Versione PROVVISORIA del contributo presentato al Convegno Annuale

La lotta informatica per la Democrazia cibernetica

Giacomo Tesio
Italia - giacomo(«»)tesio.it

ABSTRACT
L'invenzione di processori general purpose, la rapida miniaturizzazione dell'elettronica e la costruzione di una rete internazionale di scambio dati, hanno avviato una serie di rivoluzioni sociali, culturali e politiche analoghe a quelle prodotte dalle precedenti innovazioni informatiche, ma su scala planetaria ed in rapidissima successione.
Con la propria pervasività, l'informatica sta infatti creando una nuova società cibernetica che lega, integrando sempre più profondamente, complessi sistemi automatici e persone inconsapevoli del loro funzionamento.
La realizzazione, la manutenzione ed il controllo di questi sistemi, automatici ma non autonomi, è affidata ad una nuova categoria di lavoratori, gli informatici, che iniziano lentamente ad acquisire una propria coscienza di classe, esprimendo esigenze, valori e metodi di lotta solo in parte riconducibili a quelli del passato.
Analizzando le caratteristiche di questa nuova lotta di classe "informatica", è possibile evidenziare le dinamiche di potere che operano nella società cibernetica in cui viviamo, dinamiche invisibili per chiunque non sia in grado di comprendere e modificare personalmente il funzionamento degli automatismi che lo circondano.

PAROLE CHIAVE
informatica, informazione, dato, cultura, potere, lavoro, automatismo, algoritmo, programma, bug, sicurezza, complessità

1. INFORMAZIONE E DATO

> *Computer Science is no more about computers than astronomy is about telescopes.*

Queste parole del grande Dijkstra esprimono molto bene il dramma di una disciplina che per ragioni storiche si è sviluppata intorno agli strumenti della propria ricerca, senza aver chiaro il proprio oggetto di studio.

Il termine Informatica deriva dalla contrazione dell'espressione "Informazione Automatica".
Definiamo automatico un qualsiasi meccanismo che, una volta avviato, produce una serie di effetti predeterminati senza richiedere ulteriore intervento dell'uomo per svolgere le operazioni che lo caratterizzano. L'intelligenza umana opera pesantemente nella costruzione dell'automatismo, nella sua messa in funzione e nella gestione degli effetti prodotti, ma non influenza l'esecuzione dell'automatismo stesso (a meno di non voler alterare gli effetti per cui è progettato).
L'automatismo non ha però alcuna autonomia: anche quando cambia il proprio comportamento in funzione dei dati che riceve, tale variazione è predeterminata, in modo consapevole o meno, da chi lo ha realizzato.
I computer sono sempre sistemi automatici, siano essi venduti come smartphone, dispositivi medici o server "nel cloud".

L'informatica tuttavia non si limita alla creazione di automatismi riproducibili elettro-meccanicamente, ma attraverso di essi studia l'informazione, come possa essere trasferita, preservata, rappresentata, interpretata e trasformata, nonché le tecniche che applicano queste conoscenze.

Il termine Informazione, dal latino *in formare,* denota un'idea, un costrutto della mente umana che può essere comunicato in modo preciso ad altri esseri umani. L'informazione esiste esclusivamente nella mente di un essere umano, come esperienza soggettiva e cosciente di pensiero comunicabile, sia essa elaborata autonomamente, derivata da esperienze dirette o appresa da altri esseri umani. L'insieme delle informazioni disponibili ad un essere umano definisce le sue Conoscenze, mentre l'insieme di informazioni condivise dai membri di una comunità definisce la loro Cultura.
L'informazione nasce circa 164.000 anni fa come effetto collaterale dell'evoluzione umana, prodotta dal vantaggio evolutivo fornito dal linguaggio. Ogni informazione nasce per essere comunicata e si distingue da altre componenti dell'esperienza soggettiva umana, come intuizioni o esperienze mistiche, proprio per questa sua comunicabilità. Sono informazioni i numeri, le teorie scientifiche, le riflessioni filosofiche, le analisi storiche o politiche così come le favole, le notizie, le credenze religiose o le osservazioni su un fenomeno fisico.

Fino all'invenzione della pittura, circa 40 mila anni fa, la condivisione delle informazioni avveniva esclusivamente in modo sincrono: due o più persone dovevano incontrarsi fisicamente e condividere l'esperienza che dava origine all'informazione o metterla in comune attraverso un linguaggio sufficientemente espressivo.
Con l'invenzione della pittura e successivamente con la scrittura, l'uomo ha imparato a comunicare in modo asincrono, attraverso rappresentazioni simboliche trasferibili ed interpretabili da altri esseri umani.
Nacque così il dato, dal participio passato di *dare*, ovvero la rappresentazione di un'informazione che può essere appunto

data, trasmessa ad altri, condivisa con altri esseri umani lontani nel tempo o nello spazio.

Il Dato è una delle possibili rappresentazioni di un'informazione, impressa su un supporto fisico ed interpretabile dall'uomo. Sebbene quasi sempre spontaneo ed inconsapevole per la mente umana, il processo di interpretazione del dato non è riducibile ad una relazione matematica: il dato veicola sempre informazioni aggiuntive rispetto al messaggio intenzionalmente espresso dal mittente e la sua decodifica in informazione da parte del destinatario dipende sempre dalle informazioni di cui questo già dispone, dalle sue conoscenze e dalle culture delle comunità di cui è membro.
Grazie a Gödel, sappiamo che è possibile dimostrare matematicamente la coerenza fra dati, ma non la loro veridicità.

Questa relazione organica bidirezionale fra informazione e dato costituisce la chiave per comprendere l'informatica.

Per migliaia di anni la scrittura è stata la forma prevalente di dato, prodotto, elaborato, riprodotto e distribuito manualmente da personale specializzato la cui enorme influenza politica derivava direttamente dai limiti dei loro strumenti, fossero essi geroglifici nell'antico Egitto, logogrammi nella Cina imperiale o le raffinate calligrafie dei monaci amanuensi. Dunque per millenni la creazione, l'elaborazione e la riproduzione del dato è stata prerogativa di uomini al servizio del potere.

Con l'invenzione della scrittura a caratteri mobili di Gutemberg, venne introdotta una prima automazione in questo processo, ma esclusivamente nella fase di riproduzione dei dati: l'elaborazione e la distribuzione dei testi scritti rimaneva prerogativa umana, ma la loro riproduzione semi-automatica abilitò rivoluzioni culturali e politiche impensabili in precedenza, dal Rinascimento all'Illuminismo, dalla Riforma Protestante all'alfabetizzazione di massa.
Le conseguenze di questa piccola innovazione informatica divennero presto chiare alle istituzioni politiche dell'epoca che cercarono di arginarne il potenziale rivoluzionario attraverso liste di libri proibiti, licenze governative che garantivano piccoli monopoli editoriali in cambio del controllo statale sui contenuti pubblicati, fino all'introduzione nel 1710 del concetto di Copyright nello Statuto di Anna.

2. ALGORITMO E PROGRAMMA

Nel 1645, con l'invenzione della prima calcolatrice meccanica da parte del giovane Blaise Pascal, nacque il primo meccanismo automatico in grado di elaborare dati. L'interfaccia utente era estremamente semplice: l'operatore inseriva i dati in input esprimendo i numeri da sommare o sottrarre attraverso alcune rotelle attivate da uno stilo ed interpretava i dati in output restituiti da alcuni dischi numerati. Alla Pascalina seguirono diversi calcolatori meccanici, tutti vincolati ad elaborazioni predeterminate e specifiche, come la calcolatrice di Leibniz o il calendario perpetuo di Giovanni Plana.

La macchina analitica ideata da Charles Babbage e proposta pubblicamente nel 1837 fu la prima a prevedere l'esecuzione di un automatismo variabile, espresso e fornito alla macchina sotto forma di schede perforate: Babbage ideò il Programma ovvero la rappresentazione di un algoritmo eseguibile automaticamente da una macchina. Per diverse ragioni incidentali, la macchina analitica non fu mai realizzata e fu necessario aspettare quasi cento anni prima che Turing ideasse la Universal Turing Machine, un modello matematico in grado di descriverne ed astrarne il funzionamento.

L'ideazione della Universal Turing Machine, una macchina programmabile in grado di riprodurre il funzionamento di qualsiasi altra macchina programmabile (con gli stessi dispositivi di input/output), ha posto le basi per realizzare linguaggi interpretati, macchine virtuali, emulatori, VPS e business model come l'IaaS (infrastructure as a service).
Malgrado ciò, la ricerca di una definizione formale del concetto di Algoritmo ha impegnato per circa 200 anni filosofi, matematici ed informatici come Church, Gödel, Minsky e Knuth, ma a tutt'oggi non esiste una definizione universalmente accettata. La ragione di questo ritardo risiede nella confusione fra informazione e dato, la cui relazione organica e bidirezionale si ripresenta fra algoritmo e programma.

Un Algoritmo è una particolare tipologia di informazione che descrive una sequenza di azioni applicabili ad un sistema.
In quanto informazione, esiste sempre ed esclusivamente nella mente di un essere umano. La sua espressione, la sua trasformazione in dato trasferibile ed eseguibile automaticamente, è invece un Programma.
E' importante sottolineare come, contrariamente ai calcolatori meccanici precedenti, limitati alla specifica elaborazione per cui erano costruiti, le macchine Turing-complete permettono l'esecuzione di elaborazioni variabili, limitate esclusivamente dalle possibilità espressive del linguaggio in cui vengono rappresentate e dall'hardware disponibile.

Per poter intuire la portata di questa innovazione (e del potere che ne deriva) bisogna considerare che quando un computer esegue un programma, ciò che sostanzialmente è un atto di pura immaginazione della mente di un programmatore, dopo essere stato espresso in un certo linguaggio, inizia ad operare sulla realtà fisica e sociale, riproducendone la volontà ed imponendola a coloro che con tale programma interagiscono per percepire, interpretare o agire sulla realtà che li circonda.

Comprendere la differenza fra Algoritmo e Programma è dunque fondamentale per sventare i tentativi di nascondere, dietro un velo di matematica neutralità, gli interessi e la responsabilità delle scelte che il software esprime.
In quanto dato, in quanto rappresentazione, il programma è infatti soggetto a diverse categorie di errore:

1. errori di comprensione dell'algoritmo, nella mente del programmatore;
2. errori di rappresentazione dell'algoritmo, introdotti durante la scrittura del programma;
3. errori di esecuzione del programma, dovuti alla macchina che lo esegue automaticamente;
4. errori di utilizzo del programma, dovuti per lo più a problemi di documentazione o a scelte commerciali.

La seconda e la terza categoria attengono alla rappresentazione dell'algoritmo, al programma, e sono tipicamente definiti Bug. Le altre due categorie affliggono invece gli interlocutori, che l'esecuzione automatica del programma mette in comunicazione asincrona attraverso la collaborazione che veicola: i programmatori e gli utenti.

Le probabilità di queste categorie di errore variano in modo indipendente: la comprensione dell'algoritmo dipende dalla cultura del programmatore e dalla sua esperienza del problema che intende risolvere, la probabilità di errori di rappresentazione dipende anzitutto dalla dimensione complessiva del programma (che include le sue dipendenze dirette o indirette) e la probabilità degli errori di esecuzione dipende dalla complessità dell'hardware, reale o virtuale, che lo esegue. Infine la probabilità di errori nell'utilizzo di un software da parte dell'utente dipende dalla sua comprensione del suo funzionamento, ovvero degli algoritmi che questo implementa.

La probabilità che un programma funzioni correttamente corrisponde al prodotto dei complementi di queste probabilità.
La probabilità che un sistema di programmi sviluppati indipendentemente ma integrati per realizzare una determinata applicazione funzioni correttamente, decade in modo esponenziale all'aumentare dei programmi integrati.

3. PROTOCOLLI E RETI

Parallelamente allo sviluppo dei processori general purpose, il progresso dell'elettronica ha permesso di miniaturizzare una straordinaria varietà di componenti: qualsiasi smartphone è composto di microcamere, microfoni, giroscopi, ingressi USB, ricevitori GPS, Bluetooth, WiFi, LTE etc, componenti che appaiono ormai anche su automobili, forni e frigoriferi.
Tutti questi componenti registrano dati dall'ambiente e li scambiano fra loro e con sistemi automatici esterni attraverso programmi che rappresentano una particolare categoria di algoritmi: i "protocolli".

Un Protocollo definisce la sintassi e la sequenza dei messaggi che diversi sistemi automatici possono scambiare per coordinare l'esecuzione di un'elaborazione più complessa, talvolta distribuita nel tempo o nello spazio. Ciascuno di questi protocolli determina trade-off tecnici ed economici che favoriscono determinate topologie (centralizzate, decentrate, distribuite) e determinate relazioni fra i software (client/server, P2P, federazione, incompatibilità reciproche) determinando complesse relazioni di potere fra le persone che programmano, dirigono o partecipano a vario titolo nell'elaborazione.

Veicolate da un unica rete globale sempre attiva, questi programmi automatizzano contemporaneamente la distribuzione di due categorie di dati: quelli che le persone esprimono intenzionalmente e quelli che le descrivono contro la loro volontà.
Questa distribuzione di dati avviene attraverso la creazione virtualmente istantanea di innumerevoli copie non tracciabili, distribuite attraverso la rete ad un numero non determinabile di intermediari automatici, *data controller* e *data processor*.
Azioni che appaiono semplici, come la visita di una pagina Web o l'ingresso in una stanza dotata di IoT, avviano complesse elaborazioni automatiche che travalicano legislazioni e continenti, continuando ad operare per anni attraverso i profili comportamentali che alimentano. Poiché la copia dei dati è istantanea, gratuita e non lascia tracce, privacy e sicurezza dipendono dal controllo costante su chi vi ha accesso: sono Pubblici i dati reperibili attraverso informazioni pubblicamente disponibili, mentre sono Riservati quelli accessibili solo a coloro che detengono specifiche credenziali riservate.

4. COMPLESSITÀ E POTERE

> *Everyone knows that debugging is twice as hard as writing a program in the first place.*
> *So if you're as clever as you can be when you write it, how will you ever debug it?*

Quando Brian Kernighan scrisse queste parole, nel 1978, Unix consisteva di 14.795 linee per il kernel e 141.833 per tutto il resto del sistema, includendo un compilatore C, diversi linguaggi di scripting ed alcuni giochi. All'inizio del 2020, il solo kernel Linux contava quasi 28 milioni di righe di codice scritte da decine di migliaia di programmatori, con una crescita di circa 4 volte rispetto al 2005. Tuttavia un kernel, pur nella sua complessità, è un programma indipendente.

La stragrande maggioranza dei programmi esistenti dipende da altri programmi, sia durante la scrittura che per

l'esecuzione.
Su Debian GNU/Linux, Firefox dipende direttamente o indirettamente da 157 programmi diversi[1]. Chromium (Google) conta addirittura 241 dipendenze[2]. Su Android o su Windows, questi software hanno dipendenze completamente diverse.
Questa enorme complessità, in parte accidentale ed in parte scientemente perseguita, ha determinato la nascita di una nuova categoria di lavoratori, gli informatici, in grado di articolarla e dirigerla verso obbiettivi economici o politici.

Aziende ed agenzie di intelligence hanno interesse a massimizzare la diffusione dei dati espressi (contenuti o programmi) o emessi (*personal data*) dai cittadini, minimizzando al contempo la diffusione dei dati che descrivono la propria operatività (trasparenza) e controllando l'accesso ai dati che producono (*intellectual properties*).
Queste organizzazioni, ben consapevoli del fatto che la conoscenza è potere solo dove l'ignoranza è diffusa, sfruttano la complessità dell'informatica contemporanea in molti modi: per scongiurare regolamentazioni efficaci, per rendere incomprensibili le proprie clausole contrattuali, per instaurare monopoli commerciali, per costringere la stampa sulle proprie piattaforme di distribuzione (controllandone capillarmente la circolazione) o per marginalizzare il software libero attraverso software open source talmente complessi da vanificare le 4 libertà o trasformarle in privilegi elitari.

Per decenni, attraverso salari molto elevati, gratifiche, benefit aziendali ed una retorica del *culture fit* e dell'auto-imprenditorialità, le aziende del comparto IT hanno persuaso migliaia informatici a condividerne valori e obiettivi.
Tuttavia queste aziende combattono da sempre una strenua lotta contro l'autonomia intellettuale dei propri dipendenti.
Nel 2015 Apple, Google, Intel e Adobe hanno sborsato 415 milioni di dollari porre fine alla *High-Tech Employee Antitrust Litigation*, evitando una condanna che vietasse gli accordi con cui riducevano la concorrenza per i "top talent".
A fine 2019, Google ha licenziato i quattro dipendenti che avevano organizzato i *Google Walkout* contro la gestione delle molestie sessuali dell'azienda. A inizio 2020 il top management di Facebook discuteva su come realizzare e vendere un sistema di *content control*, per identificare e censurare le discussioni dei dipendenti riguardo al concetto di "*unionize*".
Sempre nel 2020, Amazon ha pubblicato due offerte di lavoro per un "*Intelligence Analyst*" ed un "*Sr Intelligence Analyst*" in grado di identificare "*labor organizing threats against the company*", per poi assumere il Generale Keith Alexander (ex-direttore del NSA). Successivamente, la diffusione di alcuni documenti riservati ha rivelato diverse attività di sorveglianza adottate da Amazon per contrastare la diffusione di contenuti sindacali ed ambientalisti fra i dipendenti, anche in Europa.

Il timore che questa ostilità evidenzia, deriva dalla consapevolezza che gli informatici costituiscono la prima categoria di lavoratori della storia a detenere il pieno controllo dei mezzi di produzione, saldamente ancorati sulle proprie spalle.

Nonostante l'egemonia culturale della Silicon Valley, è possibile osservare i segni di questa emergente coscienza di classe lungo tutta la storia dell'informatica moderna. Sin dalla caccia alle streghe degli anni '80, durante la quale molti giovani hacker e phracker statunitensi furono arrestati per aver ignorato i tabù posti a protezione dei dati di molte aziende, apparve evidente come l'accumulo di denaro non costituisse l'interesse prevalente di molti informatici.

Emblematico, già nel 1989, fu l'arresto dei membri del P.H.I.R.M., rei di aver pubblicato una guida su come accedere ai dati gestiti dal sistema di home banking di Bank Of America... dopo averlo appreso dalle brochure dell'azienda.
O le più recenti disclosure di Phineas Fisher che, nell'agosto 2014 e nel luglio 2015, pubblicò i documenti ottenuti dai sistemi informatici di due aziende (GammaGroup e HackingTeam) specializzate nella creazione di software di spionaggio e sorveglianza per regimi totalitari, rivendicando l'importanza politica del proprio gesto in una guida all'hacking informatico intitolata *HackBack! A DIY Guide for those without the patience to wait for whistleblowers*.
Nel febbraio 2018, uno studente di 22 anni, Luigi Gubello, fu denunciato dal Movimento 5 Stelle e processato per aver rivelato come tutti i dati della piattaforma Rousseau fossero pubblici, accessibili a chiunque sapesse come farseli restituire automaticamente dal software del partito, senza bisogno di credenziali private.
Simili "data-breach" vengono effettuati routinariamente da giovani sistemisti, agenzie di spionaggio e gruppi di attivisti informatici come Anonymous, che usano comuni conoscenze informatiche per ricevere automaticamente dati che gli utenti credono riservati, ma che le aziende non proteggono per non sostenere i costi di un'infrastruttura IT appropriata.

Analogamente, molti informatici consapevoli dell'inadeguatezza delle leggi sul copyright, ideato per proteggere un sistema tardo-feudale ed incompatibile con democrazie in cui la sovranità appartenga al popolo, ne combattono la natura oscurantista, sia attraverso licenze copyleft dal *reach* molto esteso (dalla AGPLv3 alla HESSLA), sia attraverso software libero (DeCSS, BitTorrent etc…) che ignora i ridicoli tabù posti a sua protezione (CSS, DRM etc...).

[1] Contando solo le dipendenze runtime: `apt-rdepends firefox-esr|grep Depends|awk '{print $2}'|sort|uniq|wc -l`
[2] Contando solo le dipendenze runtime: `apt-rdepends chromium|grep Depends|awk '{print $2}'|sort|uniq|wc -l`

Sul fronte della privacy e del controllo sociale, diversi add-ons liberi come *uBlock Origin* o *nuTensor*, vengono sviluppati da volontari per ridurre il danno prodotto da Web browser open source come Firefox o Chrome, veri e propri strumenti di sorveglianza di massa. Altri gruppi provano coraggiosamente a forkare Android, per rimuovere le funzionalità di tracciatura inserite nel sistema da Google. Altri ancora sperimentano protocolli alternativi, meno ostili agli utenti, come 9P2000, Gemini o GNUnet.

Piccole comunità di informatici provano ad attaccare il potere alla radice, sviluppando sistemi operativi distribuiti come 9front, che "fanno pensare" anteponendo la semplicità e la leggibilità dei programmi alla facilità d'uso per i profani.

Ignoti sono il numero ed il costo dei bug scientemente introdotti nel software per protesta, ed in diverse occasioni informatici indignati per il comportamento predatorio o per le scelte politiche di un'azienda software, ne hanno danneggiato fortemente la credibilità (ed i fatturati), semplicemente rimuovendo dal Web il software che avevano sviluppato o mantenuto gratuitamente per anni. Nel 2016 Azer Koçulu riuscì ad interrompere il funzionamento di milioni di siti Web rimuovendo 11 righe di JavaScript da NPM. Tre anni dopo, Seth Vargo rimosse da GitHub una propria libreria Ruby interrompendo per ore le attività di sviluppo dei clienti di Chef, fra cui Facebook, SAP, IBM e Wallmart.

5. LA LOTTA INFORMATICA... PER LA DEMOCRAZIA

Appare dunque evidente che i valori, le istanze e le metodologie di lotta degli informatici, esattamente come le strutture di potere con cui si confrontano, non sono completamente riconducibili alle categorie novecentesche.
In una società cibernetica, chi non è in grado di comprendere e modificare il funzionamento degli automatismi da cui è circondato, diventa un ingranaggio inconsapevole nelle mani di chi li controlla. Invece di inseguire una frettolosa "alfabetizzazione digitale" in un contesto che è ancora ai geroglifici, la lotta informatica per la democrazia richiede

1. nuovo software, che anteponga la semplicità intrinseca ad una manipolativa facilità d'uso;
2. nuove leggi, che sostituiscano l'*IP protection* tardo-feudale con norme al servizio di un popolo davvero sovrano;
3. nuove reti FTTH, che permettano ai cittadini il completo controllo fisico dell'hardware che elabora i propri dati;
4. una nuova didattica, che permetta ai cittadini di comprendere la società cibernetica in cui vivono, partecipando attivamente alla sua costruzione: in una Democrazia cibernetica, la Scuola rimane più importante del Parlamento.

L'alternativa è un mondo in cui un giovane giornalista come Daniel Verlaan, può accedere ad una videoconferenza segreta dei Ministri della Difesa dell'Unione Europea, trovando su Twitter credenziali fotografate dal Ministro olandese.
Un mondo in cui tale intromissione produce grandi risate, ma nessun interrogativo su chi altri abbia avuto accesso a simili riunioni, "segrete" ma pubbliche, perché visibili per chiunque sappia come interrogare gli automatismi che le veicolano.

La posta in gioco è alta: dobbiamo scegliere fra una democrazia cibernetica ed una oligarchia del controllo globale.

È la scelta fra diventare hacker o robot. Cittadini o schiavi.

BIBLIOGRAFIA

[1] Bietti, Elettra. From Ethics Washing to Ethics Bashing: A View on Tech Ethics from Within Moral Philosophy. Proceedings to ACM FAT* Conference. 2020 https://ssrn.com/abstract=3513182

[2] Brooks, Frederick P. No Silver Bullet: Essence and Accidents of Software Engineering. IEEE Computer 20 (1987) https://web.archive.org/web/20160910002130/http://worrydream.com/refs/Brooks-NoSilverBullet.pdf

[3] Dark Creaper. Hacking Bank Of America's Home Banking System. 1989. http://www.textfiles.com/hacking/boahack.txt

[4] Dijkstra, Edsger W. On the nature of Computing Science. 1984. (circolato privatamente) https://www.cs.utexas.edu/users/EWD/transcriptions/EWD08xx/EWD896.html

[5] Doctorow, Cory E. IP. Locus Magazine 716 (2020) https://locusmag.com/2020/09/cory-doctorow-ip/

[6] Fisher, Phineas. HackBack! A DIY Guide For Those Without The Patience To Wait For Whistleblowers. 2014. https://www.exploit-db.com/papers/41913

[7] Kernighan, Brian W. e P. J. Plauger. The Elements of Programming Style, 2nd Edition. McGraw Hill, New York, 1978

[8] Knuth, Donald E. Computer Science and Its Relation to Mathematics. The American Mathematical Monthly, vol. 81 (1974) https://www.maa.org/sites/default/files/pdf/upload_library/22/Ford/DonaldKnuth.pdf

[9] Krug, Steve. Don't make me think: a common sense approach to Web usability. New Riders. 2005

[10] Lessing, Lawrence. Code: Version 2.0. New York: Basic Books. 2006. http://codev2.cc/download+remix/Lessig-Codev2.pdf

[11] Mahieu, René L. P. e Jef Ausloos. Harnessing the collective potential of GDPR access rights: towards an ecology of transparency. Internet Policy Review. 2020. https://policyreview.info/articles/news/harnessing-collective-potential-gdpr-access-rights-towards-ecology-transparency/1487

[12] Morozov, Evgeny. The Meme Hustler - Tim O'Reilly's crazy talk. The Baffler. 2013 https://thebaffler.com/salvos/the-meme-hustler

[13] Snowden, Edward J. Permanent Record. Metropolitan Books. 2019

[14] Stallman, Richard. Why Open Source misses the point of Free Software. (last update on: 2020/01/07 13:55:48) https://www.gnu.org/philosophy/open-source-misses-the-point.html.en

[15] The Mentor. The Conscience of a Hacker. Phrack Inc. Volume One, Issue 7, Phile 3 of 10 (1986) http://www.phrack.org/archives/issues/7/3.txt