to contemporary social media platforms, demonstrates how users can actively contribute to the design of communication platforms. IRC is an online chat protocol created in Finland in the late 1980s that allowed millions of Internet users to have real-time, polyphonic, written conversations through a decentralized, always-up platform long before the arrival of ICQ, MSN Messenger, Google Talk, and Twitter (Latzko-Toth, 2010). IRC started as a modest program with limited features, evolving into a large, complex technical infrastructure consisting of myriad independent networks of servers. A handful of these networks (e.g., EFnet, IRCnet, Undernet, DALnet, freenode) attract the majority of users. Today,IRC is routinely used by software development communities to support real-time online collaboration.

Drawing on a case study of the early development of the first two major IRC networks—EFnet and Undernet—this article examines the co-construction of actors and
artifacts constituting the socio-technical device known as IRC. It documents how ordinary users succeeded in establishing themselves as co-designers of IRC despite the denial of power and legitimacy they faced from the co-opted group of IRC operators. The analysis suggests that openness to user agency is not an intrinsic property of software-based media, resulting instead from architecture choices and governance structures.

From construction to co-construction

The idea of a clear division of roles between powerful emitters/designers and passive receivers/users of media has long been taken for granted in media studies, and is epitomized by Everett Rogers’ *diffusion of innovations* theory (1995). In the decade spanning from the mid-1980s to the mid-1990s, research from the sociology of ICT uses and consumption demonstrated the shortcomings of this perspective. Much of this work sought to identify forms of user agency and power. Research emanating from both the French-language tradition of sociology of uses (see Jauréguiberry & Proulx, 2011) and British domestication studies (Haddon, 2011; Silverstone & Hirsch, 1992) focused largely on individual levels, while that emerging from the constructivist sociology of innovation tended to be more group-focused.

A recent wave of research in Science and Technology Studies (STS) and related fields (e.g., sociology of action, distributed cognition, activity theory) has been re-examining categories of actors and taken-for-granted dichotomies in a wide range of domains. A common denominator linking these approaches is the construction metaphor (Sismondo, 2008) derived from Berger and Luckmann’s (1966) notion of the social construction of reality. Listing various forms of social construction in early STS scholarship, Sismondo (1993) notes that the adjective “social” relates foremost to the influence of institutions, social norms, et cetera on scientists and engineers, constituting a somewhat minimalist use of the metaphor. He maintains that “the construction metaphor is adequate if we think of it in terms of large, multi-authored projects ..., where the result comes about because of competition ... as well as co-operation” (p. 530), as in “large social projects, whereby such things as cities, economies, legislation and knowledge are constructed by many people interacting, possibly with differing or conflicting goals” (p. 547).

Expounding on this view, Peter Taylor (1995) argues that the value of the construction metaphor does not lie in the quasi-tautological observation that artifacts—whether material or symbolic—are constructed, but rather in its emphasis on the process of “agents building by combining a diversity of components” (p. 356; Italics in original). He labels co-construction an implicitly collective process of joint construction involving heterogeneous elements and emphasizes the iterative aspect of this process. In his view, there is no straightforward relation between the initial state and the built artifact. It follows, therefore, that one must closely monitor the construction process in order to understand the success or failure of an outcome. Taylor avers that this view of construction as an iterative process “shifts perspective not just from separate things to jointly constructed sets of things, but from thinking mostly about the constructed state of the outcomes to examining the processes of their co-construction” (p. 352).

Sismondo’s and Taylor’s claims correspond to the canonical model of the social construction of technology (SCOT) in which the social construction of artifacts is un-
derstood as a process of cooperation and negotiation involving diverse actors. This conception of socio-technical innovation has been widely critiqued for reducing the genesis of artifacts to the outcome of a merely social activity and thus substituting technological determinism with social determinism. It also is accused of offering a naïve vision of society as a static frame within which artifacts emerge. Proponents of the SCOT approach have acknowledged these criticisms and adjusted the model accordingly, proposing a refined SCOT approach, based on the notion of co-construction (Oudshoorn & Pinch, 2003).

SCOT “2.0” research suggests that the work of innovators involves simultaneously building artifacts and their environments, including users. This work has benefited from the interactionist perspective advanced by feminist technology studies, which contends that the homogeneous categorization of actors must give way to more fine-grained and flexible taxonomies capable of accounting for the multiple ways in which one can play the role of a user or a designer. Saetnan (2000), for instance, distinguishes between different categories of users, notably intermediate users and lay end users. The latter are the least empowered, the least able or authorized to have a say in the development of the technology in question, and the most configured by the socio-technical script embedded in the artifact. She highlights the existence of a gradient of human agency akin to that for non-humans. This establishes user and designer categories as dynamic roles, or identities, that can be performed in several ways. As she puts it, “[U]sers can have multiple identities. In addition to being users, they can perform activities and identities traditionally ascribed to designers” (p. 17). This view is echoed by Oudshoorn and Pinch’s (2003) assertion that “users and technology are seen as two sides of the same problem—as co-constructed” (p. 3).

Differences exist within the constructivist perspective regarding how the agent of the co-construction process is defined. For some observers, co-construction entails a discursive and aesthetic operation in which innovators establish representations of anticipated or targeted users of their products or services. An alternative articulation of co-construction assigns both users and designers the role of constructing the artifact, its uses and users, and their representations. These various constructions may co-exist or appear at different moments of the history of an artifact (Lindsay, 2003). The key idea is that actors are constructed—and construct themselves—in tandem with artifacts. This notion parallels the claim from theories of co-evolution, that through constant interaction users and technological artifacts—particularly computer-related artifacts—adapt to each other (Bardini, 2000; Boczkowski, 1999; Orlikowski, 1992). It also aligns the concept of co-construction with the idea of a reciprocal construction of the artifact and its human environment within which the roles of user and designer may be played by either distinct or identical actors.

Within the context of computer-mediated communication, the notion of a mutual construction of people and artifacts (Boczkowski, 1999; Eglash, 2000) transcends claims of the “mutual shaping of social groups and technologies” (Bijker, 1995, quoted in Oudshoorn & Pinch, 2003, p. 3), which simply acknowledges the reciprocal influence of technology and society. As Boczkowski (1999) notes, with computer-mediated communication devices, there is “an interplay among technological features and users’
actions in which all the entities at stake have the potential to influence each other at some point during the unfolding of their relationships” (p. 89).

This brings us to a third conception of co-construction, a construction that in Latour’s (1999) terms is not “a mere recombination of a fixed list of already present ingredients” (p. 139) but where humans and non-humans “mutually exchange and enhance their properties” (p. 124). Seen through this lens, co-construction refers to the active participation of human and non-human entities endowed with agency (i.e., actants) in the development of a socio-technical device, and a process through which the identities of actants transform and cross-define themselves.

In summary, the co-construction of a device can be defined in three different ways: 1) a collective or social construction in the sense of cooperation/negotiation between a variety of actors; 2) a simultaneous construction of complementary entities (i.e., artifacts, organizations, users, et cetera); and 3) a mutual construction of actants (i.e., actors and artifacts shaping each other). These differing definitions are not mutually exclusive. Indeed, they are three aspects of the same process.

**Internet Relay Chat and its actants**

Before turning our attention to the case at hand, it is necessary to introduce the terminology of Internet Relay Chat (IRC) and some of its basic concepts. IRC is a multi-server text conferencing program whose original code was written during the summer of 1988 by Jarkko Oikarinen, who at the time was a student at the University of Oulu, Finland. The ways IRC entities interact, including human users, have since been formalized in a set of publicly technical documents.2

To begin using IRC one must install a software client, choose a screen name (nickname), and establish a connection between the client and an IRC server. The client allows one to send messages through, and instructions to, a server that in most cases is part of an IRC network.3 In order to talk with each other, IRC users need not connect their clients to the same server, but the servers through which they connect must be part of the same network. Although users can have private conversations, they most usually engage in public conversations by joining forums called channels. These channels are global in the sense that they exist network-wide. Servers keep a constantly updated copy of the global list of connected users and open channels, and promptly advertise any local-level change to the networked servers. This keeps their databases in sync.

Entering a channel allows one to see a list of participants (identified by their nickname), among which some have the visibly distinct status of channel operator, or chanop. Individuals with this status are granted special privileges by the IRC technical protocol, and they can remove or ban users from the channel they oversee. They can also configure the mode of the channel and promote other users to channel operator or voice status.4 Their status is not permanent and reverts to that of basic user when they leave the channel.

At the top of IRC governance structure are the server administrators (admins). They manage servers and nominate IRC operators (also known as ops or ircops). The status of these individuals is inscribed in the server configuration file. IRC operators help maintain the network in working order, including policing the communica-
tion space and acting against “disruptive users” (hence the pun on “cop” in “ircop”). They have much power at their disposal, notably, the ability to access and use the “kill” command that enables them to disconnect users from the network.5

Within the IRC environment, every human is first and foremost a user who may encapsulate different roles. One may be an IRC operator and yet a simple user on a specific channel, while simultaneously being a “voice” on another channel. That said, not every user is human. The ecology of IRC entities is complicated by the presence of bots and scripts (i.e., non-human actants). A bot is an autonomous program capable of signing on to IRC by itself and interacting with other servers and clients. In addition to bots, there are less elaborated code objects known as scripts. These tend to be client add-ons (macros) written in the integrated programming language provided with some popular clients to allow users to automate repetitive tasks. Many IRC users, even beginners, customize their client with a number of ready-made scripts available on the Web. Like bots, script-augmented clients can react to online events without any direct intervention from the user.

Methodological considerations

For this case study, a three-prong inquiry strategy was adopted based on virtual ethnography protocols (Hine, 2000, 2005). The first prong comprised several years of online observation and participation with IRC providing me with practical knowledge and an insider’s perspective of this communication platform.6 The second involved undertaking a discourse analysis of official websites, plus documentation archived on the Web and in public and private mailing lists used by IRC officials. This helped to produce detailed accounts of debates between actors based on their discourses at the time of the events being analyzed. Third, I conducted online interviews with key actors in order to validate and refine the discourse analysis.

Table 1: IRC forums analyzed

<table>
<thead>
<tr>
<th>Mailing lists</th>
<th>Usenet groups</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Irclist</strong>: First list gathering early IRC actors of all kinds (1989?–1991)</td>
<td><em>altirc</em>: General forum gathering users of all IRC networks (1990–)</td>
</tr>
<tr>
<td><strong>Operlist</strong>: First official list of IRC operators. Eventually become devoted to EFnet (1990–).</td>
<td><em>altirc.undernet</em>: Forum devoted to Undernet matters (1993–)</td>
</tr>
<tr>
<td><strong>Wastelanders</strong>: Former official list of Undernet server administrators. Open to all (1993–1996)</td>
<td></td>
</tr>
<tr>
<td><strong>Undernet-Admins</strong>: Official list of Undernet server administrators. Strictly regulated membership (1996–)</td>
<td></td>
</tr>
</tbody>
</table>

The early IRC development forums were diverse (see Figure 1). Three types of interaction spaces were used to discuss and share views about the evolution of IRC as a whole, specific networks (e.g., Undernet), or a particular component of the platform (e.g., a specific client like ircII). They comprised email forums (mailing lists), Usenet forums (newsgroups), and IRC channels. Given their relative persistence, the first two remain accessible to contemporary researchers. However, very few channel-based IRC
discussions have been preserved. The information gathered to produce and support the analyses presented below was obtained largely from a corpus of archived messages posted to mailing lists and Usenet groups. The main forums examined for this study are set out in Table 1. With the exception of the Wastelanders and Undernet-Admins archives, which are privately held and to which I was granted access by a former list maintainer, archives of the forums listed in the table are publicly accessible online.

Figure 1: Main IRC development forums (with an emphasis on EFnet and Undernet networks): mailing lists (in italic), IRC channels (with the # prefix), and Usenet groups (others)

The simultaneous birth of EFnet and the ordinary user

The first major controversy in IRC development erupted in August 1990, two years after its inception. The open servers crisis, as I call it, resulted in a first partition of the network and in the birth of the Eris-Free Network (EFnet), when the link with the last open server (Eris) was permanently cut.

The person who controls a server also controls who may connect to it as an IRC operator. All that is needed is a username and a password provided by the server administrator. Although the IRC operator status is, in principle, a technical role that should only be used when network maintenance is required, it became a marker of social status setting apart those who have “a connection to the twilight zone,” as the phrase goes in IRC lingo, from ordinary users—also sarcastically referred to as lusers.

IRC operator access is granted locally on a specific server. However, operator status applies globally, such that an operator-only command can be entered by any IRC operator regardless of the server through which she is connected to the network. In the early days of IRC, server administrators could appoint any user—including themselves—as an IRC operator simply by adding a line to the configuration file. To become an IRC operator it sufficed to link one’s server to an open server. The latter are servers
configured in a way that allows any server to connect to them. By contrast, closed servers are configured to only accept connection requests from specific servers. With this simple hack, a self-appointed IRC operator was able to control any server attached to the network. Put simply, open servers let untrusted operators enter the network (see Figure 2).

Concerns about unwanted IRC operators were frequently expressed during the debate about open servers that took place on the Irclist and Operlist mail forums. The two comments below are exemplary of concerns expressed about the potential for abuse created by open servers:

It will no longer be true that Joe User can become an operator and generate spurious kills. (G. L., Operlist, 08/23/1990)

The point is that the *risk* of having unwanted operators should be removed. (S. K., Operlist, 11/21/1990)

Open servers were also seen as a threat because they enabled servers that did not conform to the agreed-upon IRC protocol to join the network (Oikarinen & Reed, 1993). To this end, concerns were frequently voiced about the risk associated with the so-called exotic features of some open servers, including mechanisms allowing for their operators to breach the privacy of channel conversations or, even worse, private conversations. In a message posted to Irclist in the spring of 1990, Jarkko Oikarinen clearly articulated this concern with regard to a server release he did not approve:

[D]oes the new server include things which affect irc privacy? Like operators being able to see secret/private channels? I think I’m going to quit using irc soon if that’s true. (J. O., Irclist, 04/06/1990)
While some server administrators and IRC creators viewed open servers as a threat to the network, others saw them as a means of guaranteeing democratic governance. Evidence of this can be seen in the following comments posted to the Operlist:

[C]ontrol freaks happen. This is part of why open servers are a good thing. To ensure that there is always another game in town, and that a few people *never* have an opportunity to dictate policy to the network. (R. T., quoted by J. G., Operlist, 09/12/1990)

I think that open servers embody a useful “higher power” that helps prevent people who want to “take power over other people” from doing so. (J. S., Operlist, 09/18/1990)

The open server crisis took a decisive turn on August 7, 1990, when Jarkko Oikarinen posted an email message to Operlist calling for a vote on a proposal to prevent two remaining open servers from connecting to the IRC network, and requiring that henceforth only approved closed servers be permitted to connect to the network. This came to be known as the closed server policy. The result of the vote—a strong majority in favour of the policy—was posted a week later, and implemented one month later as a patch to the server code that effectively quarantined server Eris (eris.berkeley.edu).

Up to this time, running one’s own server on the IRC network was considered a legitimate aspiration open to all users. The closed server policy rendered this option a privilege reserved for a few trusted operators who were sponsored by their peers. As such, it raised non-trivial questions about what it meant to use IRC. For some, use was understood as including installing and running a server, while for others—a majority—the definition was more limited in scope:

I think it needs to become more of a common perception that a server is not a prerequisite to running IRC … (M. P., Operlist, 08/07/1990).

People don’t have to be able to run a server in order to use IRC, a fact that’s not made tremendously clear in the documentation. (M. P., Operlist, 09/18/1990).

Another argument favouring the closed server policy was that the users/server ratio—hence the user/operator ratio—was too low:

Let’s cut back on the number of servers we have and INCREASE the number of clients. We should make some sort of policy that to have a server at a site you must have a set number of users at that site that will be using it. (S. M., Operlist, 08/20/1990)

At the same time, some IRC coders recommended that the client program (irc) and the server program (ircd) be released separately, instead of being packaged together as they were at the time:

Suggestion #1: Prepare a client-only distribution package, and encourage its use. (M. B., Irclist, 08/01/1990)

What seems to be at stake in these exchanges is the distinction, within the undifferentiated agglomeration of roles and statuses characterizing early IRC enthusiasts, of a specific category of users, namely, the ordinary user. This category was constructed
negatively by excluding actors from the group of administrators, despite the criticisms of some protagonists:

IRC is a peer-admined network of servers. Nobody has a *right* to be a member of this group simply because they can compile a server. (M. V. L., Operlist, forwarded to Wastelanders, 08/11/1993)

I am worried about this trend of making IRC some kind of “closed group.” *Anyone* should be able to run a server… (M. S., Operlist, 08/20/1990)

Some IRC operators labelled ordinary users “non-operators,” demonstrating how the categories of operator and user cross-define each other. Although the term operator initially referred to a technical role, with time it transformed into an identity marker of social status designating a category of actors who recognize themselves as belonging to the same community of practice by right (Lave & Wenger, 1991). The fact that the mailing list used as the main IRC development forum was named Operlist speaks directly to this semantic evolution.

The open servers controversy illustrates both collective aspects of a co-construction process as evidenced by the group discussions, polls, and the distribution of tasks, and simultaneous construction features that can be observed as boundaries between actors were produced along with new code, new distribution packages, and new regulations.

**Ordinary users participating in the creation of IRC services**

The co-construction of IRC often took the form of a conflictual collaboration regarding the addition of features, or services as they came to be known, to the initial design. Within the IRC context the term service is polysemic. It can mean a feature, a bot, a daemon, or a subset of an organization. Early IRC programmers employed the term *service* to denote an extension to the set of IRC features that was socially acknowledged as being useful and relevant, and providing added value to the otherwise basic chat features of the platform. Asynchronous messaging, file swapping, and games all fell under the rubric of service. Although these software agents were frowned upon when troublesome, within an IRC community composed largely of computer science students, programming these extensions was seen as a playful and creative activity.

*NickServ, the prototype of services*

In the years following the foundation of EFnet, more and more users sought means to protect what we refer to today as digital identifiers. Claims over nicknames and channel names became important issues, sparking controversies about the addition of related services. The first significant debate of this kind centred on a nickname registration service called NickServ.

Developed by three students from University of Technology, Munich, NickServ was a bot that compared the nicknames of IRC users with those in its database and sent out notices to those who had not registered with this service that someone else was already using their nickname. In order to perform this surveillance, the bot had to be connected to a specific, tailored server that was linked to the whole network (see Figure 3). NickServ had a fellow bot called NoteServ that allowed users to leave a message to a registered NickServ user, thereby extending IRC features to include asynchronous messaging.
When a user connected to EFnet with a nickname that was already registered with NickServ, the bot would send the user a private message resembling this modified example posted to Operlist in January 1992:

```
-NickServ- *— — — — — — — — — — — — — — — — — — — — — — — — -*
-NickServ- ! Attention - Nickname "XYZ" is already allocated by
-NickServ- ! *****@****.info.uqam.ca (The Roaming Soul).
-NickServ- ! This may cause some confusion. Please choose another nickname.
-NickServ- ! If you are the real XYZ, but you are logged into a different
-NickServ- ! computer, you should use the ACCESS command to tell NickServ
-NickServ- ! about this. Type /msg NickServ@service.de help ACCESS.
-NickServ- *— — — — — — — — — — — — — — — — — — — — — — — — -*
```

The service was very popular among ordinary users. Indeed, it was their explicit support that helped its creators to justify and defend its existence. Nonetheless, it was controversial among IRC operators, who were divided over the issue of granting NickServ any power, beyond pestering people by sending them multiple messages. As noted by one fervent user, NickServ did more than just monitor nickname usurpation:

> [I]t indicated when a nickname was registered (thus how long ago), the email address of the person in question, the last time the person was on IRC, and gave the closest we had to a statistical breakdown on where people were ircing from, and thus, an idea of at least the proportional use of irc in various schools and countries. (E. A., email forwarded to Operlist, 05/11/1991)

Some operators wanted to use the bot as a basis for identifying and disconnecting nickname usurpers. Others opposed delegating this level of agency to a bot. Paul Verhoeven’s RoboCop movie (1987) was not far from the imagination of IRC administrators commenting on this matter:

> Giving NickServ an enforcement, RoboCop type of role is a bad idea and a step in the wrong direction, in my opinion. (M. P., Irclist, 04/13/1991)

> My point of view is extremely simple: no way an automate should kill, or incite people to do so by sending messages [to] another logged user. ... In case such a feature (/kill) would be implemented, either I’ll modify the server in order to ignore NickServ, either I’ll simply remove my server from IRC and incitate [sic] others to do so. (C. W., Irclist, 04/13/1991)
Eventually, a “nicknames are not owned” policy was established on EFnet. It was premised on the rationale that the scarcity of popular nicknames resulting from the 9-character limit set by the original protocol, combined with the path dependencies this constraint was subject to, precluded enforcing a one-user one-nickname structure. As some commentators stated:

[E]nforcing nicknames is a bad idea. The whole motive behind nickserv is the problem. With millions of potential users, enforcing one-person-per-nickname isn’t practical. (R. T., Irclist, 04/19/1991)

[I]t’s very technically possible to own nicks. It’s not a technical issue—it’s a policy type issue [italics added]. … Also, once again, NickServ is just a bot. Who owns a nick would be more of a judgement call than a bot database type of issue … (D. M., alt.irc, 10/11/1993)

A much-discussed point in the naming controversy was the fact that NickServ gave ordinary users a semblance of control over their nickname, which otherwise was the exclusive privilege of IRC operators. The latter could “kill” a user to take back a nickname or simply use their technical skills and computer resources to keep their nickname:

The best way to gain “ownership” of a nick is to sit on it 24 hours a day, 7 days a week …, leave yourself logged in, write a bot, whatever. Sure, it uses computer resources, but it’s what many of the net-big.dogs do. (D. M., alt.irc, 10/11/1993)

The “nicknames/channel names are not owned” policies that, until April 1994, were the rule on EFnet widened the gap between operators and ordinary users who, for the most part, accessed IRC through dial-up Internet connections and who could not set the permanent connection required to keep control over a nickname and/or a channel (Bechar-Israeli, 1995).

Bots and scripts as users’ code contribution to the device

When IRC users were unhappy with the limitations imposed either by its technical design or its governance, they had two options. They could leave the network and establish a new network with different rules—in de Certeau’s (1984) terms, a strategic approach—or they could stay and leverage the affordances of automation by delegating specific tasks to bots and scripts (i.e., a tactical approach). Whereas the first wave of IRC users managed to secure rights and privileges by virtue of their early adoption of the platform, those who joined the network later had to contend with an existing socio-technical configuration that offered channel creators no means of securing a grip over their channels.

Consequently, in the early days of IRC it was common for patrons of an IRC channel to take shifts as channel operators so as to keep the channel alive and safe from disruptive users. Individuals who were more computer literate and better equipped technologically used bots as proxies for human operations. Bots were used, for instance, to sit on specific channels, to enforce channel rules and policies by monitoring public conversations, and to take actions against those violating channel rules, as well as to give certain individuals operator status on request. In the 1990s, running a bot required
having an account on a Unix machine with a stable Internet connection and a sponsor among server administrators or IRC operators, because bots were generally regarded as *non grata* on EFnet (Quittner, 1995). Hence, their use reflected and accentuated inequalities between users.

As Ishii (2005) puts it, bot and scripts were the way users managed to interfere with the code of IRC by adding their own layer of code to client-server and server-server protocols. Drawing on the case of Wikipedia, Geiger (2014) suggests that the fine-tuning of software-based communication infrastructure by adding ancillary code (e.g., bots) has become a widespread tendency in digital media development. He calls bespoke code the software code that “extends or transforms the operation of software platforms, but runs on top of or alongside existing systems instead of being more directly integrated into and run on software-side codebases” (p. 343). On IRC, bots come in a variety of forms (e.g., chatbots, warbots, gamebots). They appear as workarounds that innovative IRC users have created in response to what they viewed as missing features or services. Nowadays, most bots are used to look after channels and to protect them against unwanted intrusions, but in the first years of IRC existence, they were often used for entertainment purposes. They also, very importantly, offered ordinary users alternative ways to participate in IRC governance, particularly at the channel level.

The case of the Eggdrop bot is exemplary of the role users played in completing IRC design using bespoke code. This particular bot was developed by Robey Pointer, in December 1993, to protect the EFnet channel #gayteen against constant fights for its control and homophobic raids against the channel. With Eggdrop, Pointer established a standard of guard bots. Its name has since become a generic word for this class of bots (Leonard, 1997; Pointer, 1997). It stood apart from the plethora of similar bots for two reasons. First, Pointer published its source code under the GPL licence, allowing IRC users who were not skilled at programming to use it. Second, the possibility of extending its features, thanks to built-in scripting interfaces, made it highly pliable and, thus, easily tailored to the specific needs of individual channels. It was essentially the first generic bot and, consequently, somehow democratic.

**Undernet and the enrolment of users**

The Undernet IRC network was founded in 1992 by a small group of individuals who wanted to run their own servers, at a time when the main IRC network—EFnet—was already saturated with servers. Driven by the enthusiasm and entrepreneurial spirit of its administrators, Undernet sought to become a viable alternative to EFnet, which would focus on the needs of its users. The originality of Undernet lay in its efforts to establish a more user-centric IRC network by enrolling users directly into its governance structure under the auspices of a User Committee:

Some of the users ... have shown interest in taking part in the committees so I agree, we should include our users in our committees. Give them some say, and I think you will find that they will become much more supportive, and involved. ... Personally, I would like to see more users involved in various things on the undernet, i.e., wouldn’t it be nice to narrow the gap between OPERS and USERS ... (D. M., Wastelanders, 10/05/1994)
The question of whether user bots should be welcome on Undernet was one of the first issues to arise on the Undernet operators' mailing list, Wastelanders. This discussion pertained to the issue of channel governance: specifically, whether channel founders should be entitled to maintain control over their channel, either by using a guard bot or by asking an Undernet operator to intervene after a channel takeover. Put simply, could someone claim ownership over a channel? It was agreed that users should be allowed to connect a bot to the network as long as it was not troublesome. Undernet operators also agreed on the necessity of being able to regain control over channels and return them to their rightful managers, but the IRC protocol provided no means of determining who the rightful managers were. Drawing on the NickServ experience, server-side features would be accessed through the Uworld bot that allowed any IRC operator to regain control over a channel. However, when certain IRC operators began to use this process for their own benefit—most IRC operators are channel managers as well—it was clear that a more impartial device was needed.

A user who was also the first User Committee chair suggested the solution to this problem. He proposed establishing of a Channel Service Committee (CSC) bringing together users, IRC operators, and coders to manage a channel register and to oversee an optional process of channel registration. An aspiring channel manager who obtained sufficient votes from other users of the channel would be granted access to an official bot called X. If needed, X would send a request to the Uworld service, sparing Undernet operators the responsibility of directly intervening in cases of a channel takeover. Contrary to Uworld, which acted on channels like a *deus ex machina*, X behaved like a regular channel bot, an approach deemed more friendly to users given their presumed familiarity with bot agency. This was a deliberate decision from CService developers, and it was part of a strategy aimed at users:

> It was also decided that the service should *look like* a bot in order to keep users comfortable with it. Instead of a server op, they see a user op, which makes all the difference. (Robin Thellend, original CService developer, Wastelanders, 07/13/1995)

The channel service concept was implemented in a very different way on other networks, such as DALnet, where it worked on a first-come, first-served basis. The originality of the solution implemented by Undernet was its concern with democratic representation for channel managers. Political representation was performed through human mediation provided by the organizational wing of the service, whereas channel access regulation was delegated to its informational/algorithmic wing. In other words, CService was a hybrid solution combining the material and organizational with human and non-human agency. It also was hybrid insofar as it was the product of the contributions of lay users (through their actions and the voicing of their ideas within the committee) and expert users (through official and bespoke code). In this sense, it shows how IRC services were co-constructed by users and operators.

**Discussion**

The early development of Internet Relay Chat offers a clear illustration of the co-construction process. On one level IRC may be understood as a collective construction in
the sense of being a collective, highly distributed project where different actors, unequally empowered, participated in the design of the device. On another, complementary level, it may also be seen as exemplifying a process of simultaneous construction. Actors were shaped in tandem with the development of IRC. Its developers invented user categories such as operators, ordinary users, disruptive users, et cetera. IRC practices, including communication practices, were also constructed along with the artifacts mediating these exchanges (e.g., practices relating to nicknames and their limitations). More fundamentally, the very notion of use is itself a simultaneous construction. As shown above, social interactions between actors involved in the early development of IRC led to certain activities being excluded from what was considered as constituting using IRC. A second step in the construction of the practices associated with legitimate use was the gradual exclusion of programming practices through the curbing of user bots and some types of scripts. As a result, coding became an elite activity on IRC. This suggests that notions of both use and user are socially negotiated constructs that were constructed jointly with this communication platform.

In IRC development we also observe processes of mutual construction wherein actors and artifacts cross-define and shape one another. The notion of ordinary user emerged along with the closure of the group of IRC operators. Scripts and bots also acted as a defining agent, producing new categories of actors. For instance, IRC coders and advanced users labelled one type of user (typically younger) who runs malicious programs to disrupt other users’ IRC chat experiences, script kiddies. These individuals exploit flaws in the device to disrupt its functioning for fun, to disturb or intimidate other users, or simply to challenge the power of operators. In 2001, script kiddies forced Undernet administrators and coders to transform the core code of channel services and to modify the server-client protocol in order to hide certain information that was used to target servers hosting the services.

On a different note, bots—be they user bots or official services—contributed to the emergence of a more textured form of IRC governance than that first envisioned in the original technical protocol. At the channel level, the person who controls the bot controls the channel, and since most bots feature access levels, they allow for a much more subtle scale of power than the binary structure inscribed in the IRC protocol (operators versus non-operators). New terms were created to designate different levels of channel operator status: auto-op, super-op, channel manager or founder, and so on, depending on the network, language, or specific channel culture.

We also observe in this case artifacts mutually shaping themselves. Given that bots can act, and react, much faster than humans, they shifted the temporality of actions from seconds to milliseconds, exposing shortcomings in the original protocol and forcing IRC developers to make, in some instances, drastic changes to the code. Some of these modifications were the subject of intense controversies, resulting in code forks and the differentiation/fragmentation of IRC networks. In addition to shaping the code of IRC, non-human agency contributed to redefining the experience of online chat, forcing IRC network managers to make their conception and philosophy of the media explicit.
Conclusions
User contributions to the design of digital media continue to be an area of active research. In this article, I adopted a Science and Technology Studies (STS) perspective to illustrate the technical agency of users in the co-construction of Internet Relay Chat. Examining critical moments in the development history of this software-based media platform reveals that the asymmetry in the distribution of technical agency between actors is constructed post hoc. Being a user is a role ascribed by a dominant group of self-proclaimed developers to another group of people who are kept away from certain technical activities and decisions. IRC users acted as co-designers in many ways, depending on their technical knowledge and skills. Some contributed by adding unofficial features or services to the device as a response to what they perceived as flaws in the technical protocols. When allowed to do so, others made their voice heard either by campaigning, expressing their views on the operators’ lists, or engaging as volunteers in the governance bodies of the networks. While this does not mean that all differences between actors were erased, it does show that users can, and often do, play an active role in co-designing and configuring the media through which they interact, provided they get (or create) the needed space of freedom and autonomy.

The case of IRC development also demonstrates that the plasticity of digital media does not guarantee user agency. As was noted by Gillespie (2006), digital devices can be designed so as to be even more resistant to users’ inquiry and tinkering than their physical counterparts. It follows, therefore, that the co-construction of digital media can be hindered or promoted depending on governance structures and how these structures are embodied in architecture. This observation is of critical importance as we enter an era of hegemonic proprietary social media platforms in which users are increasingly being locked-in by the device architecture (Boullier, 2012). Examining the role users play in the genesis and evolution of software-based media is crucial to articulating a critique of the design and governance of emerging media platforms.

Acknowledgments
I would like to thank Logan D.A. Williams, Kevin Fodness, Toluwalogo B. Odumosu, and Denver Xiaofeng Tang for organizing the “Knowledge from the Margins” sessions at the 4S 2011 conference and for giving me precious feedback on a former version of this article. I am also grateful to the anonymous reviewers whose detailed and insightful comments at different stages of the writing process helped me improve the final product.

Notes
1. The freenode IRC network is mostly dedicated to this infrastructural function. Various Linux distributions have been using it from the onset as a real-time complement to their asynchronous forums (e.g., mailing lists). It hosts hundreds of channels used by Wikipedia as a collaborative tool. Cyber-activist organizations, such as Anonymous, have set up their own IRC networks to support their operations (see Dagdelen, 2012).

3. It is possible to use IRC on a single server, for instance to fulfill the conferencing needs of an organization. However, the server code was originally designed to handle no more than a few hundred clients at a time. Hence the need for interconnecting servers to ensure some form of scalability.

4. This intermediate status comes with the privilege of speaking when the channel is in “moderated” mode. Most of the time, it is honorific.

5. Originally, this command was meant to manually terminate ghost connections. Servers routinely send “kill messages” to other servers when multiple occurrences of the same nickname are found. The kill command forces the emission of a kill message.

6. I have been using IRC since 1995, first on EFnet, then on Undernet (starting in 1996), where I helped establish and register a new channel.

7. Each excerpt is followed by its author’s initials, the date the message was sent, and the name of the list or forum where it was collected as well as the online space where it was originally issued if it was different. This way of “blurring” attribution of speech to actors was inspired from Mathieu O’Neil’s accounts of open source software controversies (O’Neil, 2009). It is meant to complicate the re-contextualization of online discourses and the identification of their authors, since the initial conditions of their circulation have been altered in time through archiving, re-publishing on the Web, et cetera.

8. This ironical play on the words user and loser spread in computer science jargon from MIT in the mid-1970s (Raymond, 2004). It is significant of the contempt expressed by a self-proclaimed computer elite toward users. The term has been consecrated in the form of the IRC command lusers, which returns the number of connected users. Raymond notes that luser is a synonym for lamer in hackers’ jargon. One can also note the parallel between the lexical couple hacker/lamer in the hacking scene, and the couple oper/luser in the context of IRC, a “luser” being often considered shorthand for “lame user.”

9. On different occasions when NickServ’s withdrawal was announced, dozens of users protested by sending emails to its developers or by posting messages to Usenet forums like alt.irc. Just like the #wasteland channel, the list was open to any Undernet user, not just admins and operators. Thus even before they were formally integrated into the governance structure, users were de facto taking part in discussions.

10. In July 1996, the TimeStamp vs. Delay controversy led to the “Great Split” between European and U.S. EFnet servers, resulting in the foundation of IRCnet.

References


Lindsay, Christina. (2003). From the shadows: Users as designers, producers, marketers, distributors and technical support. In N. Oudshoorn & T. Pinch (Eds.), How users matter: The co-construction of users and technologies (pp. 29–50). Cambridge, MA: MIT Press.


