Editoriale

Intelligenza artificiale o buonsenso naturale?

Nei primi anni '70, dopo aver passato tre anni all'Università di Stanford nella fucina di John McCarty, rientrava al Politecnico di Milano un giovane ricercatore brillante ed entusiasta. Sulla porta del suo studio mise uno striscione con la scritta "MP-AI Milan Politechnic Artificial Intelligence Lab". Il giorno dopo, sulla porta dello studio di fronte, comparve uno striscione con la scritta "Politecnico di Milano – Laboratorio di Intelligenza Naturale"!

Questo episodio dimostra come, sin dal suo inizio, la cosiddetta Intelligenza Artificiale (IA o AI per gli Inglesi) abbia dovuto superare più di qualche diffidenza e bonaria ironia; il suo sviluppo ha subito accelerazioni e frenate anche in relazione al progredire delle architetture e dei dispositivi di elaborazione – tipico esempio le reti neurali [Sami91]. Dall'incontro a Darmouth College durante il quale McCarty coniò il termine per indicare un insieme di studi sulla risoluzione di problemi (giocatore di scacchi), sul trattamento del linguaggio naturale (ELIZA), sistemi esperti (Mycin) e altri rimasti tuttavia in un ambito per lo più accademico fino agli anni '80, si è passati ad applicazioni industriali nell'ambito della robotica, del riconoscimento del parlato e delle immagini, dell'apprendimento automatico mediante reti neurali.

Oggi la situazione quindi sembra decisamente cambiata e *Intelligenza Artificiale* è diventata una buzzword da utilizzare ogni volta che si cita una applicazione informatica; paradossalmente, stampa, radio e televisione molto spesso citano l'IA anche per indicare una semplice "App" in grado di fare 2+2 su uno smartphone! D'altra parte esistono molte applicazioni sofisticate in sistemi in tempo reale, auto-adattativi, sensibili al contesto, che non fanno uso di tecniche di IA.

Una parte di colpa ricade sugli stessi ricercatori che, fin dall'inizio, preconizzarono "… magnifiche sorti e progressive …" del loro lavoro nel sostituire molte attività umane con applicazioni di IA [R&N95], il che, per

contrasto, ha generato una sorta di luddismo aprioristico in gran parte ingiustificato. E' vero però che oggi ci troviamo in una bolla inflattiva e che in molti casi il prodotto "non vende" se non comprende almeno una rete neurale o non adotta tecniche di "deep learning"!

Cerchiamo quindi di non farci travolgere da eccessivi entusiasmi e di mantenere una sana dose di "Intelligenza Naturale" e critica. Innanzitutto, l'uso di sistemi di apprendimento automatico è fortemente condizionato dalla fase di addestramento; questa dipende dalla qualità dei dati che vengono forniti come esempio e che possono contenere errori o rappresentare solo parzialmente la realtà, creando quindi delle distorsioni (bias) non volute nella fase operativa. Lo sviluppo di molte applicazioni fortemente innovative ha inoltre sollevato seri problemi di Etica, a partire da quello fondamentale – emerso chiaramente quando i fisici persero la loro "innocenza" con le bombe di Hiroshima e Nagasaki -: si DEVE fare tutto ciò che SI PUO' fare? Ma, a prescindere dalle posizioni filosofiche, ci sono problemi etici e legali anche in questioni più spicciole; quando una vettura autonoma e senza pilota causasse un grave incidente stradale chi dovrebbe essere incolpato: il proprietario della vettura, il costruttore, l'analista e il programmatore del sistema di controllo/comando [Ruffolo17]? Un bel problema anche dal lato assicurativo e non vorrei che la proliferazione di "class action" provocasse una carenza di lavoratori del settore, così come stiamo assistendo all'abbandono di chirurghi e ginecologi. Anche da un punto di vista strettamente tecnico, le reti neurali e le tecniche di apprendimento automatico applicate a taluni settori , quali ad esempio il riconoscimento di immagini, funzionano molto bene, ma lavorano come una "scatola nera" e non si riesce a capire *come* e *perché* producano il risultato. Ciò è un grave limite al loro uso in applicazioni "safety-critical", nelle quali la tracciabilità delle procedure è un *mantra* finora indiscusso.

Ma non solo l'Etica è in questione. Un recente studio di ricercatori dell'Università del Massachussetts ad Amherst [Strubel19] ha calcolato, mediandolo tra programmi di learning diversi, che il processo di addestramento di una rete neurale per l'elaborazione del linguaggio naturale genera, per il lavoro dei server, circa 635 Kg di CO_2, pari alla quantità attribuibile, per passeggero, ad un volo di andata e ritorno che attraversi gli USA, mentre la costruzione e l'addestramento per sei mesi di un modello usabile in pratica genera circa 35.000 Kg di CO_2! Anche l'Ecologia quindi fa capolino nel *bilancio complessivo* dei pro e contro le applicazioni di apprendimento automatico.

Mi piacerebbe sapere che cosa penserebbe oggi quel giovane ricercatore, il Prof. Marco Somalvico, prematuramente scomparso 17 anni fa, dopo aver dato un sostanziale contributo alla ricerca in AI in Italia.

Fabio A. Schreiber

[R&N95] S. J. Russel, P. Norvig - Artificial Intelligence: A Modern Approach – Prentice Hall, 1995

[Ruffolo17] U. Ruffolo (Ed.) – Intelligenza Artificiale e Responsabilità – Giuffrè, 2017

[Sami91] M. Sami, J. Calzadilla-Daguerre (Ed.) – Silicon Architectures for Neural Nets – IFIP North-Holland, 1991

[Strubel19] E. Strubel, A. Ganesh, A. McCallum – Energy and Policy Considerations for Deep Learning in NPL – arXiv: 1906.02243v1 [cs.CL], 5 Jun 2019

Sicurezza del software: l'approccio linguistico

Gabriele Costa, Pierpaolo Degano e Letterio Galletta

A Remo Bodei che ci ha lasciato

quando aveva ancora tanto da insegnarci

Sommario

La sicurezza *dei sistemi software è di importanza capitale, sia per gli individui che per la società e le sue infrastrutture. È quindi necessario che i programmi che sorreggono le nostre attività siano costruiti in modo da non offrire appigli a chi ci vuole attaccare. Esamineremo brevemente i metodi e gli strumenti tipici dei linguaggi di programmazione che assistono progettisti e programmatori nella realizzazione di componenti software sicure, e le principali tecniche che vengono impiegate per verificare tali proprietà, soprattutto quando il software viene acquisito e non sviluppato ex novo.*

Abstract

Security of software systems is of paramount importance, for both individuals, and the society and its infrastructures. The programs that support our activities must therefore be built robust against attackers. We will briefly survey the methods and the tools offered by programming languages to assist designers and programmers in implementing secure software components. We also discuss the main techniques for verifying the security properties of these components, especially when they are not developed from scratch.

Keywords: software security, programming languages, static and dynamic program analysis

1. Introduzione

Il software è dappertutto, dalle enormi batterie di elaboratori ai telefoni cellulari sempre più sofisticati, ai microprocessori inseriti negli oggetti d'uso quotidiano. Il software non pesa, è impalpabile, non si vede se non quando il programmatore o il progettista lo realizza, sia scrivendolo ex novo, sia componendo pacchetti esistenti, ma sempre usando uno o più linguaggi di programmazione.

Il software deve garantire che risolve il problema per cui è stato scritto o che il sistema che governa si comporti proprio come ci si aspetta, in altre parole che rispetti le specifiche funzionali poste al progettista. Ci sono altre proprietà importanti che un buon software deve possedere, quali l'efficienza, la leggibilità, la modificabilità e altri ancora, spesso chiamati *non funzionali*. Tra questi la sicurezza è di importanza capitale.

Infatti, i giornali sono sempre più ricchi di esempi di attacchi a sistemi non sicuri e delle conseguenze sui malcapitati che ne sono vittima. Non è necessario insistere sui rischi che si corrono quando la sicurezza sia scarsa, per cui discuteremo solo alcuni casi. Forse gli attacchi più frequenti riguardano la violazione della riservatezza delle persone i cui effetti più eclatanti, e a volte drammatici, derivano dalle cosiddette forme di stalking rese possibili attraverso i social media e alla condivisione di testi e immagini. Attacchi alla segretezza dei dati, per esempio ai messaggi di posta elettronica, hanno favorito campagne politiche o peggio, si pensi a WikiLeaks. Vi sono poi attacchi che possono mettere direttamente a rischio le nostre vite. È di qualche anno fa la scoperta che il sistema software di un'automobile era permeabile, per cui un attaccante era riuscito a prendere il controllo del sistema di condizionamento, poco male, e anche dell'acceleratore, dei freni e del volante, assai peggio.[1] Anche gli aeroplani sono soggetti a questo rischio: un attaccante può in alcune situazioni falsificare le risposte degli strumenti di bordo con evidenti conseguenze. [2]Ulteriori attacchi forse ancora più pericolosi sono quelli condotti contro reparti o installazioni militari o impianti chimici o nucleari (i danni alle turbine per la purificazione dell'uranio che rallentarono il programma nucleare iraniano furono dovuti a un attacco informatico, Stuxnet Worm [1]).

A lungo i linguaggi di programmazione sono stati progettati per sostenere i loro utenti nella realizzazione di sistemi funzionalmente corretti, nell'ipotesi che questi siano chiusi, ovvero che ricevano i dati di ingresso e li elaborino fornendo i risultati richiesti. Gli aspetti di sicurezza non erano quindi considerati cruciali, come invece lo sono diventati al momento in cui i sistemi sono aperti, nel senso che operano in un ambiente non noto a priori, a volte chiamato cyberspazio. A sua volta questo ambiente è costituito da un gran numero di apparecchiature hardware e da un numero anche maggiore di altri sistemi software, e sia gli uni che gli altri sono altrettanto sconosciuti e di tipologie diverse. Il numero delle apparecchiature

---

[1] *Si veda https://www.wired.com/2015/07/hackers-remotely-kill-jeep-highway e per una lista di altri casi anche recentissimi https://www.upstream.auto/research/automotive-cybersecurity*

[2] *Un esempio è in https://thehill.com/policy/cybersecurity/455319-dhs-warns-of-cybersecurity-vulnerability-in-small-airplanes*

connesse in rete è enorme, stimato intorno ai 20 miliardi a tutt'oggi, con un incremento sbalorditivo.[3] Inoltre i sistemi software diventano sempre più ingombranti e intricati – si pensi ad esempio che il software dello Space Shuttle aveva 400K linee di codice, e che oggi un'automobile ne ha tipicamente 100M!

Il cyberspazio offre un'infrastruttura con cui i vari sistemi interagiscono, scambiandosi non solo dati e informazioni, ma anche codice eseguibile. È quindi cruciale capire che cosa un sistema sicuro debba proteggere e quali siano i potenziali attaccanti, ovvero il threat model, in base al quale progettare una Trusted Computing Base (TCB), ovvero una combinazione di hardware e software che garantisca la sicurezza dei dati e dei programmi che ospita, anche quando immersa in un ambiente ostile e inaffidabile.

Per ottenere questo risultato sono necessari linguaggi di programmazione che sostengano i loro utenti nella realizzazione e la composizione di sistemi sicuri dall'inizio, piuttosto che modificarli di volta in volta quando si scoprono falle e vulnerabilità. Diventa allora fondamentale progettare e realizzare linguaggi di programmazione che includano in forma nativa meccanismi, primitive e strumenti di analisi per controllare l'uso di dati e codice e per verificare che non ne venga fatto un uso improprio.

Di seguito ci concentreremo sull'approccio alla costruzione e verifica di sistemi sicuri, chiamato *language-based security* [2], evidenziandone due filoni; tralasceremo invece la sicurezza dei sistemi operativi o dei dispositivi fisici. Il primo filone della language-based security si occupa dei meccanismi, primitive e tecniche per scrivere ex novo programmi, o parti di essi che siano sicuri. Il secondo filone pone la sua attenzione sul controllare la sicurezza di componenti software che altri hanno scritto, sia per assemblare sistemi più complessi, sia per usarli così come sono. Tali componenti potrebbero contenere delle *vulnerabilità*, cioè abilitare l'esecuzione di operazioni pericolose che causano comportamenti indesiderati o dannosi.

Vedremo che questi due approcci, l'uno più orientato a chi sviluppa software e l'altro a chi lo usa, non sono affatto disgiunti, e anzi usano un corpus comune di tecniche, analisi e trasformazioni che controllano e impongono ai programmi risultanti un comportamento sicuro.

Riquadro 1: I buffer overflow una classica vulnerabilità

Una delle vulnerabilità più note è sicuramente il buffer overflow. Il primo attacco basato questa vulnerabilità risale al 1988 con il Morris Internet Worm [3].

Questa vulnerabilità è estremamente comune ed è stata riscontrata in qualunque tipo di software, dai sistemi operativi alle applicazioni desktop e a quelle mobili. La sua causa principale è dovuta all'uso di linguaggi che non sono sicuri (safe) e a una scarsa validazione dei dati di ingresso. Un buffer overflow si ha quando un programma scrive dei dati fuori dallo spazio di memoria previsto, finendo così per sovrascrivere altre zone che sono riservate a contenere altri dati o, peggio ancora, informazioni essenziali per la corretta

[3] *Si consulti https://www.ericsson.com/en/mobility-report/internet-of-things-forecast*

esecuzione del programma, come l'indirizzo di ritorno di una funzione. Un attaccante che riesce a controllare i dati che vengono scritti sul buffer e nelle aree di memoria limitrofe può assumere il controllo dell'esecuzione del programma, e gli può far eseguire il codice che vuole. Un tipico esempio è quando l'attaccante inietta uno shellcode[4] da qualche parte in memoria (nel buffer vulnerabile o in una variabile di ambiente) e tramite il buffer overflow sovrascrive l'indirizzo di ritorno della funzione corrente con l'indirizzo di memoria dove si trova lo shellcode. Quando la funzione corrente ritorna, verrà eseguito il codice dello shellcode invece che quello della funzione chiamante. Per contrastare questo tipo di vulnerabilità negli anni sono state adottare vari meccanismi di mitigazione (vedi Riquadro 2).

Riquadro 2: Mitigazione dell'attacco buffer overflow a livello del sistema operativo

Una delle proposte maggiormente utilizzata dai sistemi operativi moderni con l'aiuto anche da parte di meccanismi hardware, integrati ad esempio nella unità di traduzione degli indirizzi (MMU), per contrastare un attacco basato su buffer overflow[5] è il *Data Execution Prevention* (DEP), la cui idea principale è di marcare come eseguibili solo quelle aree della memoria che contengono il codice prodotto dal compilatore. In questo modo quando l'attaccante cerca di dirottare l'esecuzione allo shellcode, l'attacco fallisce perché il programma viene bloccato. Un'altra mitigazione molto comune è quello delle *stack canary*, che sono dei valori sentinella posti a guardia delle aree di memoria che contengono informazioni importanti per la computazione, come ad esempio l'indirizzo di ritorno delle funzioni. L'idea è che un attaccante non possa dirottare la computazione al codice "malevolo" senza alterare il valore delle canary: è sufficiente quindi verificare se la canary è stata modificata per individuare l'attacco. Infine, un'altra mitigazione molto comune è *l'Address Space Layout Randomization* (ASLR) che consiste nell'aggiungere un offset generato casualmente all'indirizzo base a partire dal quale sono allocati i dati in memoria. In questo modo diventa difficile per l'attaccante dirottare la computazione verso lo shellcode perché il suo indirizzo in memoria è stato determinato a caso e quindi difficile da ricavare.

Queste mitigazioni sono ormai utilizzate da tutti i sistemi operativi. Tuttavia, sebbene rendano la vita dell'attaccante molto difficile, non risolvono il problema del buffer overflow alla radice. Per fare ciò la soluzione migliore è scrivere i propri programmi con linguaggi memory safe, garantendo così che un buffer overflow non può mai accadere.

---

[4] Lo shellcode è il codice binario che l'attaccante inietta nel programma vittima. Si chiama così perché storicamente questo codice avviava un terminale (la shell).

[5] Di recente sono state proposte anche soluzioni hardware che garantiscono dinamicamente la spatial safety. Ad esempio, la tecnologia Intel MPX (https://intel-mpx.github.io/) introduce istruzioni e registri per contenere informazioni sui bounds dei puntatori: qualunque accesso fuori dai bounds causa una eccezione hardware che sarà gestita dal sistema secondo le politiche più appropriate, tipicamente interrompendo l'esecuzione.

2. Scrivere codice sicuro

Le vulnerabilità di sicurezza possono essere trovate in qualsiasi software. Tuttavia, il linguaggio di programmazione che viene utilizzato per lo sviluppo ha un impatto su quanto è facile introdurre errori che diano origine a vulnerabilità. Tradizionalmente i linguaggi di programmazione si classificano in linguaggi ad alto o a basso livello in base a quanti dettagli della macchina fisica vengono nascosti. Ad esempio, un linguaggio come Java, che fornisce costrutti per definire classi e che fornisce un meccanismo di gestione della memoria automatico, è generalmente considerato più di alto livello di un linguaggio come il C che invece richiede al programmatore di gestire manualmente la memoria.

Una classificazione simile dei linguaggi esiste anche per quanto riguarda la sicurezza. In questo caso, i linguaggi vengono classificati come *safe* and *unsafe*. Intuitivamente, un linguaggio safe permette al programmatore di scrivere, tramite appositi costrutti, programmi che soddisfano certi requisiti di sicurezza, oppure fornisce degli strumenti, generalmente inclusi all'interno del compilatore (o dell'interprete), che impongono specifiche politiche di sicurezza. Invece, i linguaggi unsafe non forniscono tali meccanismi o non ne forniscono a sufficienza. Adottare, quindi, un linguaggio safe permette di scrivere generalmente software più sicuro.

A questo punto la domanda è quali caratteristiche deve avere un linguaggio per essere definito safe? Generalmente un linguaggio è definito safe quando garantisce la *memory safety* e la *type safety*. Intuitivamente, memory safety significa che tutti i programmi esprimibili nel linguaggio accedono solo alle zone di memoria che sono permesse loro di accedere. Invece, type safety significa che i dati in memoria sono acceduti secondo il loro tipo e che eventuali conversioni di tipo (type cast) rispettano le gerarchie di tipo (per esempio il tipo numerico *real* è considerato essere a un livello gerarchico superiore al tipo *integer*). Nel resto della sezione spiegheremo come generalmente vengono ottenute questi due tipi di safety e quali sono le loro implicazioni. Queste tuttavia non esauriscono il panorama delle proprietà di safety, tra le quali ci limitiamo a citare la *control flow integrity* [4] e alcune forme di controllo degli accessi quali *la stack inspection* [5].

2.1 Memory safety

La *memory safety* è strettamente legata alla nozione di puntatore. Comunemente si definisce un puntatore come un particolare tipo di dato che contiene un indirizzo di memoria e che permette di accedere in modo indiretto sia a dati che a codice; inoltre ai puntatori è associato un tipo che dà semantica alla memoria puntata. Linguaggi diversi forniscono operazioni diverse per i puntatori. Per esempio, in C/C++ l'aritmetica dei puntatori permette di incrementare e decrementare l'indirizzo contenuto nel puntatore che può così muoversi liberamente all'interno della memoria.

Per capire meglio il ruolo che giocano i puntatori per la memory safety è conveniente adottare un approccio di più astratto. In particolare, adottiamo la visione descritta da Michael Hicks [6] e consideriamo i puntatori come delle *capabilities* per accedere a certe zone della memoria. L'idea è che ogni oggetto

che viene allocato in memoria riceve una capability che descrive il suo tipo, la regione di memoria in cui risiede (denotata dai bounds), e sua presenza o meno in memoria (*validity*). Ogni volta che un puntatore viene assegnato acquisisce le capabilities del dato puntato. Un puntatore è memory safe se rispetta due principi: *spatial safety*, ovvero gli accessi mediante il puntatore riguardano la porzione di memoria entro i bounds, e *temporal safety*, ovvero tali accessi avvengono solo quando l'oggetto è allocato. Si noti che un puntatore può puntare anche fuori dai bounds, ma per non violare la spatial memory safety non può essere usato per accedervi; allo stesso modo un puntatore può avere come valore un indirizzo di memoria a un dato che è stato de-allocato, ma per non violare la temporal memory safety non deve essere usato per accedervi.

Naturalmente i linguaggi funzionali, quali Haskell, OCaml o F#, sono memory safe per definizione, poiché non hanno una nozione esplicita di puntatore, visto che operano solamente su valori, e tutte le operazioni che coinvolgono la memoria sono trasparenti al programmatore e sono gestite dal runtime support del linguaggio.

Sono memory safe anche i linguaggi a oggetti, quali Java, C#, Python, e questa proprietà viene garantita attraverso un uso disciplinato dei puntatori. Più precisamente, in questi linguaggi l'indirizzo a una zona di memoria non è un valore direttamente manipolabile dal programmatore (in gergo si dice che è un valore denotabile ma non esprimibile): infatti, in questi linguaggi si parla spesso di riferimenti e non di puntatori. Tipicamente, i riferimenti puntano a un oggetto allocato in memoria, ma l'unica operazione che il programmatore può esprimere nel linguaggio è l'accesso all'oggetto: non c'è un'aritmetica dei puntatori. Inoltre, questi linguaggi forniscono un operatore `new` per la creazione di nuovi oggetti che si occupa di allocare e inizializzare la memoria necessaria a contenere il nuovo oggetto. Non esistono costrutti per de-allocare la memoria, che è gestita esclusivamente dal runtime support. Questo libera la memoria non più usata attraverso il garbage collector e controlla che gli accessi avvengano entro i bound, segnalando a tempo di esecuzione eventuali violazioni. Un esempio interessante di linguaggio memory safe è Rust[6] progettato per la programmazione di sistema: il meccanismo che garantisce la safety è completamente statico essendo demandato ad un componente del compilatore chiamato *borrow checker*. La sua caratteristica principale si basa sul concetto di *ownership*, che implementa un particolare modello di gestione delle risorse (i sistemi di tipo affini [7]). L'idea di base è che la responsabilità della gestione di un dato allocato in memoria è demandata a un singolo puntatore che è chiamato *owner.* L'owner è l'unico puntatore che può accedere al dato. Quando un puntatore termina il suo ciclo di vita (*esce dallo scope*) perché l'esecuzione sta uscendo da un blocco o ritornando da una chiamata a funzione, il dato associato viene de-allocato. Una caratteristica importante di questo meccanismo è che in ogni istante per ogni dato esiste sempre un unico owner: l'assegnamento tra puntatori trasferisce l'ownership del dato al puntatore assegnato. Dopo l'assegnamento qualunque accesso al dato tramite il

[6] *Si veda https://www.rust-lang.org/*

puntatore precedente genera un errore di compilazione: si dice che l'assegnamento ha una *move semantics*. Sebbene garantisca una gestione safe dei dati allocati in memoria, questo meccanismo può risultare poco flessibile in certi casi, ad esempio quando vogliamo passare un puntatore a una funzione senza trasferire l'ownership del dato alla funzione chiamata. Per questo motivo, Rust fornisce anche un meccanismo di delega, chiamato *borrowing*, che permette di trasferire temporaneamente la responsabilità del dato ad altri puntatori: quando il puntatore che ha ricevuto la delega esce dallo scope, il dato non viene de-allocato, e l'ownership ritorna al precedente owner. Anche in questo caso qualunque tentativo di accesso al dato tramite un puntatore diverso da quello che ha la delega causa un errore di compilazione. Il meccanismo fornito da Rust è così potente che non solo garantisce la memory safety, ma anche l'assenza di *data race*, aiutando il programmatore a scrivere programmi concorrenti più corretti.

Esempi di linguaggi non memory safe sono C e C++ perché delegano completamente al programmatore la gestione della memoria. Ci sono alcune proposte per mitigare questa mancanza, sia dal mondo industriale che dal mondo accademico. Ad esempio, le più recenti versioni del C++ hanno introdotto diversi tipo di *puntatori smart* (*smart pointer*[7]). Uno smart pointer è un tipo di dato astratto che estende i puntatori con ulteriori funzionalità come il controllo dei bound e la gestione automatica della memoria. Nelle versioni moderne del C++ gli smart pointer permettono di implementare particolari idiomi di programmazione che aiutano a limitare gli errori nella gestione della memoria e più in generale delle risorse. L'idioma maggiormente utilizzato è chiamato *Resource Acquisition Is Initialization* (*RAII*). L'idea di base è legare la gestione di una risorsa al ciclo di vita di un oggetto allocato sullo stack: la memoria sullo heap è allocata quando l'oggetto sullo stack è creato ed è rilasciata quando l'oggetto esce dallo scope. Usando questo idioma e gli smart pointer è quindi possibile semplificare la gestione della memoria mediante un semplice meccanismo di ownership per i dati allocati sullo heap. La libreria standard del C++ fornisce diversi tipi di smart pointer che implementano diverse politiche.[8] Sebbene questi meccanismi aiutino a scrivere codice più safe, sono di difficile applicazione se questo deve essere retro-compatibile con versioni vecchie del linguaggio, e perdono la loro efficacia quando è necessario interagire con codice legacy che usa i "vecchi" puntatori non smart. Inoltre, alcuni programmatori hanno criticato tali meccanismi perché non aiutano veramente a scrivere codice più sicuro, ma anzi possono aprire la strada a ulteriori vulnerabilità [8].

Anche il mondo accademico ha proposto delle possibili soluzioni per ottenere la memory safety nei linguaggi C/C++. Grossolanamente, possiamo distinguere due approcci. Il primo considera la rimozione o limitazione dei costrutti unsafe creando dei dialetti safe. Il secondo invece si concentra su tecniche per

[7] *Si veda http://www.cplusplus.com/reference/memory/*

[8] *Si veda https://docs.microsoft.com/en-us/cpp/cpp/smart-pointers-modern-cpp?view=vs-2019 per una panoramica delle diverse possibilità*

instrumentare il codice, ovvero per inserirvi dei controlli che regolamentino l'utilizzo dei costrutti unsafe.

Un esempio del primo approccio è Cyclone [9] che fornisce al programmatore un sottoinsieme safe del linguaggio C. In particolare, Cyclone inserisce automaticamente controlli per quanto riguarda i bound degli oggetti allocati in memoria e per i puntatori a NULL. Inoltre, limita l'aritmetica dei puntatori, richiede che i puntatori siano inizializzati prima del loro uso e fornisce una particolare analisi statica, chiamata *region analysis*, che aiuta a prevenire *dangling pointer* (puntatori che indirizzano oggetti non piú allocati). Il progetto di Cyclone ha ispirato il meccanismo di ownership di Rust.

Un esempio abbastanza influente del secondo approccio è CCured [10], la cui idea di base è usare un sistema di tipo per analizzare il codice e individuare quelle che parti possono violare la memory safety a tempo di esecuzione e instrumentarle con controlli dinamici. Più precisamente l'analisi di CCured classifica i puntatori in tre tipologie in base a come vengono utilizzati nel codice e adotta una nozione di *fat pointer*, cioè un puntatore che oltre all'indirizzo di memoria contiene anche dei metadati. I puntatori non soggetti all'aritmetica e non coinvolti in conversioni di tipo sono classificati come SAFE. L'uso di questi puntatori viene instrumentato solamente aggiungendo un controllo che garantisca che il puntatore non sia `NULL`. Invece, i puntatori che sono coinvolti in operazioni di aritmetica sono classificati come SEQ e vengono instrumentati aggiungendo dei controlli sui bound. Infine, i puntatori che sono coinvolti in conversioni di tipo sono classificati come WILD e il codice relativo viene instrumentato sia con controlli sui bound sia sul tipo del dato puntato.

Tuttavia, uno degli svantaggi dell'approccio proposto da CCured è che il codice prodotto non è facilmente interoperabile con codice esistente o codice legacy perché non è possibile utilizzare insieme fat pointer e puntatori normali. Per superare queste limitazioni, recentemente è stato proposto CheckedC [11]. L'idea alla base di CheckedC è simile a quella di CCured, ma il meccanismo implementato permette di mantenere la retro-compatibilità tra i fat pointer e i normali puntatori.

Altri sistemi hanno cercato di garantire l'interoperabilità memorizzando i metadati dei fat pointer in delle strutture apposite. Ad esempio, questo è l'approccio di SoftBound [12], una tecnica di instrumentazione di programmi C per la spatial memory safety. Infine, CETS [13] estende SoftBound per garantire anche la temporal safety.

2.2 Type safety

Un *tipo* è un'informazione che rappresenta un insieme di valori. Ogni dato è rappresentato in memoria come una sequenza di bit e intuitivamente il tipo permette di dare una semantica a quei bit, per esempio specificando se rappresentano un intero o un valore in virgola mobile. Un linguaggio sia dice tipato quando ogni valore manipolabile ha un tipo. I linguaggi di programmazione generalmente definiscono dei tipi primitivi, come gli interi, e forniscono al programmatore dei costrutti che permettono loro di definire i propri tipi di dato. Ad esempio, molti linguaggi ad oggetti, come Java e C#, forniscono

la nozione di classe che permette la definizione di nuove classi e di conseguenza di nuovi tipi. Invece, molti linguaggi funzionali permettono di definire nuovi tipi tramite il meccanismo dei tipi di dati algebrici.

I linguaggi che sono type safe garantiscono che i dati in memoria sono acceduti sempre secondo il tipo corretto. Per fare questo, tali linguaggi definiscono un sistema di tipo, che è un insieme di regole che descrivono quali sono i modi corretti con cui i programmi possono manipolare i dati, per esempio quali operazioni sono ammissibili sui dati di un certo tipo. Il vantaggio fondamentale di usare un linguaggio tipato è che il sistema di tipo permette di scartare programmi che non rispettano le regole di tipaggio. È importante notare che non tutti i linguaggi tipati sono type safe, perdendo qualunque vantaggio dato dai tipi. Per esempio, il C è un linguaggio tipato, ma fornisce un meccanismo di conversione che permette di trasformare un puntatore a dati di tipo intero in uno a dati di tipo funzione.[9] Linguaggi come Java, C#, Python e OCaml sono invece type safe.

Generalmente, i linguaggi vengono classificati in tipati staticamente e dinamicamente. Nel primo caso durante la compilazione viene eseguita una fase di controllo dei tipi che scarta i programmi che non la superano, interrompendo la loro compilazione. Nel secondo caso i tipi sono controllati mentre il programma è in esecuzione e una violazione di tipo interrompe l'esecuzione del programma. La scelta tra i due approcci ha da sempre animato il dibattito nella comunità di chi progetta linguaggi di programmazione. Indubbiamente, controllare i tipi staticamente ha il vantaggio di individuare gli errori in anticipo, permette di evitare controlli a tempo di esecuzione e i tipi forniscono una forma di documentazione del codice. Controllare i tipi dinamicamente richiede un certo overhead a tempo di esecuzione, ma in genere permette di tipare più programmi. Nella pratica la maggior parte dei linguaggi adottano entrambi gli approcci. Ad esempio, in Java la macchina astratta del linguaggio tiene traccia del tipo degli oggetti e lo usa per prevenire certi tipi di *unsoundness*. L'esempio tipico è quando il programmatore manipola un array di oggetti di tipo `Object` usando un puntatore. In questo caso, la macchina virtuale controlla che quando assegniamo un oggetto `o` a un elemento dell'array, il tipo di `o` sia uguale al tipo degli elementi dell'array.

Il sistema di tipo viene definito dal designer del linguaggio di programmazione, usualmente con una specifica logica dalla quale derivare l'algoritmo di tipaggio (v. riquadro 3). Una volta che il sistema di tipo è stato specificato logicamente è necessario assicurarsi della sua correttezza attraverso una dimostrazione matematica (*type soundness*) che garantisce che i programmi ben tipati non solleveranno errori (di tipo) a tempo di esecuzione.

La type safety può essere la base per garantire che i programmi soddisfano altre proprietà di sicurezza, per esempio la confidenzialità dei dati e anche la loro integrità, cioè che non sono stati manipolati da un attaccante. Per la segretezza, l'idea è quella di definire dei tipi speciali che rappresentano il livello di confidenzialità dei dati, ad esempio, pubblici o segreti. Il sistema di tipo in

[9] *Alcuni moderni compilatori C si limitano ad emettere un warning indicando che si sta effettuando una operazione non safe, ma comunque la compilazione termina con successo.*

questo caso garantisce che un programma ben tipato non assegna mai a una variabile pubblica un valore segreto [14]. Nel secondo caso, i dati di ingresso di un programma sono considerati untrusted e se un programma è ben tipato, allora tali dati non sono direttamente utilizzati in computazioni rilevanti dal punto di vista della sicurezza, a meno di essere stati elaborati da opportune procedure che ne verificano l'integrità [15].

Riquadro 3: Specifica logica di un sistema di tipi

Un sistema di tipi è una particolare analisi statica che associa un tipo a ogni valore calcolato e che verifica che l'utilizzo di questi valori non causa errori (di tipo) a tempo di esecuzione. Il concetto di *errore di tipo* è fortemente dipendente dall'analisi che si definisce. Nel caso più semplice siamo interessati ad avere la certezza che ogni operazione sia applicata a dati del tipo giusto, perché "non si sommano le pere con le mele." Formalmente, un sistema di tipi è un particolare sistema di deduzione logica completamente automatizzato, la cui definizione si può suddividere in tre fasi. Nella prima si definisce l'insieme di tutti i possibili tipi specificando la loro sintassi attraverso una grammatica formale. Questa grammatica descrive i tipi di base, ad esempio *pubblico* e *privato*, e come è possibile costruire nuovi tipi a partire da tipi già esistenti usando i cosiddetti costruttori di tipo, ad esempio il tipo ennupla. La seconda fase associa uno di questi tipi a ciascun elemento dei programmi attraverso una *relazione di tipo* (*typing relation*) che associa a ogni costrutto del linguaggio un tipo. Questa relazione è formalmente definita attraverso un insieme di regole di inferenza, definite per induzione sulla sintassi del linguaggio. Le regole hanno varie premesse e una conclusione, che è vera solo quando lo sono le premesse. Per esempio, se la variabile *x* è di tipo *privato* (premessa 1) anche l'espressione *x + 3* ha tipo *privato* (conclusione), anche se la costante *3* è *pubblica* (premessa 2); ciò rende l'assegnamento *x := x+3* ben tipato e la sua esecuzione non svelerà alcun segreto agli attaccanti a tempo di esecuzione. La terza fase consiste nel dimostrare che il sistema di tipi è corretto: un programma ben tipato non causerà errori di tipo a tempo di esecuzione (tecnicamente chiamato teorema di *soundness*).

Generalmente, dalla specifica logica si deriva in maniera diretta un algoritmo che realizza l'analisi.

3. Eseguire o comporre in modo sicuro codice non affidabile

Come abbiamo visto in precedenza, non è banale per uno sviluppatore scrivere codice sicuro. Ma cosa succede quando non ci possiamo neppure fidare dello sviluppatore? In molti casi infatti gli utilizzatori ricevono del software, magari scaricandolo dalla rete per eseguirlo o per incorporarlo in un nuovo programma. In questo scenario, chi esegue il software non sa nulla di chi lo ha prodotto se non, al massimo, la sua buona o cattiva reputazione. Sotto queste ipotesi, verificare che il software che andiamo a eseguire non rispetti certe proprietà è estremamente importante.

Nello scenario del **codice mobile**, uno sviluppatore non affidabile (produttore) pubblica il proprio software e un utente (consumatore) lo scarica e installa

oppure lo incorpora in un nuovo programma che sta scrivendo. Inutile dire che questa situazione è oggi la più diffusa nella nostra esperienza quotidiana. L'esempio più lampante è quello degli smartphone i cui attuali sistemi operativi permettono lo scambio di software (le app) tra due vaste platee di produttori e consumatori praticamente senza alcun rapporto diretto di fiducia. Cerchiamo di comprendere come il problema della sicurezza può essere affrontato in questo scenario.

3.1 App store: un esempio di intermediario affidabile

Come nella vita di tutti i giorni, un modo semplice per sopperire alla mancanza di fiducia tra due parti consiste nell'individuare un **terzo fidato** (trusted third party — TTP). La fiducia nel terzo fidato può derivare da molti fattori che descriveremo in seguito, come la *reputazione*, la *posizione* di rilievo e gli *strumenti* a disposizione. Nel caso del codice mobile, la TTP è quasi sempre un intermediario (broker) che raccoglie e distribuisce il software. Due esempi molto noti sono Google Play e Apple Store che offrono agli utenti prodotti di vario tipo (musica, libri, software, ecc.) per Android e iOS, rispettivamente. Anche se in passato ci sono stati casi clamorosi di app pericolose distribuite tramite questi servizi, milioni di utenti usano giornalmente i propri smartphone interagendo con questi servizi. Questo è possibile perché i broker di app sfruttano tutti e tre i fattori menzionati in precedenza e dettagliati qui sotto.

Reputazione. Per grandi compagnie come Apple e Google il danno di immagine derivante dalla distribuzione di software insicuro può essere enorme. Di conseguenza esiste una fortissima motivazione per mettere in campo le risorse necessarie per individuare e gestire prontamente questo rischio. In particolare, la rimozione delle app pericolose o sospette dagli store è un processo che da una parte riduce l'esposizione al rischio per il broker e dall'altra garantisce ai consumatori che difficilmente troveranno software che in passato si è rivelato dannoso.

Posizione. Google Play e Apple Store sono di fatto i canali principali per il trasferimento delle app e offrono agli utenti la possibilità di scambiarsi opinioni e informazioni sulle app tramite i cosiddetti feedback. Il risultato è un sistema di reputazione distribuito largamente accettato che favorisce la rapida individuazione e l'isolamento di app che appaiono sospette o pericolose ad alcuni utenti. Inoltre, quando una app viene identificata come potenzialmente dannosa, la sua rimozione dal catalogo evita la sua ulteriore diffusione.

Strumenti. I broker richiedono che ogni app rispetti alcune caratteristiche specifiche, tra cui la firma digitale delle app da parte degli sviluppatori in modo da garantire l'integrità del codice, ovvero che la app non può essere modificata illecitamente. Inoltre, il processo di pubblicazione di una app non è istantaneo. Tra quando lo sviluppatore carica il software e quando esso risulta effettivamente accessibile agli utenti può passare un intervallo di tempo non trascurabile. La ragione è che il software caricato viene sottoposto a test di sicurezza automatici allo scopo di identificare e disinnescare le eventuali minacce.

In molti contesti però gli intermediari pur essendo affidabili possono non offrire sufficienti garanzie. Per esempio, questo avviene quando il consumatore del software ha particolari requisiti di sicurezza che non vuole rendere pubblici o

che il broker non intende o non è in grado di controllare. In tutti questi casi, il consumatore può fare affidamento su varie tecniche di verifica dei requisiti, alcune delle quali sono discusse in seguito.

Riquadro 4: Struttura di un'applicazione Android

Le app Android vengono distribuite sotto forma di pacchetti compressi detti Application Packages (APK). Un APK contiene tutte le risorse necessarie alla app durante l'esecuzione. Esse includono in primo luogo il codice eseguibile che può avere vari formati. Comunemente si tratta di codice *bytecode*, simile a quello che si ottiene compilando dei sorgenti Java. Il bytecode viene combinato in un unico file detto *Dalvik Executable* (DEX)[10] che viene caricato dal sistema operativo e ulteriormente compilato per ottenere codice eseguibile. Insieme al file DEX, un APK contiene tutti i file di supporto (es. audio, video, immagini, database) e il file *manifest* che contiene una descrizione della app e dei suoi componenti software. In particolare, esso contiene una lista dei permessi di accesso necessari per l'esecuzione che l'utente deve garantire tramite un'autorizzazione esplicita. Infine, l'APK include le firme digitali di tutti i suoi elementi e quella dell'intero contenuto del pacchetto, allo scopo di prevenire modifiche durante il caricamento dell'APK.

3.2 Analisi dinamica

L'analisi dinamica consiste nell'eseguire il software sospetto e osservarne il comportamento. Per evitare i rischi derivanti dall'esecuzione del software, l'analisi dinamica avviene tipicamente in un contesto protetto detto **sandbox.** La sandbox è una copia fedele della piattaforma del consumatore, ma priva di dati e risorse che potrebbero essere danneggiati o compromessi. Inoltre, la sandbox è dotata di strumenti per monitorare il comportamento del software in esecuzione. Essa è in molti casi una macchina virtuale, cioè un programma che simula il comportamento di un computer o uno smartphone (per esempio si pensi a una macchina virtuale Windows in esecuzione su un sistema Linux). L'utilizzo di una sandbox può quindi prevenire la compromissione della piattaforma fisica. Essendo un programma, la macchina virtuale può essere utilmente ispezionata per verificare quali chiamate al sistema operativo vengono fatte dai singoli componenti software.

Vulnerability testing

In generale un software consiste di moltissime istruzioni la cui esecuzione è guidata dai dati in input e da altri fattori (ad esempio, dati ambientali come il l'ora del giorno o messaggi ricevuti dalla rete). All'interno di software sviluppato da altri si possono annidare vulnerabilità di molti tipi diversi (per un esempio si veda il riquadro 1) e questa informazione potrebbe essere nota al produttore, ma non al consumatore del programma.

Il consumatore ricorre allora ai *test di vulnerabilità* che consistono nell'eseguire il codice e valutare se questo si comporta in modo da esporre la vulnerabilità

[10] *Dalvik era il nome della virtual machine originariamente presente sui dispositivi Android e dismessa dalla versione 5 del sistema operativo. Attualmente è stata sostituita dall ART virtual machine, ma l'acronimo DEX è rimasto in uso.*

cercata (ad esempio scrivendo nella memoria in locazioni che non dovrebbero essere accessibili e quindi violando la memory safety). Per cercare di scoprire la presenza di vulnerabilità è però cruciale disporre di una serie di test adeguatamente rappresentativi del comportamento di un programma, adattando i classici modi per testare il software. Le tecniche di *code coverage* generano automaticamente questi test tramite delle euristiche. Per esempio, the American Fuzzy Lop[11] (AFL) genera ed esegue test in maniera automatica. Ogni esecuzione viene monitorata registrando quali istruzioni vengono eseguite e se queste mostrano comportamenti associati a vulnerabilità note. A ogni ciclo vengono generati nuovi test perturbando in modo casuale (fuzzing) quelli eseguiti in precedenza, ad esempio premiando nel tempo i test che visitano porzioni di codice non ancora scrutinate, massimizzando cosí l'efficacia dei test. Per quanto efficiente, il fuzzing incontra enormi difficoltà nell'esplorazione dei programmi. Il motivo è che la logica che regola i flussi di esecuzione è difficilmente prevedibile. Per esempio, si immagini un programma che effettua una certa operazione di interesse quando riceve in input una specifica sequenza di caratteri: lo spazio delle possibili sequenze è talmente vasto che un fuzzer non riuscirà praticamente mai a indovinare quella giusta. Per superare questo ostacolo si sono recentemente affermate le tecniche di *concolic* (*conc*rete + symb*olic*) *testing*, che combina l'efficienza dei fuzzer con la capacità dei metodi formali di trovare la soluzione esatta di un problema. Per esempio, D*riller* [16] sfrutta AFL per generare rapidamente molti test, ma quando questi non riescono a raggiungere nuove aree del codice, utilizza tecniche di dimostrazione automatica (per esempio un *SMT* solver) che sono meno efficienti, ma esatte, con cui trova un test che copre una nuova area. Il nuovo test serve come ponte per permettere ad AFL di ricominciare la propria esplorazione del codice.

Taint analysis

Una buona batteria di test di vulnerabilità può evidenziare alcuni difetti di un programma, ma può non scoprire se manipola in modo scorretto i dati. Un modo per fare ciò consiste nell'identificare quali sono le fonti dei flussi di informazione all'interno del programma e quali le loro destinazioni. Un software viene classificato non sicuro se da una fonte che contiene informazioni riservate c'è un flusso esplicito che raggiunge una destinazione in cui chiunque può leggere. Fonti e destinazioni possono essere file, database o indirizzi di rete. In alcuni contesti, certe fonti possono essere identificate come più o meno critiche in base al tipo di dati che esse contengono. Questo avviene comunemente nei nostri smartphone che hanno accesso alla rubrica dei nostri contatti, la lista delle chiamate e dei messaggi, la nostra posizione geografica, ecc. La taint analysis si occupa di identificare i flussi di informazione che hanno origine dalle sorgenti con un dato livello di criticità e che possono raggiungere destinazioni a un livello inferiore.[12]

---

[11] *Si veda* http://lcamtuf.coredump.cx/afl/

[12] *Questo tipo di analisi tuttavia non considera i flussi impliciti di informazione, per esempio quando un salto dipende da un valore riservato [27]*

Program instrumentation

Le tecniche descritte in precedenza permettono di testare la presenza di vulnerabilità in un software che viene eseguito all'interno di una sandbox. In alcuni casi però non è possibile eseguire tutto il software. Il motivo principale risiede nel fatto che molte applicazioni interagiscono con servizi remoti durante la loro esecuzione, che non possono essere ispezionati direttamente. Il *gray-box testing* (test a scatola grigia) evidenzia la presenza di software ispezionabile (*white-box*, scatola bianca) e non (*black-box*, scatola nera). La ripetizione dello stesso test può portare a risultati diversi a causa dello stato interno delle componenti remote e di conseguenza è più difficile individuare le vulnerabilità.

Questa situazione è tipica delle applicazioni web in cui l'interazione principale avviene tra il front-end (software mobile eseguito dal consumatore) e il back-end (software eseguito su una macchina remota, sotto il controllo del produttore). Uno dei principali fattori di complessità del testing di vulnerabilità è che i comportamenti di interesse dipendono da una piccola parte di codice immersa in un grandissimo numero di istruzioni. Inoltre, non è sufficiente isolare solo le istruzioni rilevanti (per esempio la creazione di una connessione verso l'indirizzo *x*) visto che il comportamento di tali istruzioni dipende anche da quelle precedenti (per esempio se *x* viene calcolato come la concatenazione di "http://" e *y*), oltre che dai dati scambiati tra il back-end e il front-end.

La *program instrumentation* è una tecnica che permette di interagire con l'esecuzione di un programma inserendo nuove istruzioni in alcuni punti specifici. Per esempio, una strategia di instrumentazione consiste nell'aggiungere istruzioni che stampano il valore dei parametri di certe chiamate di interesse come gli indirizzi di rete verso cui vengono stabilite connessioni. In questo modo è possibile focalizzarsi sull'interazione tra il back-end e il front-end e comprendere più dettagliatamente il funzionamento del software remoto e meglio individuare le possibili vulnerabilità. Attualmente la piattaforma per program instrumentation che sta riscuotendo maggiore successo è sicuramente *Frida.*[13] I motivi di tale successo sono il supporto multi-piattaforma (che permette di instrumentare sia applicazioni desktop che mobile) e la possibilità di definire i comportamenti instrumentati tramite linguaggi di alto livello come Javascript.

3.3 Analisi statica

L'analisi dinamica permette di ispezionare l'esecuzione di un software, ma ha una limitazione: per quanto possiamo testare un programma non potremo mai eseguirlo in tutti i modi possibili per ottenere la code coverage completa. Inoltre, ci sono contesti in cui è preferibile evitare del tutto l'esecuzione di un software, per esempio nel caso in cui sospettiamo si tratti di un malware, cioè un programma intenzionalmente malevolo come un virus.

Le tecniche di analisi statica mirano a comprendere il comportamento di un programma ispezionando il suo codice, senza eseguirlo. Per fare questo si ricorre a processi di approssimazione e astrazione del comportamento e dei dati reali. Ciò porta a trascurare alcuni dettagli causando una perdita di precisione

[13] *Si veda https://www.frida.re/*

nei risultati, con il rischio di introdurne di falsi risultati; tuttavia queste approssimazioni e astrazioni sono inevitabili a causa dell'impossibilità di risolvere il "problema della fermata".[14]

Le tecniche di analisi statica sono largamente impiegate nella verifica del software e sono state spesso adattate e usate con successo nel caso della sicurezza; di seguito ci soffermeremo su quelle più diffuse.

Model checking

Il *model checking* è una delle tecniche più usate per verificare se un sistema (anche hardware) ha una certa proprietà, espressa come una formula di una opportuna logica temporale. Con questo termine si identifica una vasta famiglia di algoritmi che, visitando un modello a stati finiti, verificano che in ogni suo stato la proprietà valga.

Il model checking è una tecnica esatta che può portare a due risultati, entrambi di grande importanza: (i) una dimostrazione di correttezza o (ii) un controesempio. La dimostrazione di correttezza garantisce che nessuna esecuzione del sistema potrà mai violare la specifica. Invece, il controesempio descrive una esecuzione del sistema che viola la specifica (dimostrando che esiste un rischio effettivo di violazione).

Esistono alcune limitazioni all'uso del model checking per l'analisi del software. La prima è il costo computazionale: il model checking richiede algoritmi molto costosi che non ammettono ottimizzazioni oltre alcuni limiti teorici. Di conseguenza, l'analisi di sistemi con molti stati diventa rapidamente intrattabile. La seconda limitazione riguarda la generazione dei modelli. Rappresentare un software tramite un modello fatto di un numero finito di stati e transizioni causa necessariamente delle approssimazioni legate, appunto, al problema della fermata. Nella migliore delle ipotesi un programma può avere moltissimi stati ed è quindi quasi sempre necessario procedere a una loro aggregazione tramite varie tecniche di astrazione. Il risultato però può essere quello di fondere stati che dovrebbero essere tenuti distinti e incorrere così in falsi positivi. Una terza limitazione nasce dal fatto che, molto spesso, non è nota la specifica del sistema da analizzare.

Nonostante le limitazioni menzionate in precedenza, il software model checking è una tecnica che ha portato a notevoli risultati e che è alla base di strumenti allo stato dell'arte come Java PathFinder.[15] Inoltre, alcuni lavori recenti hanno applicato il model checking nell'ambito del rilevamento di malware e virus informatici, per esempio il lavoro di Song and Touili [17]. Generalmente, i comuni sistemi di rilevamento di virus usano dei meccanismi basati su

---

[14] L'indecidibilità del problema della fermata consiste nella non esistenza di un algoritmo che in un numero finito di passi decida se un generico programma eseguito con un ingresso arbitrario si arresterà o ciclerà per sempre. Il problema di decidere se un programma soddisfa una proprietà, per esempio l'assenza di vulnerabilità, è strettamente collegato al problema della fermata e quindi è indecidibile.

[15] Si veda https://github.com/javapathfinder/jpf-core

signature.[16] Le principali limitazioni di questo approccio sono che è necessario mantenere un database di signature aggiornato, che la verifica dell'infezione può essere fatta solo a posteriori quando un malware è noto (così come la sua signature) e che una verifica di questo genere è facile da aggirare cambiando il codice del malware, ad esempio usando tecniche di offuscamento. Un modo di superare queste limitazioni è quella di fare un'analisi comportamentale senza eseguire il programma sospettato di essere infetto. Per fare ciò si genera un modello a stati finiti del programma sospetto e si caratterizza il comportamento malevolo come una formula in un'opportuna logica temporale: il programma sarà considerato dannoso se la formula vale per il programma. Più recentemente, un approccio simile è quello di Talos [18] che si specializza nel rilevamento di ransomware su sistemi Android usando model checking.

Riquadro 5: Semantiche non standard

La semantica di un linguaggio di programmazione definisce l'effetto delle istruzioni sullo stato di un programma. Esistono diversi modi per definire formalmente la semantica di un linguaggio come, per esempio, la semantica operazionale small-step, in cui ogni operazione corrisponde a una transizione da uno stato sorgente a uno stato destinazione, o la semantica denotazionale, in cui ogni programma corrisponde a una funzione matematica che trasforma lo stato iniziale nello stato finale.

Le semantiche non standard differiscono perché descrivono il comportamento dei programmi a diversi livelli di astrazione, sia sul controllo che sui dati. Per esempio, i cicli vengono percorsi una sola volta, oppure i dati vengono raggruppati in classi di equivalenza, per esempio, invece che avere tutti i naturali si ha il valore astratto ***0*** che identifica il numero zero e ***N*** che rappresenta tutti gli altri numeri. Le astrazioni devono essere corrette, cioè il comportamento descritto dalla semantica non standard è una sovra-approssimazione di quello della semantica standard.

Un ulteriore esempio è la semantica simbolica che sostituisce le classiche operazioni aritmetiche con operazioni che compongono espressioni simboliche. Per comprendere meglio si consideri il seguente frammento di codice C:

```
unsigned int foo(unsigned int x, unsigned int y) {
            if (x > 51)
                  y = y/(x+1);
            else
                  y = y/(x+2);
            return y;
      }
```

Se applichiamo le regole della semantica standard del C alla chiamata foo(0,10) otteniamo tre passaggi: (i) visto che 0 > 51 è falso il programma salta al ramo

[16] *Si può pensare alla signature come delle sequenze genetiche di identificano univocamente un malware, però queste sequenze sono calcolate solo sulla sequenza di istruzioni del programma (sintassi) non tenendo conto del comportamento che implementano (semantica).*

"else"; (ii) viene calcolato il nuovo valore di y come 10/(0+2) = 5; e (iii) viene restituito 5. Se consideriamo i valori astratti ***0*** e ***N*** e valutiamo la chiamata foo(***0***,***N***) al passo (i) il confronto diventa ***0*** > ***N*** (che risulta comunque ancora falso); al passo (ii) il nuovo valore calcolato per y è ***N/(0+N)*** = ***N/N*** = ***N*** ottenuto con le operazioni astratte che sono la controparte delle normali operazioni di divisione e somma; infine al passo (iii) viene restituito ***N***.

Abstract interpretation

Questa tecnica si basa soprattutto su un'accurata astrazione dei dati che vengono raggruppati in valori astratti, in accordo con specifiche proprietà di interesse per i controlli che si intendono fare. Di fatto il linguaggio di programmazione è dotato di una semantica astratta (v. riquadro 5) che sia allo stesso tempo semplice da trattare e adeguata a evidenziare certe vulnerabilità.

L'abstract interpretation è la tecnica alla base di strumenti commerciali come ad esempio Julia[17] che effettua una serie di analisi su Java bytecode. Lo strumento prevede delle analisi per scoprire vulnerabilità che aprono la strada ad attacchi di code injection nelle applicazioni web [19]. Ciò viene fatto mediante una taint analysis eseguita a tempo statico e tuttavia molto precisa. Tra le varie analisi c'è anche l'individuazione di potenziali `NullPointerException`, un errore che un programma Java genera quando si tenta di utilizzare un puntatore nullo, cioè non correttamente inizializzato. In questo caso specifico, il dominio astratto considerato consiste di soli due elementi: il puntatore nullo e qualsiasi altro puntatore. La semantica astratta analizza il flusso di esecuzione del programma, individuando le operazioni sui puntatori nulli, trascurando gli effetti delle operazioni su quelli correttamente inizializzati.

Symbolic execution

La *symbolic execution* è una tecnica di analisi basata su una semantica non standard (v. riquadro 5) che opera su simboli e su espressioni simboliche. La valutazione simbolica di un programma rappresenta tutte le valutazioni possibili quando ai simboli siano stati assegnati dei valori concreti.

La principale limitazione associata alla symbolic execution è legata agli enormi costi computazionali. Da una parte, la dimensione delle espressioni generate cresce moltissimo anche in pochi passi. Inoltre, la risoluzione di tali espressioni, necessaria per valutare se una certa esecuzione è possibile o meno, è un problema complesso (NP-hard) e quindi anch'esso molto costoso, anche con espressioni simboliche non troppo grandi.

Nonostante questo, gli strumenti per l'esecuzione simbolica, per esempio Mayhem[18], angr[19] e Triton[20], stanno ricevendo molta attenzione e si sono

---

[17] *Si veda https://juliasoft.com/abstract-interpretation/*

[18] *Si veda https://users.ece.cmu.edu/~dbrumley/pdf/Cha%20et%20al._2012_Unleashing%20Mayhem%20on%20Binary%20Code.pdf*

[19] *Si veda https://angr.io/*

[20] *veda https://triton.quarkslab.com/*

dimostrati molto efficaci per individuare possibili vulnerabilità del software. Una dimostrazione del loro potenziale è avvenuta alla Cyber Grand Challenge[21] organizzata dal DARPA nel 2016, che è stata una competizione per strumenti di attacco e difesa completamente automatici in grado di individuare, sfruttare e correggere le vulnerabilità presenti nel sistema di gioco. In quella circostanza Mayhem e angr si sono classificati rispettivamente primo e terzo.

Il motivo principale è che questi strumenti sostengono il lavoro di un analista e traggono vantaggio dalle sue intuizioni. In particolare, invece di esplorare tutto il codice di un programma, questi strumenti implementano meccanismi per il debugging simbolico in cui si procede solamente pochi passi alla volta. In questo modo, l'analista riceve una precisa descrizione di come l'esecuzione del programma si può svolgere e tiene sotto controllo la dimensione delle formule. Allo stesso tempo, il numero di percorsi da esplorare si riduce nel momento in cui alcuni di essi, che l'analista riconosce come non interessanti, vengono scartati. Il risultato è una interazione uomo-macchina che potenzia le capacità dell'analista umano e la sua comprensione dei possibili sviluppi nel comportamento del software.

A conferma della sua efficacia, l'esecuzione simbolica ha recentemente sostenuto lo sviluppo di Oyente [20] un tool per l'analisi di sicurezza di smart contract per Ethereum, una infrastruttura computazionale decentralizzata a livello globale basata sulla tecnologia blockchain. Gli smart contract sono programmi che incorporano funzionalità finanziarie o asset economici, eseguiti da Ethereum. Essi sono stati vittima di diversi attacchi [21] con cui milioni di dollari sono stati rubati o congelati. Oyente verifica l'assenza delle vulnerabilità più note prima che lo smart contract venga messo in esercizio sulla blockchain.

4. Conclusioni

La sicurezza del software è uno dei pilastri della sicurezza informatica. In questo contesto, le tecniche di analisi del codice sono fondamentali per individuare tempestivamente le vulnerabilità e prevenire gli attacchi. Abbiamo brevemente esaminato i principi che i linguaggi di programmazione soddisfano o dovrebbero soddisfare per sostenere il progetto e la scrittura di programmi sicuri. Inoltre, abbiamo presentato alcune tecniche e strumenti che assistono un utente di componenti software nell'individuazione, per quanto possibile, di vulnerabilità di tali componenti.

Nonostante il grande volume di lavoro svolto in questo campo, molto rimane da studiare per incrementare la sicurezza e l'affidabilità del software che usiamo, soprattutto perché c'è una continua gara tra chi tenta di attaccare i nostri sistemi e chi li difende. Una delle ragioni per cui questa competizione è così difficile è che gli attaccanti non rispettano alcuna regola e operano a diversi livelli di astrazione, dal programma ad alto livello fino alla architettura fisica. A questo proposito, una tra le linee di ricerca più recenti riguarda la compilazione sicura [22] [23] [24] [25] [26] ovvero lo studio di come garantire che i vari passi di compilazione e i vari passaggi dal linguaggio sorgente a quello macchina, non

[21] *Si veda https://www.darpa.mil/program/cyber-grand-challenge*

indeboliscono le garanzie di sicurezza date ad alto livello. Altre linee riguardano l'uso di tecniche quali quelle basate sul machine learning, o sulla valutazione empirica o statistica dei rischi (risk assessment).

Bibliografia

[1] D. Kushner, «The Real Story of Stuxnet,» *IEEE Spectrum,* vol. 50, n. 3, pp. 48-53, 2013.

[2] D. Kozen, «Language-based security,» in International Symposium on Mathematical Foundations of Computer Science, Szklarska Poreba, 1999.

[3] E. H. Spafford, «The Internet Worm Program: An Analysis,» 29 Novembre 1988. [Online]. Available: https://spaf.cerias.purdue.edu/tech-reps/823.pdf. [Consultato il giorno 12 Novembre 2019].

[4] M. Abadi, M. Budiu, Ú. Erlingsson e J. Ligatti, «Control-flow Integrity Principles, Implementations, and Applications,» *ACM Trans. Inf. Syst. Secur.,* vol. 13, n. 1, pp. 1-40, 2009.

[5] E. Felten e G. McGraw, Securing Java, Wiley, 1999.

[6] M. Hicks, «What is memory safety?,» 21 Luglio 2014. [Online]. Available: http://www.pl-enthusiast.net/2014/07/21/memory-safety/. [Consultato il giorno 12 Novembre 2019].

[7] B. C. Pierce, Advanced topics in types and programming languages, MIT press, 2005.

[8] A. Gaynor, «Modern C++ Won't Save Us,» 21 Aprile 2019. [Online]. Available: https://alexgaynor.net/2019/apr/21/modern-c++-wont-save-us/. [Consultato il giorno 12 Novembre 2019].

[9] T. Jim, G. Morrisett, D. Grossman, M. Hicks, J. Cheney e Y. Wang, «Cyclone: a safe dialect of C,» in *USENIX Annual Technical Conference, General Track*, 2002.

[10] G. C. Necula, J. Condit, M. Harren, S. McPeak e W. Weimer, «CCured: type-safe retrofitting of legacy software,» *ACM Trans. Program. Lang. Syst.,* vol. 27, n. 3, pp. 477-526, 2005.

[11] A. Ruef, L. Lampropoulos, I. Sweet, D. Tarditi e M. Hicks, «Achieving safety incrementality with CheckedC,» in *Principles of Secuirty and Trust (POST)*, 2019.

[12] S. Nagarakatte, J. Zhao, M. M. Martin e S. Zdancewic, «SoftBound: Highly Compatible and Complete Spatial Memory Safety for C,» *SIGPLAN Not.,* vol. 44, n. 6, pp. 245-258, 2009.

[13] S. Nagarakatte, J. Zhao, M. M. Martin e S. Zdancewic, «CETS: Compiler Enforced Temporal Safety for C,» *SIGPLAN Not.,* vol. 45, n. 8, pp. 31-40, 2010.

[14] D. Volpano, C. Irvine e G. Smith, «A Sound Type System for Secure Flow Analysis,» *Journal of Computer Security,* vol. 4, n. 2-3, pp. 167-187, 1996.

[15] A. C. Myers, A. Sabelfeld e S. Zdancewic, «Enforcing robust declassification and qualified robustness,» *Journal of Computer Security,* vol. 14, n. 2, pp. 157-196, 2006.

[16] N. Stephens, J. Grosen, C. Salls, A. Dutcher, R. Wang, J. Corbetta, Y. Shoshitaishvili, C. Kruegel e G. Vigna, «Driller: Augmenting Fuzzing Through Selective Symbolic Execution,» in *NDSS Symposium*, 2016.

[17] F. Song e T. Touili, «Pushdown Model Checking for Malware Detection,» in *Tools and Algorithms for the Construction and Analysis of Systems (TACAS)*, 2012.

[18] A. Cimitile, F. Mercaldo, V. Nardone, A. Santone e C. A. Visaggio, «Talos: no more ransomware victims with formal methods,» *International Journal of Information Security,* vol. 17, n. 6, pp. 719-738, 2018.

[19] F. Spoto, E. Burato, M. D. Ernst, P. Ferrara, A. Lovato, D. Macedonio e C. Spiridon, «Static Identification of Injection Attacks in Java,» *ACM Trans. Program. Lang. Syst.,* vol. 41, n. 3, pp. 1-58, 2019.

[20] L. Luu, D.-H. Chu, H. Olickel, P. Saxena e A. Hobor, «Making Smart Contracts Smarter,» in *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security*, 2016.

[21] N. Atzei, M. Bartoletti e T. Cimoli, «A Survey of Attacks on Ethereum Smart Contracts SoK,» in *Proceedings of the 6th International Conference on Principles of Security and Trust*, 2017.

[22] M. Abadi, «Protection in Programming-language Translations,» in *Secure Internet Programming*, Springer-Verlag, 1999, pp. 19-34.

[23] P. Agten, R. Strackx, B. Jacobs e F. Piessens, «Secure Compilation to Modern Processors,» in *25th IEEE Computer Security Foundations Symposium*, 2012.

[24] M. Patrignani e D. Garg, «Secure Compilation and Hyperproperty Preservation,» in *30th IEEE Computer Security Foundations Symposium*, 2017.

[25] C. Abate, R. Blanco, D. Garg, C. Hritcu, M. Patrignani e J. Thibault, «Journey Beyond Full Abstraction: Exploring Robust Property Preservation,» in *32nd IEEE Computer Security Foundations Symposium*, 2019.

[26] M. Busi e G. Letterio, «A Brief Tour of Formally Secure Compilation,» in *Proceedings of the Third Italian Conference on Cyber Security*, 2018.

[27] M. Balliu, D. Schoepe e A. Sabelfeld, «We Are Family: Relating Information-Flow Trackers,» in *22nd European Symposium on Research in Computer Security*, 2017.

Biografie

Gabriele Costa è Assistant Professor presso la Scuola Alti Studi IMT di Lucca. In precedenza è stato ricercatore presso l'Istituto di Informatica e Telematica del CNR di Pisa e presso il Dipartimento di Informatica, Bioingegneria, Robotica e Ingegneria dei Sistemi dell'Università di Genova. La sua ricerca si concentra su vari aspetti della cybersecurity tra cui la verifica formale dei sistemi, le tecniche avanzate per VAPT e le metodologie per il training e la promozione della cyber hygiene.

Pierpaolo Degano è professore ordinario presso il Dipartimento di Informatica dell'Università di Pisa; è stato direttore del Dipartimento e del dottorato in Informatica; ha coordinato l'associazione nazionale dei professori e ricercatori in Informatica (GRIN) e il consiglio dei presidenti di dottorato in Informatica. È stato uno dei fondatori dell'IFIP TC1 WG 1.7 on Theoretical Foundations of Security Analysis & Design, di cui è stato presidente, e della conferenza POST; è stato il responsabile scientifico di molte conferenze; in altre ha tenuto relazioni invitate; è membro del comitato editoriale di alcune riviste scientifiche internazionali e nazionali. La sua area di ricerca include i linguaggi di programmazione, con particolare attenzione alla loro semantica e agli aspetti relativi alla sicurezza.

Letterio Galletta è ricercatore a tempo determinato presso la Scuola Alti Studi IMT di Lucca. In precedenza, è stato assegnista di ricerca presso il Dipartimento di Informatica dell'Università di Pisa. La sua ricerca si concentra sulla progettazione e l'utilizzo di tecniche tipiche dei linguaggi di programmazione e dei compilatori per affrontare problemi di sicurezza nel software.

Macchine senza guidatore: considerazioni morali

Maurizio Balistreri

Sommario

I veicoli autonomi sono presentati come il futuro del mondo dei trasporti. Lo sviluppo di macchine intelligenti e senza guidatore pone importanti questioni morali. Queste macchine, ad esempio, potranno avere un giorno rilevanza morale ed avere anche diritti? Inoltre, una macchina capace di risolvere dilemmi può essere considerata un 'agente' morale? Infine, queste macchine dovranno essere programmate ad agire moralmente: ma a quali principi morali ed a quali valori vogliamo che esse facciano riferimento?

Abstract

Autonomous vehicles are presented as the future of the world of transport. Their greatest advantage would be the reduction in deaths and injuries on the road due to accidents. The development of driverless vehicles raises important moral questions. Firstly, we are going to ask which duties we have towards intelligent machines and whether an autonomous driverless car may be considered a moral 'agent'. Further, the question must be faced of how to programme a machine to act morally and of which morality to take as a planning model to allow the autonomous car to respond to our expectations.

Keywords: self-driving car, trolley problem, ethics, autonomous robots, algorithms, Artificial Intelligence

1. Introduzione: perché sviluppare macchine senza guidatore?

I veicoli autonomi vengono presentati come il futuro del mondo dei trasporti [1,2,3]. Quando avremo le prime macchine senza guidatore? Alcuni prevedono prima del 2020 [3,4,5,6], altri invece ritengono che ci vorranno non meno di altri due o tre decenni [7,8]. Si pensa poi che ai veicoli autonomi si arriverà passando per una fase intermedia, in cui i compiti del guidatore saranno piano piano delegati interamente ai robot intelligenti. Via via che le macchine diventeranno sempre più autonome, probabilmente il guidatore perderà progressivamente la dimestichezza con la guida e la capacità di affrontare situazioni impreviste, in quanto avrà sempre meno opportunità di esercitarsi [9]. Inoltre, i veicoli intelligenti ma soltanto parzialmente autonomi potrebbero risultare poco graditi o soddisfacenti per il consumatore, in quanto lo obbligano, negli viaggi in macchina, ad essere sempre attento alla strada. Per questa ragione, sarebbe meglio affidare la guida completamente al robot: ma almeno per il momento e per il prossimo futuro questa possibilità è da escludere, in quanto le macchine non hanno ancora la stessa prontezza di un essere umano nel percepire eventuali ostacoli, evitarli ed affrontare con dimestichezza nuove possibili circostanze. Per diventare competitive ed affermarsi, comunque, le macchine autonome non dovranno essere in grado di evitare qualsiasi danno a cose e/o persone [9]: basta che esse assicurino un numero di incidenti e di morti più basso di quello attuale. «Se i veicoli senza guidatore producessero meno incidenti sulla strada dei guidatori umani – ricorda l'OMS – questo darebbe una ragione forte di politica pubblica per muovere il prima possibile ad un traffico su strada interamente lasciato alle macchine autonome. Il costo umano del pedaggio stradale è tremendo. Il costo sociale ed economico del prendersi cura a breve e lungo termine delle vittime di incidenti stradali produce un consumo significativo delle risorse pubbliche» [10].

Con lo sviluppo di macchine autonome sempre più sicure la guida di una macchina potrebbe essere vietata oppure assicurare una macchina che può essere guidata potrebbe diventare così costoso che quasi nessuno potrebbe permettersi di acquistarla. In altri termini, come afferma Robert Sparrow «Se i veicoli senza conducenti non diventeranno sicuri, non sarà morale venderli. Una volta che saranno più sicuri dei conducenti umani, per quanto riguarda i rischi a terzi, allora dovrebbe essere illegale guidarle: a quel punto eventuali guidatori sarebbero simili a robot ubriachi» [9]. Per altro, nel caso in cui le macchine con guidatore causassero incidenti, anche le case di produzione potrebbero essere chiamate a risponderne, in quanto producono e mettono in vendita cose che possono arrecare un danno alle persone: «una parte offesa o i suoi familiari, o forse lo stesso governo potrebbe denunciare per la produzione di un prodotto non sicuro la società che ha venduto veicoli senza capacità di guida autonoma e/o quelle che permettono l'opzione della guida umano» [9].

Le macchine completamente autonome possono essere utili a qualsiasi persona: non soltanto ci farebbero risparmiare tantissimo tempo negli spostamenti, ma ci permetterebbero di impiegare il tempo del viaggio in attività

più 'produttive' della guida. Inoltre, esse ci permetterebbero di muoverci in condizioni di maggiore sicurezza. Mentre siamo al volante può capitare facilmente di distrarsi: ad esempio, possiamo distogliere lo sguardo dalla strada per rispondere ad una telefonata, per scrivere un messaggio, per cambiare la stazione della radio, per riallacciare la cintura di sicurezza, per osservare una pubblicità o buttare un occhio su un articolo di giornale. Sono piccole distrazioni che però sono pericolose sia per noi che per le altre persone. Le macchine autonome non avrebbero questi problemi, perché sarebbero programmate a monitorare la situazione del traffico e la strada per tutta la durata del viaggio: quello che accade all'interno della macchina o lungo la strada e che non riguarda la sicurezza non potrebbe catturare la loro attenzione – nemmeno per un secondo. Inoltre, le macchine autonome potrebbero risultare particolarmente vantaggiose a quelle persone affette da gravi disabilità o altri problemi e che oggi non guidano. Una persona non vedente, ad esempio, oggi può spostarsi per la città in macchina solamente se chiama un taxi o viene accompagnata da un adulto con la patente. E lo stesso vale per le persone anziane che non si muovono con facilità o con prontezza: anche loro non potrebbero mai mettersi al volante di una macchina o se lo facessero potrebbero mettere a rischio non soltanto la loro vita ma anche quella degli altri. Le macchine autonome non permetterebbero a queste persone di guidare, ma darebbero loro la stessa libertà di spostarsi per la città che oggi hanno le persone normodotate. È evidente, poi, che ci sarebbero vantaggi anche per le persone più giovani: esse non avrebbero più bisogno di chiedere un passaggio a mamma o papà, in quanto potrebbero farsi accompagnare a scuola o a casa dei loro amici dalla loro macchina (questa poi potrebbe ritornare a casa da sola per essere a disposizione degli altri membri della famiglia oppure andare a prendere le altre persone che l'hanno prenotata).

Le città cambierebbero perché non avremmo più bisogno di parcheggiare 'sotto casa': i veicoli potrebbero essere lasciati in qualsiasi luogo perché essi non avrebbero bisogno della nostra assistenza per trovare un'area di sosta o un parcheggio a lunga sosta. I parcheggi, inoltre, non dovrebbero trovarsi necessariamente nelle aree residenziali, in quanto quando ci serve una macchina potremmo probabilmente prenotarla. Questo significa che con il tempo potrebbero liberarsi importanti spazi abitativi: «Le torri per uffici, i cinema e i centri commerciali potrebbero eliminare i parcheggi che attualmente li circondano e li isolano dalla comunità circostante e potrebbero dotarsi invece di una piccola piazza o parcheggio dove le persone potrebbero essere lasciate e passate a prendere dai loro veicoli. Tutti questi vantaggi si realizzerebbero in misura ancora maggiore se, come abbiamo suggerito, è probabile che l'avvento di veicoli senza conducente ridurrà il numero totale di veicoli sulle strade» [9]. A livello stradale, inoltre, ci potrebbe essere un impatto positivo sul traffico, in quanto ci sarebbe un maggiore coordinamento tra le diverse macchine sia nei movimenti che per quanto riguarda la scelta del migliore percorso per raggiungere una destinazione: i veicoli, poi, potrebbero ridurre la distanza di sicurezza tra una macchina e l'altra, perché tutte le macchine saranno in grado di frenare o accelerare contemporaneamente [4,12]. Inoltre, per ridurre ulteriormente il traffico e favorire gli spostamenti, si potrebbe promuovere una

maggiore integrazione tra mezzi di trasporto diversi: «Combinati in questo modo, - afferma Luis Ferreras - i treni e i veicoli autonomi potrebbero offrire un servizio di trasporto pubblico 'porta a porta', superando il problema dell'ultimo miglio che attualmente tormenta i sistemi di trasporto pubblico» [13]. Infine, le strade sarebbero più belle non soltanto perché meno congestionate, ma anche perché non ci sarebbe più bisogno di segnali stradali, semafori e altre indicazioni stradali che molto spesso impediscono di guardare al paesaggio circostante. Quando necessario, le auto regoleranno autonomamente le loro velocità [14,15].

Lo sviluppo delle macchine senza guidatore ci interessa perché solleva importanti questioni morali che saranno al centro della nostra analisi nelle prossime pagine. Inizieremo a domandarci se automobili tecnologicamente avanzate, in grado non soltanto di riconoscere e risolvere eventuali dilemmi ma anche di interagire con l'utente umano possono avere soltanto un valore strumentale oppure anche valore morale. Se, in altri termini, saranno soltanto delle cose o avranno anche una loro dignità, per cui non potremo trattarle come vogliamo ma dovremo riconoscere loro diritti. Ci chiederemo, poi, se una macchina intelligente va considerata un agente morale, cioè un soggetto che possiamo considerare responsabile per quello che fa, perché capace non soltanto di distinguere ma anche di scegliere tra diversi corsi di azione: la risposta sembra semplice, ma come facciamo a giustificare le nostre conclusioni? È vero che una macchina intelligente (automobile o qualsiasi robot) agisce sulla base della programmazione che ha ricevuto: siamo sicuri però che noi possiamo controllare e scegliere le nostre azioni? Infine, c'è la questione che riguarda la morale a cui la macchina deve fare riferimento: lungo la strada che dovrà percorre per trasportare i passeggeri a destinazione, l'automobile autonoma potrebbe trovarsi di fronte a importanti dilemmi morali. È importante che agisca bene, ma spetta a noi decidere che cosa vogliamo che faccia.

2. Una macchina senza guidatore può avere rilevanza morale?

Lo sviluppo di macchine senza guidatore pone importanti questioni di rilevanza morale. Innanzi tutto, è legittimo domandarsi quali doveri abbiamo verso queste macchine. Noi abbiamo una responsabilità anche nei confronti delle macchine non perché con i nostri comportamenti possiamo promuovere o compromettere il loro benessere, ma perché il loro cattivo funzionamento può danneggiarci, anche seriamente. Che l'autobus oppure l'autoveicolo vadano a fuoco non ci interessa più di tanto – a prescindere dal costo, in fondo si tratta di macchine e possiamo sempre sostituirle –, ma l'esplosione di una autovettura può ferire oppure uccidere persone innocenti. Per questa ragione, siamo tenuti a controllare le nostre macchine periodicamente (sia l'efficienza della performance che i livelli di emissioni inquinanti e rumorose), in quanto dobbiamo ridurre il rischio che provochino danni o incidenti evitabili. Ad ogni modo, le macchine in quanto tali hanno un valore semplicemente strumentale: mancano, cioè, di un valore intrinseco, indipendente dalla loro utilità. Tuttavia, è legittimo chiedersi se in futuro con lo sviluppo della robotica e dell'intelligenza artificiale

esse saranno ancora cose che possiamo usare come vogliamo. Negli ultimi decenni il nostro rapporto con gli animali è cambiato profondamente ed oggi sempre più persone ritengono sbagliato arrecare loro sofferenze. Quello che è successo negli ultimi anni con gli animali potrebbe succedere domani con robot intelligenti in grado di interagire con noi come se fossero autocoscienti? Noi possiamo sapere di avere davanti soltanto delle macchine tecnologicamente avanzate, programmate per interagire con noi, simulando il nostro comportamento, ma possiamo attribuire loro una vita mentale e le nostre stesse emozioni. Questo è un fenomeno che osserviamo già oggi con le tecnologie come ad esempio quelle impiegate in medicina, che non possiedono ancora elevati livelli di intelligenza: cosa accadrà quando con lo sviluppo scientifico e tecnologico ci troveremo ad interagire e a confrontarci con macchine sempre più intelligenti e (almeno apparentemente) capaci di provare le nostre stesse emozioni e sentimenti? Come potremo costruire relazioni con loro (e raccontare loro, ad esempio, i nostri pensieri e quello che proviamo), ma considerarle soltanto dei semplici oggetti [16]? Le macchine non sono persone, ma forse potremmo immaginare che lo siano.

Non stiamo ancora immaginando che un giorno le macchine diventeranno autocoscienti e che di conseguenza saranno in grado di provare i nostri stessi sentimenti. Quello che intendiamo affermare è che anche se le macchine che avremo davanti non proveranno 'veramente' i sentimenti e le emozioni che esprimeranno, questo potrebbe comunque non impedirci di considerarle l'altro di una relazione autentica [17]. È vero che una cosa sono i cosiddetti 'robot sociali' – cioè i robot autonomi o semi-autonomi in grado di interagire in maniera simile agli esseri umani, seguendo regole sociali o comportamenti legati al loro ruolo specifico – un'altra cosa le macchine che utilizziamo come mezzo di trasporto per muoverci da una località all'altra. Tuttavia, anche il software delle auto senza guidatore potrebbe essere in grado di conversare con l'utente, rispondere alle sue domande ed essere sensibile al suo stato d'animo. Se ipotizziamo, cioè, che anche le macchine senza guidatore saranno programmate per interagire come qualsiasi assistente, anche attraverso l'espressione di emozioni, per qualsiasi utente potrebbe essere facile empatizzare con il guidatore virtuale.

Avere, poi, una relazione con la macchina autonomia diventa ancora più facile, se immaginiamo che, a livello di sofware, la guida possa essere controllata dai dispositivi che impieghiamo quotidianamente a casa oppure a lavoro come assistenti vocali, in quanto avremmo già costruito una relazione con il nostro interlocutore virtuale. Qualcuno potrebbe essere scettico riguardo alla possibilità di attribuire una personalità ad un interlocutore che è presente nella nostra vita soltanto con la sua voce, ma questi dispositivi che oggi sono trasportabili, domani potrebbero essere integrati a casa oppure a lavoro con macchine in grado di muoversi ed interagire con noi come robot. Naturalmente, più tratteremo le macchine come persone e degne del nostro rispetto morale e più si apriranno dilemmi che al momento sembrano scenari di fantascienza. In caso di pericolo, ad esempio, chi dovremmo salvare: un umano o una macchina? Qualcuno potrebbe essere tentato di salvare la macchina con la quale ha costruito una relazione e che gli è stato vicino nei momenti più importanti della

propria vita. Per questa ragione, alcuni ritengono che sarebbe meglio che i robot non assomigliassero agli esseri umani perché i proprietari o, comunque, gli utenti dovrebbero aver sempre chiaro che essi non hanno alcun obbligo nei loro confronti [18].

3. Una macchina senza guidatore è un agente morale?

Inoltre, le macchine autonome ovverosia senza guidatore interessano la moralità perché la loro progettazione non può prescindere dalla progettazione di una 'competenza' etica. Queste macchine, infatti, non soltanto dovranno essere affidabili in termini di sicurezza e contribuire poi ad una riduzione degli incidenti su strada, ma dovranno anche essere in grado di risolvere i dilemmi morali che di volta in volta si presenteranno. Qui la questione che innanzi tutto si pone è se una macchina intelligente, progettata per comportarsi bene, possa essere considerata un 'agente' morale. Se assumiamo che la morale abbia a che fare con la ragione la risposta è scontata. Una macchina senza guidatore non può essere un agente morale perché comportarsi moralmente significa non soltanto distinguere tra comportamenti appropriati e non appropriati, ma anche essere capace di lasciarsi governare dalla ragione. Per questa ragione, afferma Kant, le regole della moralità valgono universalmente e si applicano a qualsiasi individuo in grado di considerare le cose razionalmente [19]: la ragione, infatti, prescrive le stesse regole o principi a qualsiasi individuo [20]. Forse un giorno anche le macchine più intelligenti mostreranno la capacità di agire secondo quella massima che può valere come massima di una legislazione universale, oppure, in un'altra formulazione, come legge 'universale' della natura. Tuttavia, fino a quando non saranno in grado di determinare da sole la loro volontà (ovverosia, non saranno capaci di darsi da sé la 'legge' o il principio della condotta) le loro scelte non saranno morali, in quanto saranno il risultato della nostra progettazione. Noi possiamo essere consapevoli della legge morale perché abbiamo una ragione che genera essa stessa le leggi o principi a cui volontariamente ci sottoponiamo: una macchina, invece, può anche essere in grado di imitare il nostro comportamento, ma manca della ragione e di conseguenza non può volere agire moralmente. Al massimo, una macchina potrebbe ingannarci con comportamenti, apparentemente morali, ma una macchina del genere non potrebbe essere un 'agente' morale. È vero che non possiamo escludere che domani riusciremo a produrre macchine così tecnologicamente avanzate da essere consapevoli di sé – cioè, autocoscienti [21]: almeno per il momento, però, si tratta di un'ipotesi remota e quasi fantascientifica.

Tuttavia, le cose cambiano se assumiamo che la ragione abbia un ruolo estremamente limitato nell'ambito della moralità e che le ragioni motivanti appartengano a sentimenti ed a passioni che sfuggono a qualsiasi controllo della ragione. Questa posizione è riassunta nel celebre passaggio del *Trattato sulla natura umana*, in cui Hume afferma che la ragione è – ed è giusto che sia – schiava delle passioni [22]. In questo caso, il fatto che la macchina manchi dell'autonomia della volontà, nel senso che non sia in grado di assoggettarsi ad una legislazione universale, non sarebbe più un motivo sufficiente per non

considerarla un agente morale, in quanto nessuno di noi ha una volontà che si lascia guidare dalla propria ragione. Per altro, è vero che l'agire della macchina sarebbe programmato da terze persone, ma anche il nostro comportamento è il risultato di processi educativi attraverso i quali impariamo ad apprezzare alcuni comportamenti ed a disprezzarne altri. Del resto, «(n)on è quindi una differenza da poco – afferma Aristotele –, se fin dalla nascita veniamo abituati in un modo piuttosto che in un altro, è importantissima, anzi è tutto» [23]. L'educazione che riceviamo, infatti, quando siamo ancora molto piccoli e non siamo ancora in grado di riflettere, crea abitudini e ci indirizza nella direzione giusta: «si deve essere abituati in un certo modo fin da giovani, come dice Platone, in modo da godere e rammaricarsi per le cose per cui è doveroso provare questi sentimenti…» [23]. In teoria, abbiamo sempre la possibilità di cambiare il nostro carattere, ma si tratta di una possibilità limitata perché «anche per l'ingiusto o l'intemperante vi era all'inizio la possibilità di non diventare persone di quel tipo – e per questo lo sono volontariamente – ma quando lo sono diventati non hanno più la possibilità di non esserlo» [23]. È vero, infine, che un agente morale può essere influenzato dai giudizio di approvazione o disapprovazione dei propri simili: anzi, si potrebbe affermare che la moralità ha proprio la funzione di incoraggiare e scoraggiare a compiere determinate azioni. Noi ci sforziamo ogni giorno, afferma Adam Smith, per fare in modo che il nostro comportamento sia in linea con i giudizi e i sentimenti di quello spettatore imparziale che vive dentro di noi ma che non rappresenta altro che la nostra natura sociale. Tuttavia, anche una macchina potrebbe essere sensibile ai giudizi delle altre persone: dovremmo solamente programmarla ad impostare il suo comportamento sulla base delle risposte (approvazione e/o disapprovazione) che riceve. Forse una macchina non sarà mai in grado di vergognarsi o essere orgogliosa per quello che ha fatto, ma potrebbe agire come se provasse questi sentimenti.

4. Come programmiamo una macchina ad agire moralmente?

Resta aperta, comunque, la questione di come programmare moralmente una macchina: cioè, quale concezione della moralità e quali valori dovremmo prendere a modello? In qualsiasi momento, infatti, una macchina potrebbe trovarsi di fronte a scelte importanti che possono riguardare altre macchine ma anche gli esseri umani [24,25,26]. Immaginiamo che la macchina senza guidatore debba scegliere se "sterzare a sinistra e colpire una bambina di otto anni, o sterzare a destra contro una nonna di ottanta anni. Data la velocità della macchina, la vittima sarebbe sicuramente uccisa dall'impatto. Se tu non sterzi, entrambe le vittime saranno colpite e nello scontro moriranno: così abbiamo buone ragioni di pensare che tu devi sterzare in un modo o nell'altro. «Ma – si chiede Patrick Lin – qual è la decisione eticamente corretta? Se tu stesso avessi la responsabilità di programmare la macchina, che cosa le faresti fare, se mai dovesse capitarle di incontrare un caso simile, non importa quanto raro possa essere?» [27]. A prima vista, la cosa più corretta da fare sembrerebbe quella di salvare la bambina, che ha davanti a sé ancora una lunga vita da vivere – ad

esempio, un primo amore, la costruzione di una sua propria famiglia, una carriera e la sua giusta quota di esperienze. Tuttavia, perché la nonna non dovrebbe aver diritto alla stessa rilevanza morale? Se, cioè, è sbagliato discriminare le persone per il genere, la religione o la disabilità, perché dovremmo avere il diritto di trattare le persone diversamente a seconda dell'età? Nemmeno l'altra possibile soluzione, però, sembra eticamente accettabile: potremmo programmare la macchina per non sterzare né a destra né a sinistra, in questo modo, però, condanniamo ad una morta certa sia la bambina che la nonna. Si pensi, poi, a quest'altro scenario: una macchina procede lungo la strada con il proprietario ed altri passeggeri a bordo, proprio al passaggio della macchina, improvvisamente un bambino attraversa la carreggiata per andare a riprendere il pallone. In una situazione di questo tipo cosa dovrebbe fare la macchina senza guidatore? Dovrebbe proseguire per la propria strada ed eventualmente investire anche il bambino, o dovrebbe cercare di evitare la morte del bambino con una manovra sull'altra corsia, anche al rischio in questo modo di causare un incidente con altre macchine ed eventualmente la morte o il grave ferimento di un numero maggiore di persone? È naturale pensare che nel caso in cui la manovra non comportasse alcun altro incidente, sarebbe giusto che la macchina cercasse di evitare di uccidere il bambino: il problema è che sterzando la macchina potrebbe causare un incidente molto più drammatico. Ad esempio, a seguito della manovra, la macchina potrebbe finire contro un autobus: l'autobus è pesante, ma lo scontro con l'automobile potrebbe farlo uscire di strada, si potrebbe ribaltare e poi prendere fuoco, ed i passeggeri non avrebbero scampo. A fronte di questi importanti dilemmi morali, possiamo scegliere di programmare la macchina a seguire un'etica deontologica oppure una morale consequenzialista [28]. L'etica deontologica afferma che un'azione o una scelta è giusta se è conforme ad una regola morale ed è sbagliata se, invece, è in contrasto con una norma morale: queste norme, inoltre, sono valide a prescindere dal fatto che promuovano il bene (ad esempio, la felicità delle persone coinvolte, il soddisfacimento delle loro preferenze oppure qualsiasi altra cosa a cui noi possiamo riconoscere una qualche valore), sono basilari e non sono derivate per induzione da giudizi su casi particolari. Di fatto – afferma Frankena – i sostenitori di un'etica deontologica asseriscono che dobbiamo sempre decidere che cosa fare alla luce di queste norme o principi [29]. Un'etica consequenzialista, invece, sostiene che la giustezza o scorrettezza di un'azione o di una scelta possa essere stabilita soltanto sulla base delle sue conseguenze. Una particolare versione di un'etica consequenzialista è, poi, l'utilitarismo che afferma che il giusto consiste nella promozione del bene generale: «(q)uesto principio afferma decisamente che il fine morale da ricercare in tutto quanto facciamo è la maggiore rimanenza possibile del bene sul male (o la minor rimanenza possibile di male sul bene) nel mondo nel suo complesso. (…) Questo implica che in ogni caso, qualunque cosa siano, il bene e il male sono tali da essere misurabili e confrontabili…» [29].

Se scegliamo un'etica utilitarista dovremmo programmare la macchina per ridurre – in caso di incidente o di scelta morale – il numero di persone ferite e uccise. Oppure, nel caso fosse in gioco lo stesso numero di persone, per salvare la persona più giovane (che ha cioè più anni da vivere) o con una

qualità della vita migliore. Un'etica deontologica, invece, ci vieterebbe sempre di 'sacrificare' persone innocenti, a prescindere dai vantaggi che si potrebbero ottenere in termini di vite salvate, in quanto, in questo caso, la cosa doverosa è rispettare la norma di non uccidere. Il principale problema di avere in circolazione macchine programmate con un'etica deontologica è che esse potrebbero interpretare le norme in maniera troppo letterale. Ad esempio, ipotizziamo che sia possibile programmare le macchine senza guidatore con le tre leggi della robotica di Asimov: 1) un robot non può arrecare un danno ad un essere umano, oppure attraverso il mancato intervento, lasciare che un essere umano sia danneggiato; 2) un robot deve obbedire agli ordini degli esseri umani tranne quando questi ordini sono in conflitto con la prima legge; 3) un robot deve proteggere la sua propria esistenza fintantoché ciò non contrasta con la prima e seconda legge. Davanti al pericolo di una collisione, la macchina potrebbe dare un'interpretazione troppo letterale di queste tre leggi e non frenare per evitare traumi ai passeggeri. Ma per le stesse ragioni la macchina potrebbe rifiutarsi di entrare nel traffico cittadino o di raggiungere certe destinazione, in quanto 'sentirebbe' la responsabilità di minimizzare il rischio di incidente per il conducente e/o le altre persone. Inoltre, più aumentano le regole e i principi morali che bisogna rispettare, più aumenta la probabilità che, in particolari situazioni, due o più principi entrino in conflitto[30]: dovremmo poi programmare le macchine autonome per affrontare queste situazioni e per "sapere" qual è, in caso di conflitto, il principio che deve prevalere. A mano a mano che si presentano nuovi scenari e nuove circostanze dovremmo pertanto aggiungere nuove regole e principi e specificare la loro applicazione. Anche questo, però, non sarebbe risolutivo, in quanto, anche ammesso che sia possibile specificare sempre quando una norma ha la precedenza sull'altra, con lo sviluppo tecnologico, le auto potrebbero trovarsi in situazioni che non avevamo previsto. «Anche se l'etica deontologica può fornire una guida in molte situazioni, non è impiegabile come sistema etico completo a causa dell'incompletezza di ogni insieme di regole e della difficoltà di articolare l'etica umana nella sua complessità in una lista di regole» [31].

Sembra, pertanto, preferibile affidarsi ad un'etica che abbia un unico principio morale. Tuttavia, anche un'etica consequenzialista presenta una serie di problemi: immagiamo, ad esempio, di voler programmare le macchine secondo un'etica utilitaristica. Noi vogliamo che le macchine autonome riducano il più possibile il danno: ma come si calcola il danno? Soltanto in termini di vite umane coinvolte? Oppure consideriamo anche il danno economico (quale sarà il costo complessivo dell'incidente)? Inoltre, per calcolare il danno dobbiamo tener conto soltanto delle conseguenze più immediate o prendere in considerazioni anche gli effetti a lungo termine. Nel primo caso, potendo scegliere se scontrarsi con un motociclista con il casco o un motociclista senza casco, la macchina dovrebbe optare per quello con il casco, in quanto il motociclista con il casco avrebbe maggiori probabilità di sopravvivere. In questo modo, però, si scoraggerebbe l'uso del casco e/o si penalizzerebbero le persone che rispettano la legge ed adottano una guida molto più prudente. Per le stesse ragioni, poi, la macchina autonoma dovrebbe preferire un incidente con una macchina più robusta (ad esempio, un SUV) rispetto ad una più leggera, in

quanto in caso di incidente la macchina più robusta sarebbe molto più resistente e perciò il rischio che i passeggeri siano feriti ed uccisi sarebbe molto più basso. A quel punto, però, chi avrebbe ancora la voglia di comprarsi questa macchina? Inoltre, si tratterebbe di decidere che prospettiva la macchina autonoma dovrebbe assumere: se proteggere sempre gli interessi del proprietario o "conducente": oppure assumere un punto di vista più fermo e generale e proteggere gli interessi della collettività? Immaginiamo ad esempio quest'altra situazione: la macchina potrebbe procedere nel modo più sicuro per il suo proprietario se, ad esempio, investisse il bambino che attraversa improvvisamente la strada per andare a riprendere il suo pallone. In fondo per qualsiasi macchina è meglio scontrarsi con l'"oggetto" più leggero: ad esempio, per lei sarebbe meglio scontrarsi con una moto piuttosto che un SUV. Se pertanto la scelta è tra un bambino e una macchina o addirittura un camion, la macchina avrebbe maggiori possibilità di uscirne indenne se colpisse ed investisse il bambino. L'alternativa è assumere un'altra prospettiva e sterzare su un'altra corsia, perché in questo modo si potrebbe salvare la vita del bambino senza conseguenze collaterali. C'è il rischio, però, che la manovra causi la morte del proprietario della macchina: sterzando, infatti, la macchina potrebbe uscire fuori strada e ribaltarsi, o oltre la strada c'è una foresta e c'è la possibilità che la macchina finisca contro un albero. Se consideriamo le cose moralmente, sembra giusto che la macchina non si preoccupi soltanto del suo proprietario o conducente, ma assuma un punto di vista generale: contano i miei interessi ed il mio benessere, ma anche quelli di qualsiasi persona. Tuttavia, anche in questo caso certe decisioni potrebbero rivelarsi controproducenti: ad esempio, programmare le macchine autonome per proteggere l'interesse generale potrebbe ridurre l'interesse per queste macchine e rallentare la loro diffusione. Le persone, infatti, potrebbero non sentirsi più al sicuro nelle loro autovetture o potrebbero preferire viaggiare su macchine su cui hanno un maggiore controllo. Anche questi sono scenari che un'etica consequenzialista dovrebbe considerare.

Inoltre, una macchina programma a scegliere sempre secondo un'etica utilitaristica e, di conseguenza, per promuovere la massima felicità per il più grande numero di persone potrebbe optare per soluzioni che possono apparire profondamente ingiuste. Pertanto, il criterio per determinare il giusto non può essere pertanto soltanto l'utilità: «(se) la giustizia può avere di tanto in tanto il sopravvento sull'utilità, allora – come ricorda Frankena – la questione di cosa sia giusto non può trovare risposta appellandosi al principio di utilità e i deontologi hanno, dopo tutto, ragione almeno in parte» [29]. In altri termini, entrambe le concezioni etiche presentano vantaggi e svantaggi: «Gli approcci deontologici richiedono regole astratte che possono non essere applicabili o che possono trovarsi in conflitto in particolari situazioni, mentre gli approcci consequenzialisti sono così rigidi che producono azioni che appaiono criticabili» [31]. Tuttavia, la programmazione di una macchina secondo una etica deontologica sembra impraticabile soprattutto per difficoltà interne a questa concezione: incompletezza di ogni sistema di regole e di qualsiasi graduatoria che intenda specificare, nei diversi casi, quale norma o principio morale debba avere la priorità. Anche il ricorso ad una concezione etica consequenzialista presenta

problemi, ma le difficoltà sembrano prevalentemente pratiche e di conseguenza superabili attraverso una selezione condivisa degli obiettivi che la macchina autonoma deve perseguire. Superare il disaccordo può richiedere tempo perché in una società pluralistica, come la nostra, abbiamo concezioni del bene e della giustizia molto diverse. Ma questo risultato può essere raggiunto con il coinvolgimento delle diverse parti in causa (popolazione, ingegneri, programmatori, industrie automobilistiche, ecc.) ed un confronto sui valori che vogliamo vedere rappresentati a livello di progettazione [32, 33, 34, 35].

Conclusioni

In molti ambiti della vita e delle professioni l'inarrestabile avanzata di robot sempre più intelligenti a spese degli umani solleva importanti questioni morali e suscita preoccupazioni importanti riguardo alla capacità delle macchine di rimpiazzare gli umani. Ad esempio, c'è il timore che una cura amministrata da robot non sarebbe più umana in quanto impedirebbe al disabile e al paziente di coltivare importanti spazi di socialità. C'è la preoccupazione, poi, che l'impiego nei conflitti armati di macchine sempre più autonome, possa portare ad una escalation senza precedenti della violenza, in quanto la guerra non sarebbe più combattuta dagli esseri umani, ma semplicemente dai robot. Con l'impiego dei robot, inoltre, la guerra potrebbe perdere la sua umanità, in quanto una macchina intelligente non può essere mossa dalla compassione o dall'empatia. Nel caso, poi, dei sex robot è stato affermato che il loro sviluppo incoraggerebbe la violenza sulle donne rappresentandole come corpi sempre a disposizione dei maschi. Non sembra, invece, che perdiamo qualcosa di importante se smettiamo di guidare o deleghiamo, come finora abbiamo fatto, la guida non ad un conducente di professione (ad esempio, un autista dell'autobus o ad un taxista) ma ad un sistema operativo. Probabilmente, alcune persone si lamenteranno perché non potranno più guidare, ma i benefici a breve e lungo termine per la società potrebbero essere considerevoli, in quanto la guida autonoma potrebbe ridurre il numero di morti sulle strade. Ogni anno, del resto, 1,25 milioni di persone muore a causa di incidenti automobilistici e all'incirca il 90% di questi incidenti dipende da un errore umano [10,11].

Oggi diversi gruppi di ricerca lavorano per rendere i veicoli a guida autonoma sempre più sicuri, ad esempio, attraverso avanzati sistemi di controllo laterale e longitudinale, di intersezione, algoritmi di pianificazione del percorso con specifici requisiti [36]. Le autovetture, però, dovrebbero essere in grado non soltanto di identificare il pericolo (ad esempio, un'altra macchina che sta arrivando sulla corsia opposta, oppure un oggetto che si trova lungo la carreggiata) ma anche reagire prontamente. Un risultato che si cerca di raggiungere predisponendo la macchina a cambiare velocità e, se possibile, anche frenare davanti all'ostacolo in tempi brevissimi [37]. Affinché, però, possano risultare veramente affidabili per i passeggeri e terzi, le macchine autonome dovrebbero essere anche in grado di comportarsi moralmente: questa è la sfida più impegnativa per chi lavora allo sviluppo di questi autoveicoli. Naturalmente se con i veicoli autonomi si potessero eliminare tutti gli incidenti, i problemi sarebbero risolti perché non ci sarebbero più scelte morali

da fare: questo, tuttavia, non sembra – almeno per il momento – lo scenario futuro più realistico. Pertanto, non abbiamo altra scelta: dobbiamo rendere le macchine morali, almeno quanto basta per fare scelte che possano essere considerate accettabili dalla società. Fino ad allora, le macchine senza guidatore saranno semplicemente una speranza.

Bibliografia

[1] Hashim H. H., Omar, M. Z. (2017). "Towards autonomous vehicle implementation: issues and opportunities", *Journal of the Society of Automotive Engineers Malaysia*, 1, 2, 111-123.

[2] Karnouskos S. (2018). "Self-driving car acceptance and the role of ethics", *IEEE Transactions on Engineering Management*, 1-15.

[3] Burns L. D. (2013). "Sustainable mobility: a vision of our transport future", *Nature*, 497, 181-182.

[4] Fagnant, D. J., Kockelman K. M. (2014). Preparing a nation for autonomous vehicles: opportunities, barriers and policy recommendations for capitalizing on self-driven vehicles, paper presented at the 93rd Annual Meeting of the Transportation Research Board, Washington DC.

[5] Fagnant, D. J., Kockelman, K. M. (2014). "The travel and environmental implications of shared autonomous vehicles, using agent-based model scenarios", *Transportation Research Part C: Emerging Technologies*, 40, 1-13.

[6] Sivak, M., Schoettle, B. (2015). *Motion sickness in self-driving vehicles*, University of Michigan Transportation Research Institute.

[7] Litman T. (2019). "Autonomous vehicle implementation predictions implications for transport planning", Victoria Transport Policy Institute, https://www.vtpi.org/avip.pdf (ultimo accesso 25 settembre 2019).

[8] Litman T. (2014). "Ready or waiting", *Traffic Technology International* (www.trafficTechnologytoday.com), 36-42; http://www.vtpi.org/AVIP_TTI_Jan2014.pdf (ultimo accesso 25 settembre 2019).

[9] Sparrow, R., Howard M. (2017). "When human beings are like drunk robots: driverless vehicles, ethics, and the future of transport", *Transportation Research Part C: Emerging Technologies*, 80, 206-215.

[10] World Health Organization (2015), *Global status report on road safety 2015*, Switzerland, World Health Organization.

[11] Anderson, J. M., Nidhi, K., Stanley, K. D., Sorensen, P., Samaras, C., Oluwatola, O. A. (2016). *Autonomous vehicle technology: A guide for policymakers, Rand Corporation*, Santa Monica.

[12] Pakusch C. et al. (2018). "Unintended effects of autonomous driving: a study on mobility preferences in the future", *Sustainability*, 10, 2404, 1-22.

[13] Ferreras, L. E. (2013), "Autonomous vehicles: a critical tool to solve the XXI century urban transportation grand challenge", in Jones, S. L. (a cura di), *Urban public transportation system 2013: proceedings of the third international*

conference on urban public transportation systems, 17-20 November 2013, Paris, 405-412.

[14] Levin, M. W., Boyles, S. D., Patel, R. (2016). "Paradoxes of reservation-based intersection controls in traffic networks", *Transportation Research Part A: Policy and Practice*, 90, 14-25.

[15] Levin, M. W., Boyles, S. D. (2015). "Intersection auctions and reservation-based control in dynamic traffic assignment", *Transportation Research Record: Journal of the Transportation Research Board*, 2497, 35-44.

[16] Dumouchel, P., Damiano L. (2019). Vivere con i robot. Saggio sull'empatia artificiale, Raffaello Cortina, Milano.

[17] Turkle, S. (2012). Insieme ma soli. Perché ci aspettiamo sempre più dalla tecnologia e sempre meno dagli altri, Codice, Torino.

[18] Bryson J. J. (2010). "Robots Should Be Slaves", in Wilks Y. (a cura di), Close Engagements with Artificial Companions: Key Social, Psychological, Ethical and Design Issue, John Benjamins, Amsterdam, 63-74.

[19] Kant I. (1989). Critica della ragion pratica, Laterza, Roma-Bari.

[20] Driver J. (2007). *Ethics. The Fundamentals*, Blackwell Publishing, Oxford, 80-101.

[21] Tegmark M. (2018). Vita 3.0. Esseri umani nell'era dell'intelligenza artificiale, Raffello Cortina, Milano.

[22] Hume D. (1982). Trattato sulla natura umana, Laterza, Roma-Bari.

[23]Aristotele (1999). *Etica Nicomachea*, Laterza, Roma-Bari 1999.

[24] Nihols S., Smids J. (2016). "The ethics of accident-algorithms for self-driving cars: an applied trolley problem?", *Ethical Theory and Moral Practice*, 19, 5, 1275–1289.

[25] Renda A. (2018). "Ethics, algorithms and self-driving cars – a CSI of the 'trolley problem'", *CEPS*, 2, 1-17.

[26] Sabin D. (2017). "New algorithm lets self-driving cars merge with traffic like a school of fish", https://www.inverse.com/article/27119-algorithm-merge-autonomous-highway.

[27] Lin P. (2015). "Why ethics matters for autonomous cars", in Maurer, M., Gerdes, J. C., Lenz, B., Winner, H. (a cura di) *Autonomous driving: technical, legal and social aspects*, Springer, Berlin, 69-85.

[28] Bonnefon, J. F., Shariff, A., Rahwan, I. (2016). "The social dilemma of autonomous vehicles", *Science*, 352, 1573–1576.

[29] Frankena W. K. (1996), Etica. Un'introduzione alla filosofia morale, Edizioni di Comunità, Milano.

[30] Li S., Zhang, J., Wang, S., Li, P., Liao, Y. (2018). "Ethical and legal dilemma of autonomous vehicles: study on driving decision-making model under the emergency situations of red-light running behaviors", *Electronics*, *7*, 264, 1-18.

[31] Goodall N. (2014). “Ethical decision making during automated vehicle crashes”, *Transportation Research Record: Journal of the Transportation Research Board*, 2424, pp. 58-65.

[32] Etzioni A., Etzioni O. (2017). “Incorporating ethics into artificial intelligence”, *Journal of Ethics*, 21, 4, 403-418.

[33] Holstein T. (2017). “The misconception of ethical dilemmas in self-driving cars”, *Proceedings,* 1, 174, 1-3.

[34] Goodall, N. J. (2016). “Can you program ethics into a self-driving car?”, *IEEE Spectr.*, 53, 28–58.

[35] McBride, N. (2016). “The ethics of driverless cars”, *SIGCAS Comput. Soc.*, 45, 179–184.

[36] Milakis, D., van Arem, B., Van Wee, B. (2017), “Policy and society related implications of automated driving: a review of literature and directions for future research”, *Journal of Intelligent Transportation Systems*, 21, 4, 324-348.

[37]Bichiou, Y., Rakha, H. A. (2019), “Real-time optimal intersection control system for automated/cooperative vehicles”, *International Journal of Transportation Science and Technology*, 8, 1, 1-12.

Biografia

Maurizio Balistreri è Ricercatore di filosofia morale presso il Dipartimento di Filosofia e Scienze dell’Educazione dell’Università di Torino. Ha collaborato al *Dizionario di bioetica* curato da Eugenio Lecaldano (2000), è autore di numerosi saggi di bioetica e di *Etica e romanzi* (Le Lettere, 2010), *Superumani: etica ed enhancement* (Espress edizioni 2011), *La clonazione umana prima di Dolly* (Mimesis 2015) *Il futuro della riproduzione umana* (Fandango 2016) e *Sex robot. L’amore al tempo delle macchine* (Fandango 2018; Rowman & Littlefield International 2020).

Twitter: @m_balis

Email: maurizio.balistreri@unito.it

Come adottare le scale di valutazione in ambito forense. [1]

Quando l'ignoranza scientifica diventa un problema etico.

Giuseppe Mastronardi

Sommario

Per ottenere valutazioni obiettive delle prove di colpevolezza (evidence) in un contesto giudiziario, ove si richiede un risultato in termini di identificazione personale, ottenuto confrontando una caratteristica biometrica (voce, volto, altezza, ecc.), i metodi generalmente utilizzati implicano l'estrazione di parametri che vengono osservati e quindi analizzati alla ricerca di caratteristiche (template) che possono essere codificate e confrontate. Queste valutazioni richiedono interpretazioni statistiche appropriate e indicazioni sia dell'intervallo di precisione dello strumento di acquisizione sia dell'errore intrinseco del metodo di rilevazione utilizzato. Finora, sono state elaborate diverse classificazioni di caratteristiche con pesi diversi, al fine di coprire un valore reale di somiglianza, caratterizzante o individualizzante. Pertanto, diventa importante esprimere una valutazione corretta attraverso un adeguato livello di compatibilità, per rappresentare al meglio la rarità di ogni singola caratteristica.

Tuttavia, sarebbe opportuno che questi livelli di compatibilità corrispondano a indici probabilistici che non sono facilmente e numericamente determinabili. Per questo motivo, sono state introdotte scale concettuali testuali. Gli scienziati forensi e le istituzioni forensi internazionali sono costantemente alla ricerca di standard minimi e criteri di valutazione uniformi. Ma, per il corretto utilizzo di queste scale, è comunque necessario che gli esperti forensi siano dotati di adeguate competenze tecniche e scientifiche, insieme a pensiero critico e sensibilità etica.

Abstract

In order to obtain objective evaluations of evidence in a judicial context, the generally used methods involve the extraction of parameters and indices which are analyzed in search of features that can be coded and compared. These evaluations require appropriate statistical interpretations and the range connected both to the accuracy of the eventual acquisition tool and to the intrinsic error of the detection method used. Different classifications of features with different weights have been drawn up, in order to cover a real identity values. Therefore, it becomes important to express a correct evaluation through an adequate level of compatibility, to better represent the rarity of each individual feature in a population. The most reliable evaluation is the one that contemplates a real statistical calculation in terms of probability LR (Likelihood Ratio), and which also expresses the probability of FAR (False Acceptance Ratio) or FFR (False Rejection Ratio). However, these probabilistic indices are not easily and numerically determinable. For this reason, verbal compatibility scales have been introduced. Forensic scientists and International Forensic Institutions are continuously seeking minimum standards and uniform evaluation criteria. But, for the proper use of these scales, it is required that forensic experts are provided with appropriate technical and scientific competences, critical thinking and ethical sensitivity.

Keywords: Digital Forensics, Forensic Ethics, Statistical Evidence, Verbal Compatibility Scales

1. Introduzione

I metodi, generalmente utilizzati per confronti biometrici, fisionomici e antropometrici, comprendono l'osservazione e il confronto di parametri fisici, morfologici e morfometrici. Questi parametri, estratti dai rispettivi reperti (tracce audio, immagini, resti umani, ecc.), vengono analizzati alla ricerca di caratteristiche sufficientemente individualizzanti. Queste caratteristiche, codificate come proprietà fisiche, morfologiche o morfometriche, devono essere supportate da opportune valutazioni statistiche del range di errore, connesso sia allo strumento per l'acquisizione di questi parametri sia al metodo utilizzato per il loro confronto, affinché possano essere scientificamente accettate.

Se in contesti diversi l'impronta digitale costituisce un punto di riferimento universalmente riconosciuto, nel contesto delle immagini di videosorveglianza lo stesso valore è fornito da segni come cicatrici, nei o tatuaggi, o da alcuni distretti corporei come l'orecchio o il naso. Tuttavia, i vari distretti corporei non hanno identico peso per attribuire un'identità: infatti, sicuramente un'analogia

può essere un elemento di compatibilità, ma solo l'unicità (o la rarità) di una singola caratteristica può avere un reale valore identificativo in una popolazione.

Analogamente alle impronte digitali, anche nel caso del volto, del padiglione auricolare o di altre strutture anatomiche, varie classificazioni di caratteristiche (*features*) sono state elaborate con diversi pesi e indici di rarità nella popolazione: alcuni più comuni, altri più peculiari [2,3].

Insieme all'identificazione delle strutture anatomiche con valore identificativo, è anche molto importante esprimere adeguati livelli di compatibilità a seguito dell'analisi delle caratteristiche biometriche di due soggetti messi a confronto. In questi casi, le scale comparative finalizzate a valutare il grado di compatibilità tra l'individuo indagato/imputato e quello ripreso nelle immagini di videosorveglianza sono molteplici e diverse [4].

Ovviamente, la valutazione più attendibile è quella che prevede un calcolo statistico reale in termini di probabilità e "rapporto di verosimiglianza" (**LR**: "Likelihood Ratio"), che esprime la probabilità di una possibile identificazione o, in alternativa, tiene conto della probabilità di una possibile falsa identificazione, quando la stessa caratteristica del sospettato/accusato può essere trovata in qualsiasi soggetto della popolazione

Sebbene la **LR** possa essere definita come il rapporto fra la probabilità che un dato riscontro appartenga al sospettato fratto la probabilità che lo stesso riscontro possa appartenere a una qualsiasi altra persona diversa dall'indagato/imputato, talvolta questi indici probabilistici non sono facilmente calcolabili in modo numerico, per svariati motivi come l'assenza di vere e proprie percentuali di frequenza nella popolazione di una data caratteristica oppure la complessità del concetto stesso di **LR**, senza un'adeguata preparazione matematico-statistica, ecc.).

Infatti, la valutazione più attendibile è quella che prevede un calcolo statistico reale in termini di probabilità **LR** e che esprime anche la probabilità di un Falso Positivo (**FAR** = False Acceptance Ratio) o di un Falso Negativo (**FFR** = False Rejection Ratio).

Per questo motivo sono state introdotte scale concettuali che permettono di tener conto di variabili non numeriche ma verbali (molto raro, raro, ecc.) [5].

Professionisti e Istituzioni Internazionali nel campo delle Scienze Forensi da tempo stanno spingendo l'adozione di standard minimi e criteri di valutazione uniformi a livello Europeo. Vediamone le più comuni, con i relativi vantaggi e limiti [6].

2. Scala di valutazione Italiana del 1997

Seguendo un ragionamento coerente con le attività peritali, per il dato tecnico si parla di "scale di valutazione" e non di "scale di giudizio", poichè il "giudizio" è prerogativa inderogabile del Giudice. Infatti, nessun esperto può emettere "giudizi" ma solo formulare valutazioni su basi tecnico-scientifiche, al fine di fornire qualificati elementi a supporto del più completo quadro indiziario e probatorio. In ambito internazionale la parola "giudizio" è tassativamente proibita all'esperto.

Una "scala verbale" largamente diffusa nel campo delle comparazioni antropometriche, messa a punto nel 1997 da un gruppo di esperti nazionali e da tecnici del Servizio di Polizia Scientifica, tuttora utilizzata in Italia sia da Privati sia da Forze dell'Ordine, è qui di seguito rappresentata:

NON COMPARABILITÀ: quando l'immagine di riferimento per la comparazione non consente un esame comparativo per bassa definizione, esiguità di dettagli, travisamenti e non favorevoli posizioni dei soggetti ripresi; tale valutazione porta a dichiarare "non comparabili" gli elementi a disposizione.

NON COMPATIBILITÀ: quando, nelle due immagini ritraenti gli individui a confronto, è presente almeno un particolare, non posticcio e non alterabile col tempo, che permette di escludere che le due immagini ritraggano lo stesso individuo; tale valutazione porta alla cosiddetta "**esclusione**".

COMPATIBILITÀ PARZIALE: quando, in considerazione della scarsa definizione e/o visibilità nell'immagine di riferimento, non è possibile rilevare particolari anatomici facciali e somatofisici che permettano di giungere a una valutazione positiva di comparazione; si riscontrano, comunque, alcuni particolari simili in entrambi gli individui messi a confronto; tale valutazione porta alla cosiddetta "**affinità**".

COMPATIBILITÀ (SEMPLICE): quando, in entrambi gli individui messi a confronto, gli elementi visibili ed estrapolabili permettono di rilevare numerosi particolari antropometrici (fisionomici e somatofisici) simili; comunque, considerata la bassa definizione di una o più immagini di riferimento, non è possibile evidenziare contrassegni (nei, cicatrici, rughe caratteristiche, ecc.) che possano portare a una più sicura valutazione di "piena compatibilità".

COMPATIBILITÀ TOTALE (o PIENA): quando, nelle immagini degli individui ritratti e messi a confronto, sono presenti molti particolari fisionomici simili, comprese le relative proporzioni antropometriche, oltre a eventuali particolarità anatomiche singolari e contrassegni simili.

Questa tipologia di scala ha sicuramente il vantaggio di essere piuttosto chiara e di facile interpretazione per un lettore "non tecnico", tuttavia è asimmetrica e non completa. E' asimmetrica perché ci sono 3 livelli positivi (affinità o compatibilità parziale, compatibilità semplice, compatibilità totale o piena) e uno solo rappresenta la negatività del confronto (esclusione o non compatibilità) e non prevede la valutazione di situazioni intermedie che, nei fatti, o vengono dirottate verso un'esclusione certa, oppure archiviate con l'espressione "non comparabile". In altri termini, risulta riduttiva, poiché non contempla un sufficiente livello di sfumature e non presenta un egual numero di livelli nella direzione di una maggior compatibilità e di assenza di compatibilità.

I livelli di positività, a loro volta, appaiono piuttosto qualitativi e incentrati soprattutto sul confronto diretto delle caratteristiche a disposizione, come nella "sovrapposizione parametrizzata"[7]. Infatti, solo in presenza di caratteristiche fisionomiche e somatofisiche sovrapponibili, questa metodica consente il confronto di interessanti parametri numerici ottenuti dalle geometrie misurate come distanze relative e fattori di forma[8]. Ciononostante, non viene espressa la valutazione della rarità degli elementi riscontrati. La risposta che si fornisce in

questi casi al Giudice potrebbe, pertanto, essere considerato solo come possibile indirizzo identificativo, comunque privo di un'apprezzabile classificazione; non fornisce, infatti, la completezza degli elementi che dovrebbero concorrere a una decisione certa, cioè priva di possibili valutazioni soggettive.

Si diceva che la valutazione più affidabile è quella che prevede il calcolo statistico della **LR** che è piuttosto complesso, ma certamente dotato di basi oggettive e scientifiche. Alcuni autori hanno studiato una sorta di tabella di conversione, che possa tradurre un esito numerico in una frase più comprensibile, senza perderne però il presupposto probabilistico. Tale conversione prevede un giudizio del tipo "Gli esiti degli esami forniscono molto forti / forti / deboli / neutrali elementi a supporto dell'ipotesi di identificazione o non identificazione". In ogni caso, anche la risposta "verbale" deve essere motivata e supportata da una valutazione matematica o pubblicazione scientifica e non sul semplice intuito o su una presunta e soggettiva "esperienza".

3. Il parere dell'ENFI

La tipologia di scala sopra riportata non è la più raccomandata dall'ENFI (European Network of Forensic Science Institutes) che da venti anni promuove l'utilizzo dell'approccio Bayesiano che ha il vantaggio di essere trasversale a tutte le branche delle Scienze Forensi [9]. Questo approccio non è solamente applicabile ai confronti antroposomatici, ma si estende a tutti gli esami che possono fornire un'identificazione di un soggetto o di un oggetto. Sul portale dell'ENFSI (www.enfsi.eu) è disponibile una nutrita bibliografia e, in particolare, nell'ottobre 2015 è stato pubblicato il risultato di un lavoro triennale, indirizzato alla presentazione in ambito forense di risultati di carattere scientifico (ENFSI GUIDELINE for evaluative reporting in Forensic Science), che rientra in un più vasto progetto finanziato dalla Comunità Europea [10].

In questo ampio lavoro dell'ENFI il tema centrale è l'utilizzo del rapporto di verosimiglianza (**LR**) che è il rapporto fra la probabilità che un dato riscontro sia riconducibile al "nostro" sospettato e la probabilità che lo stesso riscontro possa appartenere a una qualsiasi altra persona diversa dall'indagato/imputato del caso in esame. Questo è l'elemento chiave per una più corretta scala di valutazione, ed è quanto un esperto dovrebbe esattamente produrre.

Per conferire concretezza, fruibilità e applicabilità più universale al metodo, anche con tale valutazione quantitativa si è pensato di esprimere il dato numerico ottenuto sotto forma di una più ampia scala verbale qualitativa [11].

In altri termini, avendo a disposizione il dato quantitativo, questa operazione consiste in una banale "trascrizione" in parole di un numero; tuttavia, considerando che in molti casi è quasi impossibile riuscire a fornire un valore preciso, sia perchè in alcuni campi questi indici probabilistici non sono noti o facilmente calcolabili (come per assenza di vere e proprie percentuali di frequenza nella popolazione di una data caratteristica), sia perché non tutti gli esperti possiedono sufficienti conoscenze nel campo statistico-matematico,

l'approccio prevede la possibilità di una stima "soggettiva", l'importante è focalizzare l'attenzione sul fatto che all'esperto sono sempre richiesti due valori: il livello di positività dell'esame e la rarità degli elementi di riscontro [12].

Rifacendoci alla formula di Bayes, l'elemento che spetta all'esperto è quindi la **LR**, che NON è il risultato finale della probabilità di colpevolezza/innocenza, che può da solo portare al verdetto, ma rappresenta quella componente intermedia che, moltiplicata per la "probabilità a priori" (*prior odds*), può fornire il risultato che si cerca, la "probabilità a posteriori" (*posterior odds*).

La coerenza epistemologica impone, pertanto, che per il dato scientifico ci si esprima sempre come elemento a supporto di un'ipotesi (quantificazione a priori) e mai in termini di "probabilità" in senso generico (perché genera confusione), né tantomeno come "probabilità finale" (perché sarebbe un errore, in quanto mancherebbe il contributo dei prior odds).

Proprio sull'onda di questo, l'ENFSI (European Network of Forensic Science Institutes), che è una delle più importanti organizzazioni europee a cui afferiscono le più accreditate istituzioni forensi, ivi incluse le Forze dell'Ordine di molti Paesi Europei, fra cui Polizia e Carabinieri, ha stilato alcune linee guida ufficiali che comprendono anche una sezione relativa alle comparazioni di caratteristiche estratte da immagini acquisite da impianti di videosorveglianza.

4. Scala di valutazione di Nordgaard (a 9 livelli)

La scala valutativa, proposta da Nordgaard [11], ricalca quella già stilata nel 2011 dal Swedish National Laboratory of Forensic Science che esprime i livelli di compatibilità/identificazione in un range da -4 a + 4. La scala valutativa proposta recepisce le linee guida tracciate da Aitken [2] ed Evett [3] nella fine degli anni '90 e si presenta, per praticità, su 9 livelli:

LIVELLO -4 LR < 0,0001

Il risultato dell'analisi supporta l'ipotesi di **molto forte esclusione** (non identificazione). La possibilità che gli stessi elementi possano essere trovati nel sospettato può essere esclusa con assoluta certezza.

LIVELLO -3 LR < 0,001

Il risultato dell'analisi supporta l'ipotesi di **forte esclusione**. La possibilità che gli stessi elementi possano essere trovati nel sospettato può essere considerata inverosimile.

LIVELLO -2 LR < 0,01

Il risultato dell'analisi supporta l'ipotesi di **discreta esclusione**. Vi sono elementi differenzianti che riducono ragionevolmente la possibilità che possano essere trovati nel sospettato.

LIVELLO -1 LR < 0,1

Il risultato dell'analisi supporta l'ipotesi di **limitata esclusione**. Il bilanciamento degli elementi è lievemente maggiore a supporto di una **non compatibilità** piuttosto che di una compatibilità.

LIVELLO 0 LR ~ 1

Il risultato dell'analisi è **inconclusivo**. Gli elementi non sono confrontabili o sono utilizzabili o sono talmente limitati qualitativamente, o comuni in una popolazione, che non lasciano propendere per un'ipotesi rispetto a quella alternativa.

LIVELLO +1 LR > 10

Il risultato dell'analisi supporta l'ipotesi di **limitata identificazione**. Il bilanciamento degli elementi è lievemente maggiore a supporto di una identificazione piuttosto che all'ipotesi contraria.

LIVELLO +2 LR > 100

Il risultato dell'analisi supporta l'ipotesi di **discreta identificazione**. Vi sono elementi compatibili di discreta rarità che riducono ragionevolmente la possibilità di essere riscontrati in un soggetto diverso.

LIVELLO +3 LR > 1000

Il risultato dell'analisi supporta l'ipotesi di **forte identificazione**. Gli elementi emersi conferiscono per qualità e/o quantità un indice di rarità fortemente individualizzante.

LIVELLO +4 LR > 10.000

Il risultato dell'analisi supporta l'ipotesi di **molto forte identificazione**. La possibilità che questi risultati possano essere trovati anche in un soggetto diverso dal sospettato può essere considerata inverosimile.

Dovendo il numero di livelli non essere limitatamente "rigidi", per conferire simmetria alla scala, è necessario che il loro numero sia dispari, dovendo contemplare un "livello zero" in caso si giunga a un esame inconclusivo. La classificazione dei livelli da -4 a +4 non è casuale, perché questi numeri rappresentano l'esponente della base 10 che esprime il relativo valore della **LR**.

Come si può notare questa scala completa, in realtà, quella "italiana" del '97. Infatti, i livelli di confronto positivo +4, +3, … prevedono che vi sia un certo grado di compatibilità, oltre a una valutazione più marcata dell'aspetto di tipicità degli elementi riscontrati. Con questo tipo di scala non è, quindi, importante la quantità di caratteristiche misurate (o misurabili), quanto la loro rarità o incidenza nella popolazione.

In altri termini, se gli elementi caratterizzanti fossero solo 3, altezza, colore degli occhi e colore dei capelli, la scala italiana non potrebbe andare oltre la "compatibilità parziale", perché mancherebbero all'appello la maggioranza di altri elementi che andrebbero normalmente e correttamente essere considerati. Tuttavia, se chi ha compiuto il reato, per esempio una rapina in banca, risultasse più alto di 2 metri, avesse gli occhi rossi e i capelli bianchi, ecco che le conclusioni cambierebbero sensibilmente tra le due scale. Infatti, con la scala europea è richiesto di valutare quanto sia inconsueta la presenza sul territorio di un albino (occhi rossi e capelli bianchi) più alto di 2 metri. Poiché tale frequenza nella popolazione è abbondantemente inferiore a 1/10.000 (PFI: probabilità di falsa identificazione), avremmo una **LR** (1/PFI) superiore a 10.000, quindi un livello +4.

In modo analogo, la scala proposta dall'ENFSI, concettualmente del tutto sovrapponibile e dotata dello stesso rigore scientifico, ribadisce la necessità di esprimere una valutazione della probabilità che la tesi relativa al quesito posto dall'Autorità Giudiziaria, ossia della probabilità che presentando le caratteristiche riscontrate, il soggetto indagato/imputato e il reo siano la medesima persona, sia supportata rispetto a quella della tesi contraria (che un altro soggetto qualsiasi, diverso dall'indagato/imputato, possa presentare le medesime caratteristiche: falso positivo). In altri termini, qualsiasi sia la scala verbale ritenuta più idonea dall'operatore, è fondamentale che presenti maggiore scientificità, cioè il minor numero possibile di elementi soggettivi.

Per sintetizzare il concetto di "valore identificativo" e "scala valutativa", a mero titolo esemplificativo, trattiamo l'esempio di un riconoscimento biometrico basato solo su due caratteristiche anatomiche, per esempio l'altezza e il colore dei capelli. Il primo test valuta se un soggetto biondo alto 207,2 cm (indagato) può essere distinguibile o meno da un altro, biondo di 207 cm (rapinatore), mentre il secondo test esplicita che, essendo questi valori inconsueti per la popolazione italiana, il dato che integra le due caratteristiche insieme (biondo + 207 cm) ha un elevato potere identificativo (cioè piccolo errore di falsa identificazione/falso positivo), cioè equiparabile a un livello +3 della scala Nordgaard, corrispondente a $1000<LR\leq10000$ della scala ENFI.

Analogamente, sempre sulla base dei due parametri, altezza e colore dei capelli, valutiamo se un soggetto con i capelli neri e alto 175 cm possa essere distinguibile da uno, sempre con capelli neri e alto 174 cm. In questo caso la statura rientra nella media della popolazione italiana (dato attestato da statistiche ufficiali [13]), pertanto è una caratteristica molto comune, e selezionare i soggetti con i capelli neri, solo sulla base di quelle due caratteristiche, non permettere un alto livello di compatibilità, ma sola una sorta di "somiglianza" quantificabile con un livello +1 della scala Nordgaard, corrispondente a $1<LR\leq10$ della scala ENFSI.

In questo modo si consente a un Magistrato di comprendere il valore della valutazione tecnica espressa, e permette così, l'inserimento di tale dato nell'insieme degli elementi che vanno a formare il giudizio complessivo, di esclusiva pertinenza di un Giudice.

Come si è detto, l'assenza di un database inerente ai termini di paragone sopra indicati non permette di quantificare numericamente il grado di confronto; ma la conclusione delle diverse analisi e misurazioni dovrà comunque esprimere un grado di riconducibilità o meno degli elementi estratti dalle immagini a un unico soggetto.

5. Conclusione

Dunque, se si riesce a calcolare numericamente la **LR**, diventa immediato poter associare il livello di una qualunque scala simmetrica; viceversa, partendo da una base soggettiva o testuale si deve poter giungere a stimare il corrispondente valore numerico percentuale, con la migliore approssimazione possibile e verificare la sua attendibilità.

Quindi, per semplicità, ma nel pieno rispetto delle scale proposte dall'ENFSI, qualora non sia calcolabile un effettivo valore numerico di **LR** si potrebbe adottare, con molta cautela e dovizia di particolari, la seguente scala verbale, concettualmente corretta e del tutto sovrapponibile alle altre:

Elementi insufficienti a esprimere una valutazione

La pessima definizione e visibilità in almeno una delle due immagini a confronto non consente di individuare caratteristiche o connotati somatici comuni non posticci né alterabili o difformi.

Limitati elementi a supporto dell'ipotesi di identità/esclusione

La pessima definizione e visibilità in almeno una delle due immagini a confronto consente di individuare solamente in forma limitata caratteristiche o connotati somatici non posticci né alterabili.

Discreti elementi a supporto dell'ipotesi di identità/esclusione

Definizione e visibilità non ottimali in almeno una delle due immagini a confronto consente di individuare in numero congruo caratteristiche o connotati somatici non posticci né alterabili.

Forti elementi a supporto dell'ipotesi di identità/esclusione

Nelle due immagini a confronto sono individuabili, in forma evidente, numerose caratteristiche o connotati somatici non posticci né alterabili. Sono, altresì, evidenziabili elementi salienti o particolari contrassegni nei due soggetti messi a confronto.

Molto forti elementi a supporto dell'ipotesi di identità/esclusione

Nelle due immagini a confronto sono individuabili in forma molto evidente numerose caratteristiche o connotati somatici non posticci né alterabili. Sono, altresì, evidenziabili molti elementi salienti o particolari contrassegni nei due soggetti messi a confronto.

Certamente questa scala di valutazione non consente di esprimere un valore probabilistico, numericamente riferito a una popolazione, ma può agevolare la lettura del risultato qualitativo, in modo da rendere non equivoca la sua corretta interpretazione in termini di identificazione.

Per il corretto utilizzo di queste scale, è perciò necessario che esperti forensi siano dotati di adeguate competenze tecniche e scientifiche, pensiero critico e sensibilità etica.

Bibliografia

1. Mastronardi G. (**2019**) "Evaluation Scales in Search of Scientific Evidence in Judicial Processes: An Ethical Behavior Indispensable for Forensic Experts", Speech held at the 36th International Congress IALMH on Law and Mental Health (Medical Etichs session), Rome (Italy), July 22, 2019
2. Aitken C. (**1995**) "Statistics and the Evaluation of Evidence for Forensic Scientists", Chichester: John Wiley &Sons, Ltd. 1995

3. Evett I.W. et al. (**2000**) “Reporting Conventions”, Proceedings of Second European Academy of Forensic Science Meeting, Kracow, 12-16 sept. 2000
4. Sjerps M., Biesheuvel D.B. (**1999**) “The interpretation of conventional and Bayesian verbal scales for expressing expert opinion: a small experiment among jurists”, Forensic Linguistics, The University of Birmingham, University Press, 1999; 6(2): 214-227
5. Wojcikievicz J. (**2000**) “Scientific Evidence in Judicial proceedings”, Institute of Forensic Research Publishers, Krakow-Poland
6. Garbolino P. (**2007**) “Nuovi strumenti logici e informatici per il ragionamento giudiziario: le reti bayesiane”, Cass. Pen. 2007: 326-339
7. Introna F., La Sala L., Mastronardi G. (**1993**). Identificazione Personale Mediante Confronto Computerizzato di Immagini Registrate su Supporto Magnetico. In: Quaderni del CSM (Consiglio Superiore della Magistratura) “Falcone/Borsellino”, Anno 8, N. 61, IV Suppl., Vol. V, Roma, Febbraio 1993, p. 84-94
8. Mastronardi G. (**2014**). Identificazione Personale mediante Confronto di Volti. In: Mondo Digitale, vol. XIII, p. 53-64, ISSN: 1720-898X
9. Aitken C., Roberts P., Jackson G. (**2011**) “Fundamental of Probability and Statistical Evidence in Criminal Proceedings. Guidance for Judges, Lawyers, Forensic Scientists and Expert Witnesses”, Royal Statistical Society’s Working Group on Statistics and the Law (Adopted as ENFSI guidelines)
10. Nordgaard A., Ansell R., Drotz W., Jaeger L. (**2012**) “Scale of conclusions for the value of evidence”, Law, Probability and Risk, 2012; 11: 1-24
11. Zadora G., Martyna A., Ramos D. and Aitken Colin (**2014**) “Likelihood Ratio Models for Classification Problems”, Published Online: 3 JAN 2014 DOI: 10.1002/9781118763155. ch5 2014 © John Wiley & Sons, Ltd 2014
12. Wikipedia (Stature medie in Italia), http://it.wikipedia.org/wiki/Statura

Biografia

Giuseppe Mastronardi, già professore ordinario di Sistemi di Elaborazione delle Informazioni al Politecnico di Bari, ove ha insegnato Calcolatori Elettronici, Informatica Medica e Sicurezza Informatica, è attualmente docente del corso di Information Systems Security and Privacy nei corsi di Laurea Magistrali in Ingegneria delle Telecomunicazioni e Ingegneria dei Sistemi Medicali. È autore di circa 200 lavori scientifici nei seguenti settori di ricerca: visione artificiale e tecniche biometriche, informatica medica e sicurezza dati, privacy e computer ethics. Collabora dal 1978 sul territorio nazionale con Procure e Tribunali per l’identificazione personale mediante confronto di voci e volti. Impegnato come DPO in una campagna di formazione sulla privacy in sanità a seguito del GDPR, ha pubblicato un testo su “Sanità

Digitale e Privacy". Fondatore e presidente della Sezione Territoriale AICA-Puglia nel 2011, è stato Presidente AICA per il triennio 2016-2018. Componente e Probiviro del Distretto Produttivo dell'Informatica Pugliese, per oltre 40 anni ha indirizzato giovani ingegneri di numerose aziende locali e nazionali alla progettazione di soluzioni e tecnologie informatiche.

Tecnologie digitali e manipolazione del comportamento

Silvia Crafa, Matteo Rizzo

Sommario

L'odierna e crescente intima relazione uomo-tecnologia ha generato una discussione sempre più approfondita rispetto alle problematiche legate alle modalità di progettazione delle innovazioni tecnologiche. Ciò conferisce implicitamente un importante ruolo sociale ed etico alle software house, che contribuiscono a realizzare prodotti in grado di influenzare il comportamento degli utenti fino ad arrivare a manipolarlo. Tuttavia, questo stesso potere persuasivo rappresenta altresì un potente mezzo per catalizzare comportamenti positivi. In quest'ottica sorge il movimento di "progettazione etica", finalizzato alla promozione del benessere digitale e alla revisione in senso umano-centrico delle modalità di realizzazione della tecnologia.

Abstract

Today's growing and intimate relationship between people and technology has generated an increasingly in-depth discussion of the problems correlated to the design methods of technological innovations. This implicitly confers an important social and ethical role to the software houses, which contribute to the creation of products that can influence the behavior of the users to the point of manipulating it. However, this same persuasive power also represents a powerful means to catalyze positive behaviors. With this in mind, the "ethical design" movement was born, aimed at promoting digital well-being and revising the methods of technology implementation in a human-centric fashion.

Keywords: habit-forming technology, hook model, gamification, dark patterns, ethical design, ethics in computer science

1. Introduzione

Un numero sempre crescente di persone impiega una parte considerevole del proprio tempo interagendo con dispositivi digitali, fruendo di un grande numero di servizi il cui utilizzo si inserisce più o meno consapevolmente nelle loro abitudini quotidiane. Social network, applicazioni per la produttività e per lo svago, per il fitness e per l'e-commerce, sono software utilizzati comunemente da milioni di individui e il cui successo spesso coincide con l'essere accuratamente progettati per renderne l'uso una consuetudine. Innumerevoli prodotti digitali, infatti, competono costantemente per l'attenzione dell'utente, al fine di insediarsi nella sua routine e di venire utilizzati con la maggiore frequenza possibile. La tecnologia, se appositamente concepita, è infatti un mezzo in grado di alterare il comportamento degli utenti in modo silenzioso ma allo stesso tempo profondo, fino ad indurre vere e proprie abitudini. Il potere sociale che ne deriva può essere impiegato per stimolare l'utente a una condotta tanto positiva quanto negativa. Potrebbe aiutarlo a smettere di fumare o a perdere peso, così come essere utilizzato al solo scopo di monetizzarne l'attenzione, con il rischio di condurlo a comportamenti compulsivi o addirittura con caratteri di dipendenza patologica. L'esito dipende in gran parte dalle intenzioni dei progettisti e dalla consapevolezza degli utenti.

La tecnologia continua incessantemente il suo rapido processo di democratizzazione, diventando accessibile ad un numero sempre crescente di persone. La riduzione dei costi e una maggiore attenzione nei confronti dell'esperienza utente hanno infatti favorito l'accesso a prodotti e servizi tecnologicamente sofisticati. Tuttavia, sebbene da un lato negli ultimi decenni questo fenomeno abbia portato innumerevoli vantaggi sul piano economico e sociale, dall'altro la crescita del grado di comprensione generale del suo funzionamento e dei suoi potenziali rischi non tiene il passo con la velocità con cui si intensifica la relazione tra individuo e dispositivo digitale [1]. La cosiddetta *computazione pervasiva*, ovvero l'innesto di capacità computazionali in oggetti della forma più svariata, già sfiora la quotidianità del mondo contemporaneo, evidenziando i limiti e i pericoli dell'essere perennemente circondati dalla tecnologia e facendo crescere la consapevolezza della questione morale ad essa legata.

Tecnologia per la manipolazione del comportamento

Secondo Nir Eyal, consulente per l'industria tecnologica, *"le tecnologie che utilizziamo si sono trasformate in compulsioni, se non in dipendenze a pieno titolo [...] Si tratta dell'impulso di controllare la notifica di un messaggio, dell'attrazione nei confronti del visitare YouTube, Facebook o Twitter solo per pochi minuti, solo per ritrovarsi ancora a toccare e scorrere lo schermo un'ora dopo. Niente di tutto questo è un incidente, tutto corrisponde alle intenzioni dei progettisti"* [2].

Nel suo testo più famoso [2], Eyal spiega i sottili trucchi psicologici che possono essere utilizzati per far sì che le persone sviluppino abitudini legate ad un prodotto. Alcuni esempi di rilievo riguardano l'identificazione di emozioni che

possano fungere da “fattori scatenanti” l’uso, o la variazione nella tipologia e nella frequenza delle gratificazioni offerte, atte a generare interesse e motivazione. Per fare un esempio concreto, le piattaforme sociali come Facebook e LinkedIn, richiedendo all'utilizzatore di coinvolgere i suoi contatti nella piattaforma, sfruttano il naturale bisogno di reciprocità sociale per ampliare il proprio bacino di utenza e incoraggiarne l’interattività. Facebook, in particolare, dichiara di essere in grado di identificare quando gli adolescenti si sentono “insicuri”, “inutili” e “hanno bisogno di gratificazione e fiducia”, e utilizza queste informazioni per sollecitare l’utente con contenuti mirati [3]. Inoltre, i pochi secondi che si frappongono tra l’apertura dell’applicazione mobile del colosso dei social network e la rivelazione del suo feed, connotati dalla schermata monocromatica riportante il suo logo, potrebbero sembrare una normale forma di caricamento del software, oppure una disfunzione di rete o del dispositivo. In realtà, quel breve lasso di tempo è stato accuratamente progettato per alimentare la curiosità dell’utente nello scoprire su quali contenuti si soffermerà la roulette di generazione del suo *feed* delle notizie. La generazione di tale *feed* sfrutta infatti il programma a rapporto variabile [4]: il principio psicologico su cui fanno leva anche le *slot machine* per generare un desiderio di utilizzo costante; il tutto al fine di rendere il prodotto più desiderabile e accattivante, prolungare il tempo di permanenza sulla piattaforma e aumentare i conseguenti introiti [5].

Il modello “a uncino”

Scelte progettuali di questo tipo, se ben inserite nella narrativa del prodotto, vanno a sostenere un polivalente modello di generazione delle abitudini detto modello “a uncino” (schematizzato in *figura 1*). Si tratta di un particolare ciclo esperienziale attraversato dall’utente, progettato appositamente per garantire un’interazione frequente e costante con il prodotto. Il modello si struttura secondo quattro fasi principali, che si ripetono e incrementano regolarmente: la fase di (1) innesco, in cui all’utente viene indicato come agire, quella di (2) azione, che lo conduce alla (3) ricompensa, e infine quella di (4) investimento, che lo lega indissolubilmente al prodotto [2, 6].

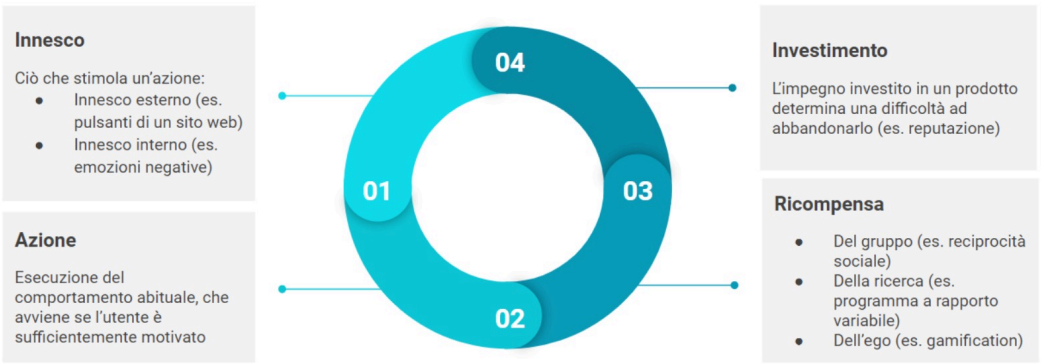

Figura 1
Schematizzazione del modello a uncino teorizzato da Nir Eyal

L'innesco consiste in una *call to action* esterna o in un'esigenza interna che induce l'utente ad agire in un certo modo. Esempio di innesco esterno è l'arrivo di una e-mail che solleciti un'azione, come l'invito a visitare un negozio online da parte di una newsletter, mentre l'innesco interno ha spesso a che fare con le emozioni, come la solitudine o la noia, che, ad esempio, fanno percepire come più interessante l'arrivo di una notifica dal social network di turno. L'azione, poi, viene definita da Eyal come "*il comportamento più semplice in previsione della ricompensa*". Maggiore è lo sforzo, fisico o mentale, richiesto dall'azione, e meno è probabile che questa verrà compiuta. La ricompensa è invece il fine dell'azione, il momento in cui le esigenze dell'utente vengono soddisfatte. Molto spesso si tratta di una gratificazione sociale, come quella che si percepisce alla notifica dell'apprezzamento di uno dei nostri post. Infine, l'investimento fa leva sul fenomeno per cui impiegare tempo e fatica in un prodotto o servizio ne aumenta il valore percepito. Tale fenomeno può essere rilevato nell'esperienza accumulata nel giocare ad un particolare videogioco, cosiccome nel tempo trascorso per rendere perfetta la pagina del proprio profilo del social network di turno.

Alla luce della sua struttura, appare evidente come il modello "a uncino" sia un potente mezzo per rendere un prodotto assuefacente e veicolare una vera e propria manipolazione del comportamento degli utenti. Le conseguenze del suo utilizzo o di tecniche affini non sono necessariamente negative, se ad esempio utilizzate per promuovere comportamenti e abitudini positive, ma resta rilevante il fatto che sono in gran parte determinate dalla consapevolezza e dalle intenzioni dei progettisti.

La "*gamification*" e il modello "esca e amo"

Si pensi ai numerosi elementi di *game design* introdotti in moltissime applicazioni per smartphone di uso quotidiano, come l'accumulo di esperienza, punti, medaglie e ricompense di vario genere. A prima vista, potrebbero sembrare semplici meccaniche studiate per rendere più piacevoli determinate operazioni. Ciò è in parte vero: gli elementi di *game design* rendono effettivamente l'interazione con l'applicazione più divertente, stimolante e, per i più avvezzi ad attività videoludiche, anche più semplice. Le app per il fitness, per esempio, fanno spesso uso di barre di progresso o permettono all'utente di accumulare punti esperienza fino al raggiungimento del proprio obiettivo (es. peso forma, lungo periodo di allenamento ininterrotto), ricompensato da un distintivo commemorativo. Gli elementi tipici di una app che fa uso di *gamification* sono esemplificata in *figura 2.*

Tuttavia, mediante la gratificazione offerta dalle meccaniche di gioco, si persegue l'ulteriore finalità di generare un potente strumento per "autoalimentare" l'interesse dell'utente, possibilmente senza che lo stesso nemmeno se ne accorga e con risvolti opinabili [7, 8, 9]. In particolare, sempre più spesso si vedono introdotte nelle applicazioni delle forme di ricompensa casuale acquistabili attraverso microtransazioni, la maggior parte delle volte mediate da una valuta virtuale che ha corrispondente valore reale. Queste ricompense casuali sono comunemente definite "*lootbox*", dei "bottini misteriosi" che contengono premi di varia natura e rarità tendenzialmente circostanziati

all'applicazione stessa (es. bonus per potenziare il proprio avatar, moneta virtuale, punti esperienza aggiuntivi, ecc..). Chiaramente, le *lootbox* affollano in primis i videogiochi, specialmente quelli per dispositivi mobili, che spesso hanno un target molto giovane. Per questo motivo è cresciuta nei consumatori e, più recentemente, nella politica, la preoccupazione che questi meccanismi di ricompensa randomizzati possano costituire una forma di gioco d'azzardo non regolamentato destinato ai minori, con possibili forme di ludopatia direttamente collegate al fenomeno. Effettivamente, alcuni studi [10] evidenziano proprio come i meccanismi sottostanti la relazione tra giocatori e *lootbox* siano simili, se non identici, a quelli che si verificano nel gioco d'azzardo. Ciò non sorprende quando si nota come nel meccanismo delle *lootbox* si riscontrino alcuni dei più notevoli principi progettuali coinvolti nelle dipendenze digitali. La randomicità della ricompensa si riconduce alla schedulazione a rapporto variabile, utilizzata per catturare ripetutamente l'attenzione dell'utente e gratificarlo, e l'intero sistema è spesso catalizzato da elementi di interazione sociale e sistemi di notifiche pressanti.

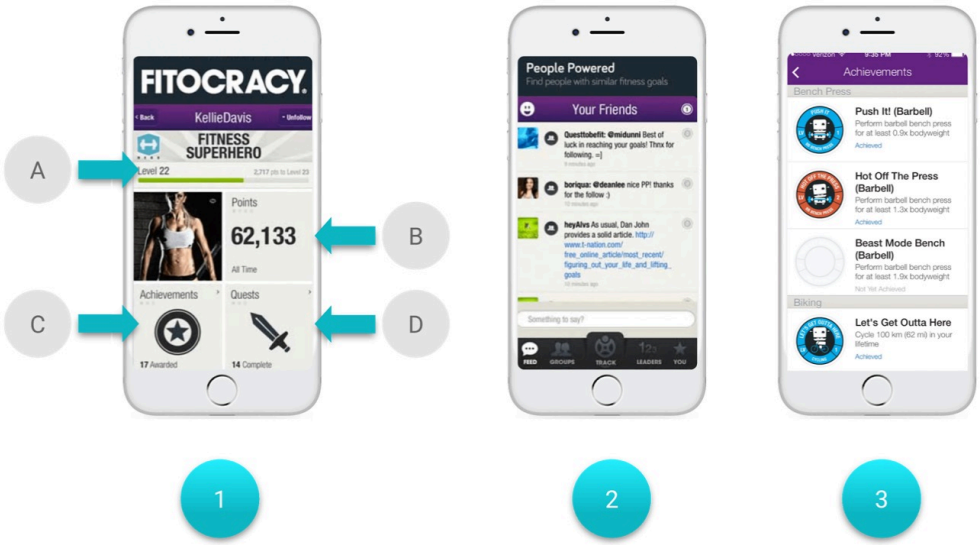

Figura 2

Tre schermate esemplificative tratte dall'interfaccia utente di Fitocracy, app per il fitness che fa ampio uso di gamification. Nella schermata (1) si notano: (A) barra di progresso, (B) punti esperienza, (C) obiettivi raggiunti, (D) "missioni" disponibili (es. correre per cinque chilometri entro la giornata corrente). La schermata (2) mostra la chat di gruppo offerta dall'applicazione, forte incentivo sociale alla realizzazione degli obiettivi proposti. La schermata (3) illustra alcuni distintivi commemoranti il raggiungimento degli obiettivi da parte dell'utente.

In un'ottica economica, quello che soggiace la scelta di utilizzare un sistema di *lootbox* e microtransazioni è un modello di business "esca e amo" dalla grande efficacia. Gli innumerevoli giochi gratuiti che popolano i negozi di applicazioni, specialmente quelli per dispositivi mobili, sono progettati per offrire un tempo di

gioco limitato e livelli difficili che richiedono l'acquisto di potenziamenti per essere superati. Questi potenziamenti appaiono spesso in forma di *lootbox* e possono essere ottenuti tramite microtransazioni *in-game*. L'utenza coglie il valore insito nel download gratuito del gioco, l'*esca*, e il suo volume cresce rapidamente grazie all'abbattimento delle barriere all'entrata. L'utente abbocca invece all'*amo* quando non riesce a superare un livello difficile senza acquistare un potenziamento. Usando questo modello di business, coadiuvato da ulteriori meccaniche di gioco assuefacenti, giochi come Candy Crush Saga sono stati in grado di guadagnare le impressionati somme di $600.000 al giorno [11]. Il potenziale economico di una forma di tecnologia che appare all'utente tanto avvincente e che contemporaneamente è in grado di plasmarne il comportamento è diventato dunque palese, cosiccome l'opinabilità etica del suo utilizzo.

I *dark pattern*

In un'ottica più generale, la "*gamification*" tende ad indurre nell'utente un certo comportamento in maniera implicita, privandolo in parte del controllo sulla sua interazione con il software. Tale concetto appare ancor più evidente quando si analizzano i cosiddetti *dark pattern*, schemi progettuali pensati per far compiere ingannevolmente all'utente azioni che non necessariamente egli voleva compiere, con l'obiettivo, ad esempio, di prolungarne il tempo di permanenza sulla piattaforma o di addebitargli acquisti indesiderati [12, 13]. I *dark pattern* possono avere numerose implementazioni, cosiccome essere più o meno dannosi per l'utente e conformi alla legge, ma in ogni caso rappresentano un esempio eclatante di progettazione non etica che predilige l'obiettivo economico piuttosto che quello umano.

Un famoso *dark pattern* è rappresentato dagli annunci camuffati da funzionalità al fine di racimolare click con l'inganno. Un esempio a riguardo si può trovare in Softpedia, popolare sito di download di software gratuito sostenuto dalle inserzioni pubblicitarie: spesso il portale mostra annunci promozionali che raffigurano un pulsante di download, inducendo gli utenti a fare clic su di essi anziché ottenere ciò che desiderano (come evidenziato in *figura* 3).

In maniera più sottile agisce invece la cosiddetta "fatturazione con rifiuto esplicito", pratica che consiste nell'aggiungere al carrello dell'utente prodotti non richiesti che gli verranno poi messi in conto a meno di una sua negazione esplicita. Prima che questo *dark pattern* fosse reso illegale nel Regno Unito e nella maggior parte dei Paesi dell'Unione Europea, il portale per l'acquisto di domini web GoDaddy forniva un perfetto esempio a riguardo. Per confermare l'operazione di acquisto di un dominio su tale sito era infatti richiesto l'inserimento di informazioni di vario tipo (es. nome del dominio, dati dell'acquirente, ecc) in molteplici passi, ad ognuno dei quali corrispondeva l'inserimento nel carrello di un ulteriore prodotto indesiderato (es. servizio privacy, assistenza, ecc). Ciò avveniva attraverso una casella di controllo preselezionata sulla conferma dell'aggiunta, che doveva essere spuntata dall'utente per esplicitare il suo rifiuto (mostrata in *figura* 4).

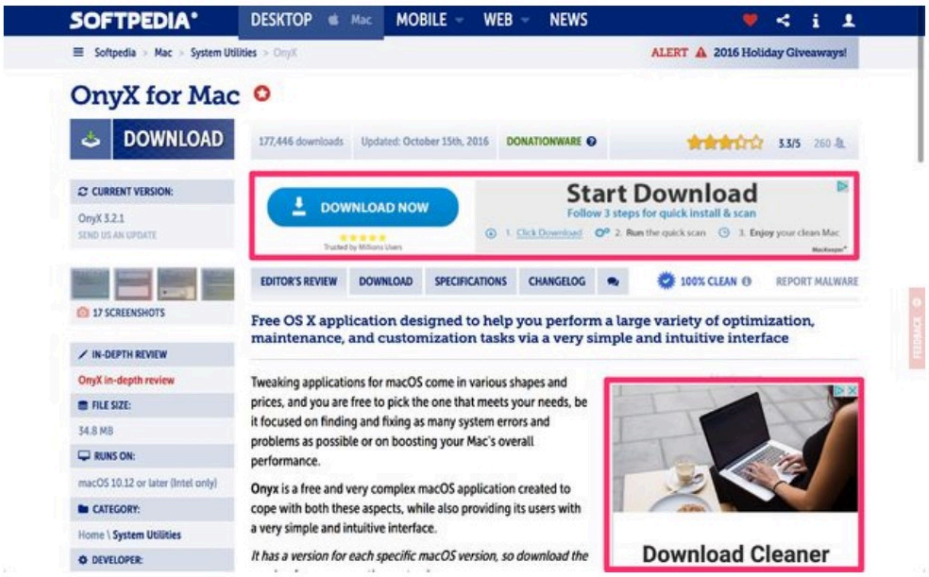

Figura 3
Esempio di annuncio camuffato in una pagina di Softpedia.

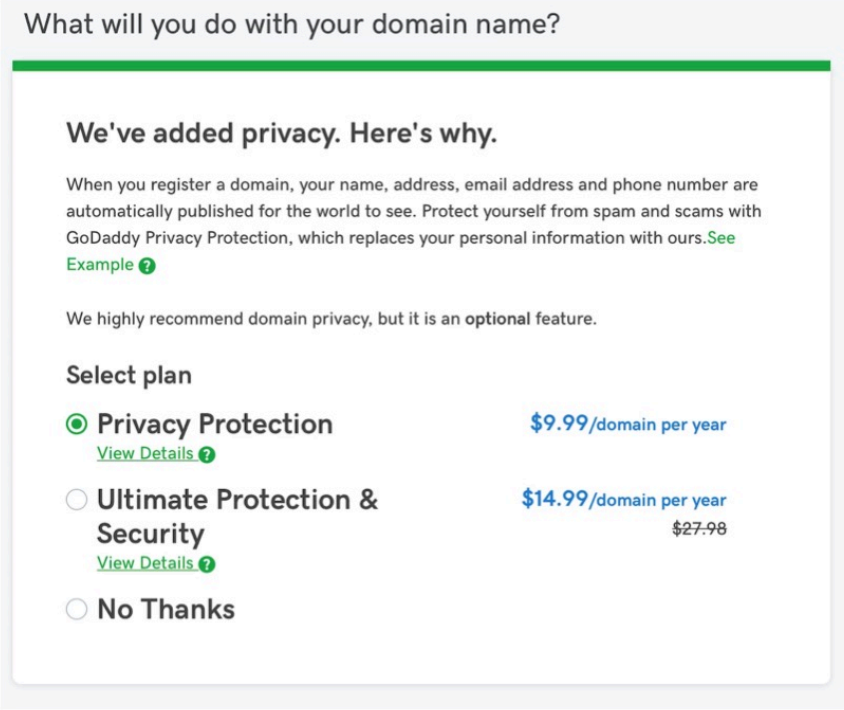

Figura 4
Esempio di "fatturazione con rifiuto esplicito" sul portale GoDaddy.

Più in generale, ciò che molti *dark pattern* cercano di costruire è un *roach motel*, un contesto in cui il design rende appositamente molto facile entrare in una certa situazione, ma estremamente difficile uscirne. Le rivendite online di biglietti come Livenation e Ticketmaster hanno fatto ampio uso di questa strategia per diversi anni. Durante l'acquisto dei biglietti, il sito di turno cercava di intrufolare nel carrello dell'utente un abbonamento a una rivista tramite una domanda trabocchetto nella pagina relativa al pagamento, in cui veniva richiesto di negare esplicitamente il consenso all'acquisto della sottoscrizione. In tal caso, il *roach motel* si palesava nel momento in cui l'utente si rendeva conto troppo tardi di aver effettuato l'acquisto indesiderato, e che l'unico modo per esercitare il diritto di recesso consisteva nello scaricare un modulo, stamparlo, compilarlo a penna e inviarlo all'azienda tramite posta ordinaria.

Progettare per il benessere digitale

Riconoscere *dark pattern* e, più in generale, esperienze utente fallaci, è un primo passo necessario ad aumentare la propria consapevolezza nei confronti di rischi e responsabilità concernenti la progettazione della tecnologia. Ciononostante, è altresì importante agire concretamente per stabilire una prassi di progettazione incentrata sull'essere umano, sia a livello personale che nella propria organizzazione. Con alcune accortezze, infatti, è possibile trarre beneficio da mezzi ad oggi utilizzati impropriamente per migliorare l'esperienza utente nel rispetto del benessere digitale delle persone. Con l'espressione "benessere digitale" si tende ad indicare il benessere psicofisico legato all'interazione con dispositivi e contenuti digitali. La locuzione spesso coinvolge anche un insieme di pratiche e di risorse dedite a promuovere il miglioramento e la consapevolezza del rapporto tra tecnologia e utente. In particolare, l'accento cade sulla qualità del tempo speso online e su come la tecnologia non debba rappresentare una fonte di distrazione, bensì catalizzare produttività e creatività nel rispetto del benessere psicologico di ciascun individuo. La ricerca ha infatti mostrato che sia la motivazione che il benessere dipendono dalla soddisfazione di determinati bisogni psicologici, che spesso però sono presi in scarsa considerazione nel processo di progettazione della tecnologia. Malgrado ciò, ogni tecnologia può influenzare deliberatamente o inavvertitamente il benessere dei suoi fruitori. Ad esempio, alcuni studi mostrano evidenza su come il numero di volte in cui le persone siano in condizione di controllare la posta elettronica accresca il loro stress, mentre ulteriori studi dimostrano come la semplice presenza di uno smartphone in una stanza degradi la qualità dell'interazione tra gli individui [14].

Per far fronte al problema, è necessario dapprima ripensare al modo in cui ci si approccia alla progettazione della tecnologia. A questo proposito, alcuni designer schematizzano una nuova formula di "progettazione etica" secondo una gerarchia piramidale (mostrata in *figura* 5) in cui, dalle fondamenta alla sommità, si sovrappongono strategie a sostegno dei diritti, dello sforzo e dell'esperienza dell'utente [15, 16, 17, 18].

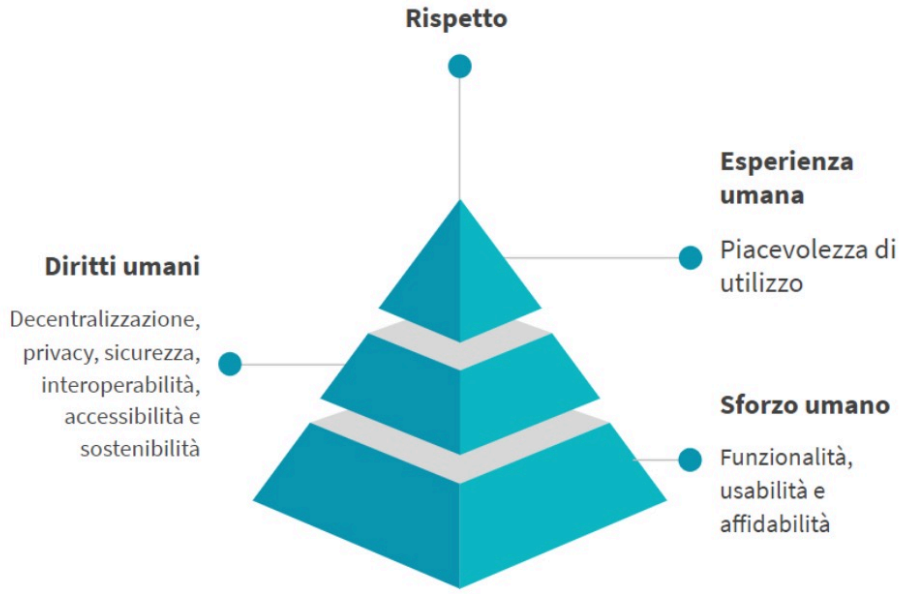

Figura 5
Schema piramidale dei principi di progettazione etica.

Vista la natura della gerarchia, ciascun livello poggia sullo strato sottostante: se un livello è compromesso, quelli superiori crolleranno a loro volta. In altri termini: se un design non supporta i diritti dell'utente, non è etico, cosiccome non lo è se li supporta ma non ne rispetta lo sforzo rivelandosi funzionale, utile e affidabile. Da un punto di vista pratico, ciò significa che prodotti e servizi che sfruttano in modo improprio dati personali, utilizzano *dark pattern* e, più in generale, agiscono a scapito del benessere dell'utente, non sono etici. Si delineano dunque alcune *best practice*: l'uso dei dati per migliorare l'esperienza umana, quello di annunci senza tracciamento, la priorizzazione di trasparenza, soddisfazione dell'utente e riduzione delle barriere di utilizzo. Più in generale, quel che si auspica è una forma di progettazione umano-centrica che rappresenti non solo un framework operativo, ma anche una determinata mentalità. Ciò implica il coinvolgimento degli utenti già in una fase progettuale, e l'interagirvi continuativamente nel corso dell'intero ciclo di vita del prodotto ricercandone attivamente bisogni e difficoltà, fornendo aiuto e soluzioni.

Metodi di promozione del benessere digitale

In seguito alle recenti ricerche che evidenziano come alcuni modelli di utilizzo dei media digitali siano problematici e soddisfino i criteri della dipendenza comportamentale [19, 20], le domande sull'etica e la responsabilità delle società che producono software sono in aumento. Con il crescere dell'attenzione pubblica sul tema, crescono anche le proposte di soluzioni atte a promuovere il benessere digitale e l'offerta di risorse e strumenti per aumentare la

consapevolezza degli utenti a questo riguardo. Nello specifico, alcune analisi della correlazione tra i dati di monitoraggio dell'utilizzo delle app e i punteggi associati a parametri di dipendenza (es. frequenza di utilizzo, tasso di ricaduta, ecc) rivelano che orari di apertura, durata e regolarità d'uso sono buoni indicatori di un utilizzo problematico della tecnologia [21]. Per questo motivo, gran parte dell'attuale promozione del benessere digitale passa attraverso la diffusione tra gli utenti della consapevolezza su modalità e tempistiche di interazione con i loro dispositivi. A tal proposito, funzionalità di tracciamento dell'interazione utente-smartphone pensate per il grande pubblico stanno venendo implementate in abbinata a funzionalità che permettono di autolimitare l'utilizzo dei dispositivi o di specifiche applicazioni.

Particolare rilievo assume lo sforzo di Google (senz'altro commisurato alla dimensione e agli interessi dell'azienda, specialmente in termini di immagine pubblica) nel realizzare risorse di varia natura per promuovere il benessere digitale. In particolare, l'azienda dichiara di voler permettere agli utenti di comprendere le proprie abitudini digitali, di concentrarsi maggiormente e disconnettersi dal mondo digitale per ridurre lo stress che esso genera. Il perseguimento di tali obiettivi si concretizza nella realizzazione di contenuti didattici rivolti a vari tipi di pubblico target, dai bambini ai professionisti, ma anche in nuove funzionalità integrate nei principali prodotti Google, dal sistema operativo Android ai servizi per l'intrattenimento come Youtube, dedite a permettere una gestione semplificata e maggiormente efficace dei sistemi di notifica e di monitoraggio delle proprie abitudini. Alla conferenza per gli sviluppatori I/O 2018, il CEO di Google Sundar Pichai ha per l'appunto annunciato una funzionalità chiamata *Digital Wellbeing*, progettata per aiutare gli utenti Android a monitorare il loro utilizzo di smartphone e app, eventualmente autolimitandolo [22, 23]. La funzionalità, che va ad integrarsi direttamente con il sistema operativo, è stata rilasciata in versione beta in concomitanza con la prima *release* stabile di Android 9.0 Pie nell'agosto 2018, per poi uscire in versione 1.0 nel novembre dello stesso anno. Questa nuova funzionalità di Android include una efficace *dashboard* sul tempo di utilizzo di ciascuna app con la possibilità di limitarne l'uso, nonché numerosi grafici che dettagliano, tra le altre metriche, quante volte l'utente sblocchi il telefono durante il giorno e quante notifiche riceva. Non da meno è stata Apple, la principale concorrente di Google nel campo dei sistemi operativi per smartphone, introducendo funzionalità del tutto analoghe in *Screen Time,* estensione del sistema operativo di Cupertino rilasciata per iPhone e iPad con la *release* Apple iOS 12 nel settembre 2018 [24].

Conclusione

Nel suo permeare la quotidianità delle persone, la tecnologia rappresenta sempre più una componente sociale di rilievo. La consapevolezza delle opportunità che offre, insieme ai nuovi pericoli che genera, sono aspetti fondamentali della contemporaneità che appaiono ancora troppo trascurati da individui e istituzioni. Ciò espone gli utenti a rischi concreti per il proprio benessere psicofisico ed esibisce le loro vulnerabilità ad aziende in grado di

sfruttare le potenzialità manipolatorie della tecnologia a fini economici, con modalità sovente eticamente discutibili.

Sebbene la promozione del benessere digitale e di una progettazione umano-centrica della tecnologia siano temi ad oggi ampiamente discussi, esiste ancora un ampio margine di implementazione di tali innovative strategie e politiche, considerando la velocità e la differenziazione con cui si sviluppano le tecnologie software e la loro applicabilità e interazione con i dispositivi personali e sociali. In particolare, emerge una pressante necessità di indagare la questione della moralizzazione della tecnologia [25], ovvero la scelta di sviluppare applicazioni che limitano esplicitamente o implicitamente le scelte e le azioni dell'utente in termini morali. Un esempio di tecnologia moralizzante potrebbe essere quello di un'applicazione che disabiliti alcuni servizi dello smartphone qualora si sia raggiunto un determinato limite di utilizzo (es. disattivazione delle app sociali dopo la mezzanotte o dopo un'ora di utilizzo continuativo). Tale applicazione va chiaramente a vincolare lo spettro di azioni che l'individuo può intraprendere (introducendo, per l'appunto, un vincolo morale). D'altra parte, questo esempio evidenza il ruolo di mediazione ricoperto dalla tecnologia: un'applicazione che disattivi alcuni servizi sta offrendo la capacità di decidere quando e quanto essere connessi, e dunque di prendere decisioni che impattano direttamente sulla qualità della vita dell'utente. Quando la tecnologia assume una così importante rilevanza morale, mediando l'etica, è necessario dunque interrogarsi su chi effettivamente decida quali vincoli questa debba veicolare. Nell'esempio in questione, la scelta di quali servizi limitare e della modalità con cui ciò debba accadere potrebbe essere lasciata completamente nelle mani dell'utilizzatore, oppure, come più spesso accade, il ventaglio di scelte potrebbe essere ristretto a priori dall'azienda produttrice. Se è vero che i vincoli di una tecnologia moralizzante, proprio come quelli imposti dalla legge, hanno il delicato compito di limitare la libertà umana per uno scopo superiore, vi è altresì un'importante differenza: le leggi vengono decise in maniera democratica, mentre le scelte progettuali relative ad una tecnologia sono realizzate autonomamente da aziende private.

Bibliografia

[1] Law, N. & Woo, David & Wong, Gary. (2018). A Global Framework of Reference on Digital Literacy Skills for Indicator 4.4.2.

[2] Eyal, N. (2014). Hooked: How to build habit-forming products, Portfolio Penguin.

[3] Levin, S. (2017). "Facebook told advertisers it can identify teens feeling 'insecure' and 'worthless'", *The Guardian*. https://www.theguardian.com/technology/2017/may/01/facebook-advertising-data-insecure-teens (ultimo accesso ottobre 2019).

[4] Cherry, K. (2019). "Variable-Ratio Schedules Characteristics". *Very Well Mind*. https://www.verywellmind.com/what-is-a-variable-ratio-schedule-2796012 (ultimo accesso ottobre 2019).

[5] Hartmans, A. (2018). “These are the sneaky ways apps like Instagram, Facebook, Tinder lure you in and get you 'addicted'”. *Business Insider*. https://www.businessinsider.com/how-app-developers-keep-us-addicted-to-our-smartphones-2018-1?IR=T (ultimo accesso ottobre 2019).

[6] Liu, A., & Li, T. (2016). “Develop Habit-forming Products Based on the Axiomatic Design Theory”, *Procedia CIRP*, Volume 53, 119-124.

[7] Hyrynsalmi, S., Smed, J., & Kimppa, K. (2017). “The Dark Side of Gamification: How We Should Stop Worrying and Study also the Negative Impacts of Bringing Game Design Elements to Everywhere”, *Proceedings of the 1st International GamiFIN Conference*.

[8] Toda A.M., Valle P.H.D., Isotani S. (2018) “The Dark Side of Gamification: An Overview of Negative Effects of Gamification in Education”. In: Cristea A., Bittencourt I., Lima F. (eds) *Higher Education for All. From Challenges to Novel Technology-Enhanced Solutions*. HEFA 2017. Communications in Computer and Information Science, Volume 832. Springer, Cham, 143-156.

[9] Ali, R. (2018). “Digital Motivation, Digital Addiction and Responsibility Requirements”. *2018 1st International Workshop on Affective Computing for Requirements Engineering (AffectRE)*, 27-27.

[10] Drummond, A., & Sauer, J.D. (2018). “Video game loot boxes are psychologically akin to gambling”. *Nature Human Behaviour*, 2, 530-532.

[11] Dickey, M. R. (2013). “Insanely Popular Game Candy Crush Saga Is Bringing In An Estimated $633,000 A Day”. *Business Insider*. https://www.businessinsider.com/candy-crush-saga-daily-revenue-2013-7?IR=T (ultimo accesso ottobre 2019).

[12] https://www.darkpatterns.org/ (ultimo accesso ottobre 2019).

[13] Gray, C.M., Kou, Y., Battles, B., Hoggatt, J., & Toombs, A. (2018). “The Dark (Patterns) Side of UX Design”. *CHI*.

[14] Firat, Mehmet. (2013). “Multitasking or Continuous Partial Attention: A Critical Bottleneck for Digital Natives”. *Turkish Online Journal of Distance Education*. 14. 266-272.

[15] https://2017.ind.ie/ethical-design/ (ultimo accesso ottobre 2019).

[16] Batchelor, Rachel, and Ania Bobrowicz. (2014). “Empathic and Ethical Design of Technology.” In *Universal Access in Human-Computer Interaction. Design and Development Methods for Universal Access*, 3–10. Springer International Publishing.

[17] Iqbal, J., & Beigh, B.M. (2017). “Computer Ethics from Obscure to Ubiquitous”. *International Journal of Advanced Research in Computer Science*. 8.

[18] Iqbal, J., & Beigh, B.M. (2017). “Computer Ethics: Job of Computer Scientist”. International Journals of Advanced Research in Computer Science and Software Engineering. 7.

[19] Cash, H., Rae, C.D., Steel, A.H., & Winkler, A.W. (2012). "Internet Addiction: A Brief Summary of Research and Practice." *Current Psychiatry Reviews* 8 (4): 292–98.

[20] Eide, Tine A., Sarah H. Aarestad, Cecilie S. Andreassen, Robert M. Bilder, and Ståle Pallesen. (2018). "Smartphone Restriction and Its Effect on Subjective Withdrawal Related Scores." *Frontiers in Psychology* 9 (August): 1444.

[21] Ding, X., Xu, J., Chen, G., & Xu, C. (2016). "Beyond Smartphone Overuse: Identifying Addictive Mobile Apps". *CHI Extended Abstracts*, 2821-2828.

[22] https://wellbeing.google/ (ultimo accesso ottobre 2019).

[23] Faulkner, C. (2019). "Google now requires Digital Wellbeing and USB-C PD charging standard for new Android phones". *The Verge*. https://www.theverge.com/2019/10/7/20902889/google-android-digital-wellbeing-parental-controls-usb-c-pd-charging-requirements (ultimo accesso ottobre 2019).

[24] https://www.apple.com/newsroom/2018/06/ios-12-introduces-new-features-to-reduce-interruptions-and-manage-screen-time/ (ultimo accesso ottobre 2019).

[25] Poel, I. van de, & Royakkers Lambèr M. M. (2011). "Moralizing Technology" in "Ethics, Technology, and Engineering: an Introduction" (pp. 206-211) (1st ed.). Malden, MA: Wiley-Blackwell.

Biografia

Silvia Crafa è ricercatrice confermata presso il Dipartimento di Matematica dell'Università di Padova. La sua attività di ricerca in ambito informatico si focalizza sui metodi formali per l'analisi dei sistemi concorrenti e dei linguaggi di programmazione. Studia inoltre l'impatto sociale delle tecnologie digitali in ottica interdisciplinare. È autrice di numerose pubblicazioni su prestigiose riviste internazionali e collabora con diverse istituzioni straniere. È membro del Laboratorio Nazionale CINI su Informatica e Società ed e' stato membro del working group Informatics Europe e EU-ACM per la definizione di un libro bianco sugli algoritmi di decisione automatica.

Email: silvia.crafa@unipd.it

Matteo Rizzo è uno studente magistrale del corso di laurea in Informatica dell'Università degli Studi di Padova. Laureatosi nel 2018 nello stesso ateneo, sta ora focalizzando i suoi studi su intelligenza artificiale e apprendimento automatico, mostrando inoltre particolare interesse nei confronti di come si articoli il moderno rapporto tra società e tecnologia.

Email: matteo.rizzo.4@studenti.unipd.it

Il business della vendita di dati, pratiche e conseguenze nell'etica sociale

Silvia Crafa, Alessandro Zangari

Sommario

L'abbondanza e la facilità di accesso ad informazioni relative ad entità e utenti del Web e alle loro abitudini, hanno condotto al proliferare di compagnie, note con il nome di data broker*, che sfruttano questa ricchezza, raccogliendo ed elaborando tali informazioni per venderle ad altre aziende. La crescente rilevanza che l'utilizzo dei servizi Internet sta assumendo nella società di oggi, unita al progresso degli strumenti di analisi dei* big data*, rende possibile ai data broker ottenere informazioni sempre più rilevanti, specifiche e sensibili. Si evidenzia dunque la necessità di porre degli obiettivi di trasparenza verso il pubblico, nonché di comprendere le implicazioni etiche dei servizi offerti da queste compagnie.*

Abstract

The wealth of digital information and personal data freely accessible on the Web has encouraged many companies, known as data brokers*, to exploit such resources. These entities gather and process large amounts of information to later sell them to other companies. Today, the combined relevance of Internet services in people's lives and the advancements in* big data *analysis allows data brokers to retain larger and larger quantities of remarkably precise and sensitive personal data. It becomes therefore necessary to enforce the principles of transparency and ethical usage of consumer information.*

Keywords: Data broker - Data segments - Data onboarding - Predictive policing - Big Data Ethics - Digital footprint

1. Introduzione

Il ruolo di mediatore che la tecnologia digitale ricopre all'interno delle ordinarie attività umane produce un costante flusso di dati generati dai dispositivi utilizzati quotidianamente, come carte di credito, smartphone e dispositivi indossabili. Sebbene non sia difficile rendersi conto che questi dati possano essere raccolti da alcune aziende, non è immediato comprendere la portata di questa pratica e capirne lo scopo e la rilevanza, sia economica che sociale.

Le informazioni raccolte possono sembrare innocue, ma in verità questi dati, estratti per periodi di tempo significativi e integrati con altre informazioni su abitudini e stato sociale (es. stipendio percepito, cronologia di navigazione, informazioni su abbonamenti), possono fornire un quadro complessivo molto preciso su ogni individuo. La nostra identità è infatti definita in funzione delle esperienze che viviamo, luoghi in cui lavoriamo o interagiamo con gli altri, interessi, famiglia e persone che ci circondano. Proprio queste informazioni, parte integrante della nostra individualità, alimentano un'industria multi-miliardaria, il cui valore stimato supererà nel 2020 i 200 miliardi di dollari [1].

Questo valore deriva in gran parte dalla vendita di *dati personali*, che il vigente Regolamento Generale per la Protezione dei Dati (GDPR) definisce "qualsiasi informazione riguardante una persona fisica identificata o identificabile", dove "si considera identificabile la persona fisica che può essere identificata, direttamente o indirettamente, con particolare riferimento a un identificativo come il nome, un numero di identificazione, dati relativi all'ubicazione, un identificativo online o a uno o più elementi caratteristici della sua identità fisica, fisiologica, genetica, psichica, economica, culturale o sociale" [2, art. 4]. Un dato è un identificatore indiretto, e pertanto comunque considerato *dato personale*, se è possibile risalire all'identità dell'interessato utilizzando tale dato congiuntamente ad altre informazioni [3]. In certi contesti, anche i *dati comportamentali*, sono da considerarsi dati personali; questi dati sono mantenuti sotto forma di una cronologia di azioni che un utente, anonimo o identificato, ha compiuto durante la sua permanenza su un sito web, o più in generale rappresentano un'indicazione di quale comportamento ha avuto un individuo in un determinato contesto [4]. Dati di questo tipo possono includere la cronologia delle pagine visualizzate, dei click effettuati, delle dinamiche di utilizzo del mouse, che possono indicare dove un visitatore di un sito web ha esitato a cliccare, o su quali prodotti si è soffermato maggiormente [5].

Peraltro, non è detto che queste informazioni riguardino solo attività online: ad esempio, i rivoluzionari supermercati Amazon Go non dispongono delle tradizionali "casse", ma tengono invece traccia degli acquisti dei consumatori utilizzando centinaia di telecamere e addebitano gli acquisti direttamente sull'account collegato [6]. Un sistema di questo tipo è concepito per la raccolta di innumerevoli dati sul comportamento dei singoli consumatori di fronte agli scaffali, che possono essere elaborati tramite *analitiche* (glossario) e utilizzati all'interno di altri prodotti. Tutti questi dati, una volta estratti, possono anche essere venduti, scambiati e integrati con quelli di altre aziende partner.

La psicologa sociale S. Zuboff chiama questo scenario *mercato dei behavioural futures,* per sottolineare quanto l'accumulazione sia mirata alla comprensione del comportamento e della socialità delle persone, caratteristiche che possono così essere sfruttate per accrescere il coinvolgimento degli utenti o la fedeltà verso un marchio [7].

Gli snodi di questa rete di scambio sono aziende chiamate *data broker*, che operano raccogliendo e acquistando dati da diverse fonti, processandoli e vendendoli ai propri clienti. Il potere, sotto forma di conoscenza, in grembo a queste aziende, le rende ideali per lanciare campagne pubblicitarie o per avviare macchine di propaganda che possono avere rilevanti conseguenze sul tessuto sociale e non si limitano quindi ad influenzare la vita online dei consumatori.

2. Le fonti dei dati

Il business dei data broker si regge sull'aggregazione di grandi moli di dati, chiamati *big data***,** che possono essere ottenuti con facilità monitorando le tracce digitali degli individui.

Il modello di business più comune per la raccolta di dati è quello basato sull'offerta di servizi o prodotti gratuiti, che essendo utilizzati da milioni di persone, consentono ai fornitori degli stessi di ottenere un enorme quantità di dati comportamentali, spesso raccolti senza un consenso informato da parte degli utenti [7]. Non sempre però tali informazioni provengono esclusivamente dal tracciamento online. Una relazione del 2014 della Federal Trade Commission [8], che ha analizzato le pratiche di raccolta e utilizzo dei dati di 9 data broker, ha catalogato le fonti di dati in 3 categorie: fonti governative, commerciali e pubbliche.

È riportato che, fra le *fonti governative*, l'U.S. Census Bureau forniva informazioni demografiche su quartieri urbani, tra cui informazioni sulle etnie, età, occupazione e livello educativo degli abitanti, mentre amministrazioni locali e tribunali fornivano informazioni sulle licenze professionali, proprietà, ipoteche e fascicoli giudiziari.

Le *fonti pubbliche* includono blog e social network, fra cui LinkedIn, dove sono raccolte informazioni di utenti che hanno impostato bassi livelli di protezione del proprio profilo. I dati raccolti includono potenzialmente tutto ciò che è visibile pubblicamente, inclusi post, immagini, e condivisioni.

Le *sorgenti commerciali* sono aziende che vendono dati ai data broker, e fra queste si annoverano case editrici, che forniscono informazioni sugli abbonamenti dei propri lettori, istituti finanziari, negozi, che forniscono informazioni più o meno specifiche sugli acquisti effettuati, come la categoria dell'oggetto acquistato (es. scarpe di alta moda, cibo per cani, prodotti sanitari), l'importo pagato e il metodo di pagamento. Infine, in molti casi, diverse informazioni sono acquistate da altri data broker, che possono averle ottenute direttamente o indirettamente da fonti terze. Ad esempio, IDMERIT, un'azienda attiva nel business dei dati, dichiara di avere a disposizione informazioni su utenti in oltre 90 paesi, provenienti da governi, liste elettorali, registri delle tasse, uffici anagrafe, servizi postali e aziende di telecomunicazioni [9].

2.1 Un esempio: i dati di Acxiom

La maggioranza dei data broker, fornendo l'accesso ai dati come un servizio, rendono possibile accedervi tramite *richieste HTTP* (glossario), ed espongono delle interfacce (API) per integrarle nelle applicazioni software. Una di queste è Acxiom, che offre molteplici servizi, fra cui risoluzione dell'identità e servizi cloud di analisi dei dati (analitiche). L'azienda dichiara di possedere fino a 10'000 attributi per ogni individuo, con una copertura di 2.5 miliardi di persone in oltre 60 paesi, fra cui l'Italia [10]. Collegandosi al portale per sviluppatori è possibile sfogliare pagine di documentazione che riportano quali attributi possono essere disponibili per ogni consumatore [11]. Si tratta di informazioni di ogni tipo, da quelle demografiche, a dati sulle abitazioni, come date di costruzione e insediamento, proprietà, valore catastale, e anche dettagli riguardo all'edificio, come categoria urbanistica (appartamento, villa, ecc.), materiale di costruzione, il numero di stanze da letto, tipologia di riscaldamento, metratura e molto altro.

Tra i pacchetti di dati disponibili vi è anche quello dedicato ai dati finanziari, che contiene informazioni riguardo ad entrate annue, propensione agli investimenti, situazione finanziaria, frequenza di utilizzo della carta di credito, mentre in altri pacchetti sono indicate informazioni sull'affiliazione politica, religiosa, l'appartenenza etnica, i canali di acquisto preferiti e le abitudini di shopping. I valori degli attributi possono essere cifre o categorie precise, intervalli di valori e in alcuni casi categorie probabilistiche (es. "likely", "very likely").

Acxiom offre la possibilità di ottenere un *portrait*, ovvero un ritratto di un consumatore arricchito con tutte le informazioni sopra citate in possesso dell'azienda. È anche specificato che in mancanza di alcune informazioni, i campi sono riempiti con i valori statisticamente più frequenti in base ai parametri di input forniti. Pertanto il "ritratto" fornito non sarà sempre preciso, ed in alcuni casi costituisce una rappresentazione statistica approssimativa dell'individuo [12].

Va puntualizzato che tale documentazione descrive un'interfaccia standard, e non vi è quindi garanzia che tali attributi siano disponibili per tutti i 2.5 miliardi di consumatori profilati da Acxiom.

3. La trasformazione dei dati

I dati raccolti e venduti senza apportarvi modifiche sono chiamati *dati grezzi*. Questi possono essere lavorati per estrarne informazioni raffinate da incorporare in prodotti venduti dall'azienda. Ad esempio, se una persona visita spesso un sito di vendita di scarpe ed è abbonata a riviste di alta moda, può essere dedotto il suo interesse per le scarpe di alta moda. Attributi di questo tipo, inferiti a partire da quelli aggregati dall'azienda, sono definiti *dati derivati*.

Nelle piattaforme di *segmentazione dei consumatori* i dati, sia grezzi che derivati, sono organizzati per creare *segmenti di dati*, ovvero categorie che raggruppano i consumatori con precise caratteristiche. Questi segmenti possono essere prodotti definendo funzioni di similarità fra consumatori utilizzabili all'interno di algoritmi di *apprendimento non supervisionato* (glossario). In secondo luogo è possibile estrarre un sottoinsieme di consumatori che hanno acquistato una certa categoria di prodotti, e utilizzare le

caratteristiche in comune per costruire un modello predittivo da applicare a nuovi consumatori al fine di determinare la probabilità che acquistino quella tipologia di prodotti, avvalendosi di algoritmi di *apprendimento supervisionato* (glossario) [8].

I segmenti di dati dividono quindi i consumatori in gruppi in base alle necessità dei clienti dei data broker. Nello studio della FTC, viene evidenziato come alcuni segmenti siano focalizzati su minoranze o sullo stato socio-economico: ad esempio la categoria "Resolute Renters" è descritta come composta da "consumatori tra i 30 e 40 anni, single e senza figli, frequenti affittuari e al livello più basso di guadagni e peso sociale", oppure "Tradition & Timecards" composta da "lavoratori di età media 53 anni, che rappresentano gli ispanici meno acculturati che risiedono in aree urbane" [8]. Altre categorie sono "Credit Worthiness", "Allergy sufferer" e "Financially Challenged". Acxiom offre un servizio di segmentazione dei consumatori chiamato Personicx, che include categorie come "Work & Causes", che nello studio della FTC era descritta come "Persone con bassi guadagni tra i 45-55 anni, che vivono in unità abitative multi familiari", "Rural Parents" e "Metro Strivers", mentre fra i segmenti riguardanti lo status finanziario compaiono "Comfort Zone", "Online Influencers" e "Budget Optimists" [13].

Viene altresì specificato che nel costruire i modelli per l'assegnazione delle categorie finanziarie di Acxiom non vengono utilizzati dati direttamente riguardanti età, sesso, etnia, colore della pelle, religione, origini geografiche e stato coniugale [14].

4. Tipologie di prodotti venduti

I prodotti venduti dai data broker sono di tre tipologie principali: i prodotti di mitigazione dei rischi, ricerca di persone e marketing [8]. Alla prima categoria appartengono prodotti di prevenzione di frodi, come quelli di verifica dell'identità. Questi sono utilizzati per confermare l'identità di un individuo durante lo svolgimento di operazioni per le quali è richiesto accertarsi della stessa, come ad esempio transazioni bancarie.

In un recente articolo, Bloomberg stima a 12.8 miliardi di dollari il valore di questo business, proiettato nel 2024 [15]. Tra i maggiori fornitori di tali servizi, vengono citate le aziende Experian, Equifax, IDMERIT e TransUnion, dove alcune di queste vendono anche prodotti di altre tipologie. IDMERIT fornisce strumenti di verifica dell'email, del numero di telefono, dell'età, che i clienti possono utilizzare per verifiche in tempo reale delle informazioni inserite, anche per adempiere agli obblighi etici e legali in materia *know-your-customer*, come ad esempio i controlli di identità previsti dalla legge anti-terrorismo PATRIOT negli Stati Uniti [8]. Queste informazioni possono essere combinate con quelle ottenute dai prodotti di monitoraggio di biometriche comportamentali, come dinamiche di utilizzo del mouse, di digitazione da tastiera, l'inclinazione del dispositivo, geolocalizzazione, e in alcune situazioni anche riconoscimento vocale e analisi della pressione corporea [16]. Questi prodotti sono indicati per istituti finanziari, piattaforme di scambio di criptovalute, agenzie governative e piattaforme di gioco online. Altri prodotti di rilevamento frodi sono indirizzati al

rilevamento di informazioni false inviate all'interno di form, per esempio riguardanti il proprio reddito [8, 17] o per il rilevamento di utilizzi non autorizzati delle carte bancarie agli sportelli ATM [18].

I prodotti per l'individuazione di persone consentono, con una ricerca, di aggregare le informazioni associate ad un individuo disponibili pubblicamente in milioni di siti web. Alcuni di questi prodotti sono utilizzabili pubblicamente come PeekYou [19], mentre altri sono rivolti ad un uso esclusivamente aziendale. TLOxp è un prodotto di questa categoria offerto da TransUnion. Offre numerose funzionalità e fornisce accesso a numeri di telefono, informazioni anagrafiche, dati riguardanti famiglia, lavoro e istruzione, fascicoli giudiziari, nickname, link a foto e post inseriti su social media, informazioni da siti di incontri e portali di e-commerce [20].

La terza e più redditizia categoria è quella dei prodotti per il marketing, il cui obiettivo è fornire ai clienti dei data broker, la possibilità di raggiungere i consumatori con messaggi pubblicitari mirati. Fra i servizi per il marketing diretto vi sono la vendita di *liste di marketing* (es. una lista di indirizzi di famiglie con bambini in una certa zona geografica) e il *data append*, attività che consente al cliente di integrare i dati di cui l'azienda è in possesso con le informazioni disponibili al data broker. Per esempio l'azienda NextMark vende liste di marketing di consumatori raggruppati in base a certe caratteristiche, in alcuni casi piuttosto sensibili: sono acquistabili liste di persone con disabilità [21], che soffrono di malattie mentali [22] o raggruppate in base ad attributi sulla situazione finanziaria [23] e età [24]. Si tratta di liste che contengono diversi milioni di elementi, la maggioranza provenienti da USA e Canada e reperite da sondaggi e database proprietari.

4.1 Il data onboarding

Una delle pratiche più diffuse finalizzata al marketing online viene definita *onboarding*, e consente di combinare dati online e offline, ovvero non relativi alle attività in rete, per ottenere una più completa conoscenza dei consumatori e rendendo possibile l'utilizzo dei dati per campagne *cross-channel* (glossario) personalizzate. Dati offline possono essere la lista di acquisti effettuata in un centro commerciale o la cronologia della posizione geografica.

L'applicazione più ovvia di questa strategia è la visualizzazione di annunci pubblicitari personalizzati, ma è anche possibile utilizzarla per la personalizzazione dei contenuti, come video e feed di notizie [25, 26].

Un cliente può quindi chiedere al data broker di trovare consumatori con particolari caratteristiche, per esempio indicando a quali segmenti di clienti desidera mostrare pubblicità personalizzata (per esempio "Allergy sufferer") o può fornire la propria lista di clienti a cui vorrebbe mostrare contenuti pubblicitari. Tali dati sono incrociati con quelli a disposizione del data broker fra cui identificatori online e identificativi di dispositivi posseduti dai consumatori (es. PC, tablet, smartphone).

I cookie sono esempi di identificatori online molto utilizzati, ma in realtà meccanismi di tracciamento più sofisticati possono risalire al proprietario di un dispositivo anche se i cookie sono stati cancellati o disabilitati. Sui dispositivi

mobili vengono frequentemente usati gli *id pubblicitari*, forniti dai sistemi operativi mobili per questo preciso scopo, mentre altri identificatori sono l'*indirizzo IP*, l'*indirizzo MAC* (glossario) o molto semplicemente l'identificativo del proprio account se si effettua l'accesso ad un sito [27]. Una volta a conoscenza dei dispositivi utilizzati dai consumatori e in possesso dei loro dati online e offline, il data broker può mostrare contenuti pubblicitari per conto del cliente, e potrebbe fornire le informazioni identificative ai circuiti pubblicitari, solitamente privandole delle *personally identifiable information* (PII), come nome e indirizzo [28].

4.2 Software data-driven nei processi decisionali per il controllo sociale

Uno dei settori applicativi più recenti dell'economia dei dati è quello del *predictive policing*. L'obiettivo è quello di determinare le aree urbane a maggior rischio di criminalità e ottimizzare di conseguenza l'allocazione delle risorse per il controllo del territorio.

Software creati a questo scopo sono IBM Blue Crush [29], HunchLab [30] e PredPol. Quest'ultimo è in grado di effettuare previsioni del crimine su aree ampie 500 x 500 piedi (circa 150 metri) [31]. In particolare il software prevede la tipologia di crimine, data e orario, utilizzando algoritmi di apprendimento automatico [32] e assumendo che reati commessi in un determinato momento abbiano maggiori probabilità di verificarsi nello stesso luogo in futuro, o in luoghi molto vicini [33].

Una recente ricerca ha inoltre analizzato il funzionamento di Harm Assessment Risk Tool, un prodotto di *valutazione del rischio* sviluppato dall'Università di Cambridge in collaborazione con il dipartimento di polizia di Durham per supportare la polizia nelle decisioni di custodia [34].

HART utilizza un algoritmo di apprendimento automatico chiamato *random forest* che combina le predizioni di 509 *alberi di decisione* [35]. Un albero di decisione classifica un input sulla base della validità di una serie di regole if-then. Ogni nodo dell'albero verifica la validità di condizioni su un attributo dell'input, e ogni arco uscente da un nodo rappresenta un possibile valore dell'attributo (ad esempio per l'attributo "genere" i valori possono essere "maschio" o "femmina", per l'età si possono utilizzare intervalli di valori come "0-18", "19-30" e così via). Dalla radice, ogni esempio è classificato sulla base degli attributi, fino a raggiungere le foglie dell'albero, che fornisce in output la predizione finale per quell'esempio. Il modello utilizzato in HART classifica una persona colpevole di reato sulla base della probabilità che ne commetta un altro, assegnando una di 3 possibili etichette: "high risk" se prevede che la stessa persona commetta un nuovo crimine grave nei successivi 2 anni, "moderate risk" se prevede un crimine non grave nei successivi 2 anni e "low risk" se non prevede crimini nei prossimi 2 anni.
Nella tabella 1 sono indicati alcuni degli attributi utilizzati in HART per produrre la classificazione finale.

Attributo	Descrizione
CustodyAge	Età del soggetto alla commissione del reato
Gender	Maschio o femmina
InstantAnyOffenceCount	Conteggio dei crimini commessi fino al presente
InstantViolenceOffenceBinary	Un valore binario per indicare se il crimine ha natura violenta
InstantPropertyOffenceBinary	Un valore binario per indicare se si tratta di reato di proprietà
CustodyPostcodeOutwardTop24	I primi caratteri del codice postale dell'incriminato
CustodyMosaicCodeTop28	Uno dei 28 codici che descrive le caratteristiche socio-geo demografiche più comuni nella Contea di Durham
FirstAnyOffenceAge	L'età del soggetto al primo reato commesso

Tabella 1
Alcuni attributi utilizzati all'interno di HART

In particolare, secondo Big Brother Watch, l'attributo "CustodyMosaicCodeTop28" mostrato in tabella si riferisce ai dati forniti da Mosaic [34], un prodotto di Experian per la segmentazione del mercato che "classifica individui e famiglie in gruppi e tipologie dettagliate permettendo di raggiungerli con comunicazioni rilevanti" [36]. Le informazioni socio-demografiche generate da Mosaic includono categorie come "Disconnected Youth" [34], "Platinum Prosperity", "Tough Times" e molte altre [37].

5. Conclusioni

L'utilizzo di strumenti di predictive policing e di valutazione automatica del rischio solleva alcuni problemi di natura etica, riguardo alla capacità e all'opportunità di effettuare scelte così delicate tramite algoritmi, il cui funzionamento rimane in parte opaco.

Nonostante in Europa il regolamento GDPR vieti l'utilizzo non consensuale di questi strumenti per decisioni esclusivamente automatizzate che incidano significativamente sulla vita di persone [2, art. 22], resta comunque la possibilità che decisori umani siano influenzati dalle criticità di tali metodologie.

Talvolta vi è la errata percezione che un algoritmo ben progettato fornisca autonomamente una soluzione corretta e imparziale. Al contrario, il risultato ottenuto dipende in larga parte dalle capacità tecniche del personale, sia nella definizione dell'algoritmo, che nella conoscenza del dominio applicativo necessaria per una preparazione dei dati che garantisca risultati significativi e non ambigui [38].

Innanzitutto un algoritmo di apprendimento automatico non può considerare tutte le possibili relazioni fra i dati, ma ha bisogno di *assunzioni*, indicate con il nome di *bias algoritmico*, senza il quale l'apprendimento non sarebbe possibile. Quindi necessariamente l'algoritmo sarà sbilanciato a favore di alcune *ipotesi* (glossario) piuttosto che altre.

In secondo luogo, l'algoritmo impara una relazione presente all'interno dei dati che gli sono forniti durante la fase di allenamento, nella quale apprende una rappresentazione di essi utile a fornire output corretti. Se però all'interno di questi dati è incorporato uno stereotipo sociale discriminatorio, l'algoritmo potrebbe imparare a riprodurlo. A titolo di esempio, consideriamo un software per effettuare una pre-selezione dei curriculum vitae inviati ad una azienda per aggiudicarsi un posto di lavoro, e supponiamo che sia stato allenato sui dati dei dipendenti migliori dell'azienda, in modo da garantire una priorità di colloquio più alta ai candidati con maggiore affinità. Se i dipendenti aziendali sono per il 90% uomini, l'algoritmo potrebbe imparare un'ipotesi che discrimina fortemente i candidati di sesso femminile. In questo caso l'algoritmo ha individuato una *correlazione* fra individui di sesso maschile e migliori dipendenti e ha inferito che con alta probabilità il candidato ideale deve essere maschio, sebbene in realtà non ci sia alcun rapporto di *causalità* fra i due attributi.

Allo stesso modo nei prodotti di policing come PredPol, l'utilizzo di attributi quali il codice postale o il quartiere di residenza potrebbe alimentare un circolo vizioso, dove persone che vivono in aree ad "alto rischio" sono sottoposte ad un controllo più accurato, portando alla luce un maggior numero di reati rispetto ad altre aree [34], rinforzando quindi il condizionamento esistente. Andrew Ferguson, docente di legge all'Università di Washington D.C., sottolinea che mancano prove e ricerca scientifica sull'efficienza e l'efficacia dei sistemi di predictive policing, aggiungendo che è necessaria una validazione esterna di questi strumenti per comprendere il significato e la valenza della metodologia [31].

Più in generale, un rischio della raccolta intensiva e dell'utilizzo non regolato dei dati personali è la crescita smisurata dell'importanza della *digital footprint*, ovvero l'identità digitale di ciascuna persona. Questo può condurre all'adesione della società ad una cultura della conformità, nella quale gli individui, consapevoli del monitoraggio e dell'importanza del loro comportamento, lo modificano in funzione di quello considerato più corretto per i propri obiettivi. Per esempio è stato riportato che diverse università americane controllano i profili social dei candidati per decidere se accettarli [39]. Questo può essere un incentivo, per i futuri studenti, a cambiare le proprie abitudini online per apparire conformi al profilo considerato "migliore" per uno studente.

Tijmen Schep, un *privacy designer* e critico tecnologico, definisce questo effetto *social cooling*, che promuove l'abitudine di avversione al rischio e una maggiore rigidità e uniformità sociale, che soffoca le qualità individuali [40]. Questa tendenza è dovuta anche al trattamento statistico dei dati, che evidenzia il comportamento della maggioranza, ma penalizza le divergenze comportamentali, che diventano anomalie, più difficili da comprendere e da gestire.

Va tuttavia sottolineato che l'economia dei dati e i prodotti *data-driven* non hanno solo conseguenze negative: un prodotto di prevenzione di frodi potrebbe segnalare una transazione in corso come illegittima e bloccare un tentativo di furto di denaro, mentre la ricezione di un'offerta pubblicitaria personalizzata potrebbe non essere sempre sgradita. D'altro canto è necessario lo sviluppo di una maggior consapevolezza da parte dei consumatori del valore dei propri dati e la promozione di una sensibilità da parte delle aziende nel garantire un utilizzo corretto e sostenibile di queste risorse.

Glossario

Analitiche
Software per l'analisi dei dati che utilizzano modelli matematici al fine di effettuare predizioni, ottenere nuove informazioni e generare raccomandazioni. Strumenti di analitica possono analizzare i dati utilizzando tecniche di data mining, machine learning, pattern matching, sentiment analysis, analisi di reti e grafi, simulazione e reti neurali. [41, 42]

Apprendimento non supervisionato
Paradigma di apprendimento automatico, dove, dato un insieme di esempi, l'obiettivo è trovare regolarità e pattern che siano veri sull'intero dominio. In questo caso gli esempi non devono essere pre-classificati da un esperto umano, quindi in questo senso non è necessaria supervisione.

Apprendimento supervisionato
Paradigma di apprendimento automatico, nel quale un algoritmo, dato un insieme di dati pre-classificati, apprende una descrizione generale degli stessi che cattura il contenuto informativo utile per effettuare predizioni o elaborazioni su nuovi dati.

Targeting cross-channel
L'insieme di pratiche con cui è possibile raggiungere un particolare consumatore con messaggi di marketing personalizzati su tutti i suoi dispositivi (es. smartphone, tablet, PC) e servendosi di diversi canali (es. email, telefono, pubblicità online). [43]

Ipotesi (machine learning)
Termine con cui è indicata una funzione che un algoritmo di machine learning è in grado di apprendere. L'insieme di funzioni che l'algoritmo può imparare è definito *spazio delle ipotesi*. [44]

Richiesta HTTP
Una richiesta di informazioni inviata da un client (come un web browser) ad un server utilizzando il protocollo HTTP (Hyper Text Transfer Protocol). [45]

Indirizzo IP
Etichetta numerica assegnata ad ogni dispositivo connesso ad una rete di computer che utilizza il protocollo IP (Internet Protocol) per la comunicazione e lo scambio di informazioni. [46]

Indirizzo MAC

Il Media Access Control address (o indirizzo MAC) è un identificatore univoco assegnato ad un dispositivo connesso ad una rete ed è spesso è assegnato dal produttore dell'hardware. [47]

Bibliografia

[1] Press, G. (2017). 6 Predictions For The $203 Billion Big Data Analytics Market, *Forbes*, https://www.forbes.com/sites/gilpress/2017/01/20/6-predictions-for-the-203-billion-big-data-analytics-market/#251c7ce32083 (ultimo accesso ottobre 2019).

[2] (2016). "Regolamento (UE) 2016/679 del Parlamento Europeo e del Consiglio", *Gazzetta ufficiale dell'Unione europea*.

[3] https://protezionedatipersonali.it/dato-personale (ultimo accesso ottobre 2019).

[4] https://www.indicative.com/blog/what-is-behavioral-data-and-behavioral-analytics/ (ultimo accesso ottobre 2019).

[5] https://conversionxl.com/conversion-optimization/mouse-tracking-and-heat-maps/ (ultimo accesso ottobre 2019).

[6] https://www.aboutamazon.it/innovazioni/amazon-go (ultimo accesso ottobre 2019).

[7] Naughton, J. (2019). "'The goal is to automate us': welcome to the age of surveillance capitalism", *The Guardian*, https://www.theguardian.com/technology/2019/jan/20/shoshana-zuboff-age-of-surveillance-capitalism-google-facebook (ultimo accesso ottobre 2019).

[8] Ramirez, E., Brill, J., Ohlhausen, M.K., D. Wright, J., McSweeny, T. (2014). "Data Brokers A Call for Transparency and Accountability", *Federal Trade Commission report*.

[9] https://www.idmerit.com/global-coverage/ (ultimo accesso ottobre 2019).

[10] https://www.acxiom.com/what-we-do/data/ (ultimo accesso ottobre 2019).

[11] https://developer.myacxiom.com/code/api/data-bundles/main (ultimo accesso ottobre 2019).

[12] https://developer.myacxiom.com/code/api/endpoints/portrait (ultimo accesso ottobre 2019).

[13] https://developer.myacxiom.com/code/api/data-bundles/level1bundle/demographics (ultimo accesso ottobre 2019).

[14] https://developer.myacxiom.com/code/api/data-bundles/bundle/personicxFinancial (ultimo accesso ottobre 2019).

[15] https://www.bloomberg.com/press-releases/2019-05-14/identity-verification-market-worth-12-8-billion-by-2024-exclusive-report-by-marketsandmarkets (ultimo accesso ottobre 2019).

[16] https://www.idmerit.com/behavior-monitoring/ (ultimo accesso ottobre 2019).

[17] https://www.dnb.com/resources/what-is-data-validation.html
(ultimo accesso ottobre 2019).

[18] Rahman, M.M., Saha, A.R. (2019). "A Comparative Study and Performance Analysis of ATM Card Fraud Detection Techniques", *Journal of Information Security*, 10, 188-197, https://doi.org/10.4236/jis.2019.103011.

[19] https://www.peekyou.com/ (ultimo accesso ottobre 2019).

[20] https://www.tlo.com/social-media (ultimo accesso ottobre 2019).

[21] https://lists.nextmark.com/market?page=order/online/datacard&id=189752
(ultimo accesso ottobre 2019).

[22] https://lists.nextmark.com/market?page=order/online/datacard&id=243005
(ultimo accesso ottobre 2019).

[23] https://lists.nextmark.com/market?page=order/online/datacard&id=142394
(ultimo accesso ottobre 2019).

[24] https://lists.nextmark.com/market?page=order/online/datacard&id=315873
(ultimo accesso ottobre 2019).

[25] https://liveramp.com/blog/customer-data-onboarding/ (ultimo accesso ottobre 2019).

[26] https://www.braze.com/docs/help/best_practices/news_feed/
(ultimo accesso ottobre 2019).

[27] https://ico.org.uk/for-organisations/guide-to-data-protection/guide-to-the-general-data-protection-regulation-gdpr/what-is-personal-data/what-are-identifiers-and-related-factors/ (ultimo accesso ottobre 2019).

[28] https://www.lotame.com/back-basics-data-onboarding/
(ultimo accesso ottobre 2019).

[29] https://www.ibm.com/ibm/history/ibm100/us/en/icons/crimefighting/transform/
(ultimo accesso ottobre 2019).

[30] https://www.hunchlab.com/ (ultimo accesso ottobre 2019).

[31] Haskins, C. (2019). "Dozens of Cities Have Secretly Experimented With Predictive Policing Software", *Vice*, https://www.vice.com/en_us/article/d3m7jq/dozens-of-cities-have-secretly-experimented-with-predictive-policing-software
(ultimo accesso ottobre 2019).

[32] https://www.predpol.com/about/ (ultimo accesso ottobre 2019).

[33] https://www.predpol.com/technology/ (ultimo accesso ottobre 2019).

[34] Scantamburlo, T., Charlesworth, A., Cristianini, N. (2019). "Machine Decisions and Human Consequences", *University of Bristol*.

[35] Oswald, M., Grace, J., Urwin, S., C. Barnes, G. (2018). "Algorithmic risk assessment policing models: lessons from the Durham HART model and 'Experimental' proportionality", *Information & Communications Technology Law*, 27, 223-250.

[36] https://www.experian.co.uk/business/marketing/segmentation-targeting/mosaic/ (ultimo accesso ottobre 2019).

[37] https://www.experian.com/assets/marketing-services/brochures/mosaic-brochure.pdf (ultimo accesso ottobre 2019).

[38] Tsiptsis, K., Chorianopoulos, A. (2011). Data Mining Techniques in CRM: Inside Customer Segmentation, John Wiley & Sons.

[39] Moody, J. (2019). "Why Colleges Look at Students' Social Media", *U.S. News & World Report*, https://www.usnews.com/education/best-colleges/articles/2019-08-22/why-colleges-look-at-students-social-media-accounts (ultimo accesso ottobre 2019).

[40] https://www.socialcooling.com/ (ultimo accesso ottobre 2019).

[41] https://dictionary.cambridge.org/dictionary/english/analytics (ultimo accesso ottobre 2019).

[42] https://www.gartner.com/it-glossary/advanced-analytics/ (ultimo accesso ottobre 2019).

[43] https://www.oracle.com/marketingcloud/resources/cross-channel-marketing.html (ultimo accesso ottobre 2019).

[44] Mitchell, T. M. (1997). *Machine Learning*, McGraw-Hill Education.

[45] https://developer.mozilla.org/en-US/docs/Web/HTTP/Messages (ultimo accesso ottobre 2019).

[46] https://en.wikipedia.org/wiki/IP_address (ultimo accesso ottobre 2019).

[47] https://en.wikipedia.org/wiki/MAC_address (ultimo accesso ottobre 2019).

Biografie

Silvia Crafa è ricercatrice confermata presso il Dipartimento di Matematica dell'Università di Padova. La sua attività di ricerca in ambito informatico si focalizza sui metodi formali per l'analisi dei sistemi concorrenti e dei linguaggi di programmazione. Studia inoltre l'impatto sociale delle tecnologie digitali in ottica interdisciplinare. È autrice di numerose pubblicazioni su prestigiose riviste internazionali e collabora con diverse istituzioni straniere. È membro del Laboratorio Nazionale CINI su Informatica e Società ed è stato membro del working group Informatics Europe e EU-ACM per la definizione di un libro bianco sugli algoritmi di decisione automatica.

Email: silvia.crafa@unipd.it

Alessandro Zangari è studente del Corso di Laurea Magistrale in Informatica presso l'Università degli Studi di Padova. Ha conseguito la laurea triennale in Informatica nel 2018 e ha proseguito gli studi interessandosi all'ambito dell'Intelligenza Artificiale, del Machine Learning e di come queste tecnologie influenzino la vita umana.

Email: alessandro.zangari@studenti.unipd.it

Ricordo di Giulio Occhini

Franco Filippazzi, Gustavo Canti

Il 17 novembre scorso è mancato Giulio Occhini, a cui si deve la nascita di questa rivista.

Ma Mondo Digitale è soltanto una delle sue tante iniziative. Egli infatti ha dato, in molteplici modi, contributi fondamentali allo sviluppo dell'informatica e della cultura digitale in Italia.

Ha svolto infatti ruoli manageriali in importanti aziende italiane del settore, ha tenuto corsi in ambito accademico, è autore di molti articoli e libri sull'informatica.

Oltre a ciò, ha ricoperto cariche direttive in organizzazioni culturali nazionali e internazionali. E' stato infatti Presidente di AICA, l'Associazione di riferimento del settore informatico in Italia, e Presidente del CEPIS, l'Ente che federa tutte le società europee di informatica.

Negli anni '90, durante la sua presidenza del CEPIS, è stata introdotta l'ECDL, la certificazione internazionale delle competenze digitali, che ha avuto grande diffusione in tutto il mondo, inclusa l'Italia.

A parte i grandi meriti professionali e culturali, Giulio è stato una persona di straordinarie doti umane. Integrità, gentilezza, generosità hanno caratterizzato il suo modo di vivere.

Per gli autori di questo breve ricordo la scomparsa di Giulio è una perdita dolorosa, perché con lui abbiamo avuto per tanti anni rapporti fraterni di amicizia e interessi comuni, dalle camminate in montagna alle discussioni filosofiche. Oltre che provetto alpinista, da giovane amava percorrere le strade di montagna anche in bicicletta. Era amante della lettura (i libri riempivano ampi spazi della sua casa, che definiva una biblioteca con servizi), ma anche di musica classica che ascoltava con gli amici a concerti e a casa. Aveva un ampio senso dell'umorismo, ma con lui si ragionava e si discuteva anche di cose serie, dal presente informatico al futuro digitale che si prospetta.

Buon viaggio Giulio, amico di una vita.

Due anniversari da ricordare

Silvio Hénin

1959 Il circuito integrato

Correva il 1959, l'anno in cui Salvatore Quasimodo e Emilio Segrè vincevano il premio Nobel, in Inghilterra appariva la piccola utilitaria Mini Minor e negli Stati Uniti la bambola Barbie. Il dittatore Fulgenzio Batista fuggiva da Cuba, lasciando il paese nelle mani di Fidel Castro. Si era nel pieno della Guerra fredda tra USA e URSS ed entrambe le potenze erano impegnate nella corsa allo spazio per motivi più militari che scientifici. Per il momento, con la messa in orbita dello Sputnik 1 due anni prima, l'Unione Sovietica sembrava in vantaggio. Quell'anno successe anche un'altra cosa, forse meno eclatante, ma non meno importante per gli effetti che avrà in moltissimi settori: nei primi mesi dell'anno al *Patent Office* degli Stati Uniti furono depositati alcuni brevetti per un nuovo dispositivo elettronico, il *circuito integrato*, frutto di due invenzioni indipendenti avvenute nell'anno precedente.

L'origine di queste invenzioni fu dovuta proprio a quella guerra non combattuta e alla corsa allo spazio. Da tempo si cercava infatti di installare complessi dispositivi elettronici, anche interi computer, all'interno di aerei, missili, siluri, satelliti artificiali e sonde spaziali, ma i problemi da risolvere erano molti, soprattutto il volume, il peso e il consumo energetico di tali applicazioni. Anche se, ormai da anni, si stava passando dall'uso delle vecchie valvole termoioniche, fragili, voluminose e assetate di energia, a quello dei minuscoli transistor, ancora imperava la cosiddetta *tirannia dei numeri*: usando singoli componenti discreti (transistor, diodi, resistenze e condensatori) la quantità e il volume dei collegamenti tra essi superava di molto quella dei componenti stessi, impedendo un'ulteriore miniaturizzazione, aumentando i rischi di malfunzionamento e limitando la velocità di elaborazione.[1] In ambito militare e spaziale furono quindi varati negli USA alcuni costosissimi progetti di ricerca per risolvere il problema. Le tre forze armate, Esercito, Aeronautica e Marina, investirono indipendentemente somme rilevanti in progetti chiamati Tinkertoy, Micro-moduli e il visionario, quanto vago, concetto di 'elettronica molecolare', che costò all' *US Air Force* circa 5 milioni di dollari senza produrre alcunché di utilizzabile.[2] Fin dal 1949, Jack Morton dei Bell Laboratories aveva immaginato "una tecnica con cui i fili di collegamento e gli elementi passivi [resistenze, condensatori] possano essere stampati in un processo continuo risparmiando dimensioni e costi". Il concetto fu riproposto tre anni dopo dall'inglese Geoffrey Dummer che esibì il modello di un microcircuito, ma senza riscuotere grande

interesse. Anche Harwick Johnson della RCA depositò nel 1953 un brevetto di circuito integrato, che non fu preso in considerazione dalla nota azienda. Furono due americani, Jack Kilby e Robert Noyce, a realizzare i primi prototipi funzionanti della nuova tecnologia.

Figura 1
Robert Noyce (sinistra) e Jack Kilby (destra). (Elektrokits.ro)

Nel luglio 1958 Jack S. Kilby era stato assunto temporaneamente alla Texas Instruments proprio per sviluppare la tecnica dei Micro-moduli, ma l'idea non lo soddisfaceva e, all'oscuro dei tentativi di Dummer e Johnson, decise di realizzare un intero circuito in un solo blocco di germanio in cui si sarebbero materializzati tutti i componenti necessari. Il suo primo prototipo, un oscillatore con un solo transistor, tre resistenze e tre condensatori, vide la luce nel settembre di quell'anno. Nel successivo febbraio, la Texas Instruments depositò la domanda di brevetto, ben conscia dell'importanza del progetto e in ansia per le voci che un'altra azienda stava puntando allo stesso obiettivo. L'altra azienda era la Fairchild, dove Robert N. Noyce era pervenuto all'idea di circuito integrato come naturale conseguenza di un nuovo modo di produrre transistor, la tecnologia detta 'planare'. Noyce immaginò che la stessa tecnica avrebbe permesso di costruire un intero circuito nel blocchetto di silicio. Entusiasta dell'idea quanto lo era la Texas Instruments, la Fairchild depositò il brevetto un mese dopo quello di Kilby. L'Ufficio brevetti però assegnò l'esame delle due proposte a due esperti diversi e successe che il primo a passare la revisione fu proprio quello di Noyce. I due progetti non erano uguali: mentre Kilby faceva uso di sottilissimi conduttori volanti per collegare i diversi componenti, il disegno di Noyce prevedeva di realizzare nel blocco di silicio anche i conduttori. Sarà quindi il modello della Fairchild che si imporrà per la facilità produttiva e il basso costo. La strana situazione dei brevetti era però insostenibile per le due aziende e arrivò in tribunale, dando avvio a un procedimento che durò undici anni e, dopo appelli, contro-appelli e sentenze che davano alternativamente ragione all'una o all'altra

parte, la decisione definitiva della Corte suprema fu favorevole alla Fairchild. Per l'invenzione del circuito integrato Kilby ricevette il premio Nobel per la fisica nel 2000, ma l'onore non poté essere condiviso con il collega perché lo statuto della Fondazione Nobel non consente di assegnare premi postumi e Noyce era deceduto da dieci anni.

Il circuito integrato non riscosse immediato consenso nel mondo dell'elettronica. I costruttori di radio, di televisori, di computer e di tutta la strumentazione elettronica analogica e digitale non vedevano di buon occhio l'acquisto di interi circuiti progettati da altri. Non minore fu la preoccupazione delle aziende che fornivano componenti discreti e che vedevano il loro mercato in grave pericolo. Unica eccezione furono i produttori di apparecchi acustici. I primi grandi consumatori di circuiti integrati furono quindi i programmi militari e spaziali, che non potevano farne a meno e che assorbirono gran parte dell'intera produzione, supportando così la nascente industria fino a quando i costi non scesero abbastanza da conquistare il mercato civile. Il primo computer a circuiti integrati fu realizzato nel 1961 dalla Texas Instruments per l'aeronautica USA e nel 1963 questi dispositivi equipaggiarono il satellite IMP della NASA. Tre anni dopo il modulo lunare e il modulo di comando delle missioni Apollo furono dotati di piccoli computer a circuiti integrati, gli Apollo Guidance Computer (AGC), per gestire la navigazione dei due veicoli spaziali. Per i computer commerciali e per uso civile l'industria informatica fu più prudente, la stessa IBM scelse di non adottare i circuiti integrati per la sua innovativa famiglia di mainframe S/360 e utilizzò invece una propria tecnologia ibrida, la Solid Logic Technology (SLT). Finalmente, nel 1971, apparve lo Intel 4004, un intero processore su un solo chip di silicio dovuto all'ingegno dell'italiano Federico Faggin. La strada per l'informatica moderna, diffusa, economica e alla portata di tutti, era iniziata.

1969 ARPANET

Nel 1969 molte cose erano cambiate da dieci anni prima, i computer non erano più una novità, ma la Guerra fredda non accennava a terminare e un suo sottoprodotto, la Guerra del Vietnam, era in pieno corso. Oltre la Cortina di ferro, a Praga, Jan Palach si dava fuoco per protestare contro l'occupazione sovietica. Volava il primo aereo civile supersonico, il Concorde, Nixon diventava presidente degli USA e il rock impazzava a Woodstock. Il motivo per cui oggi ricordiamo il 1969 è però un evento molto più importante: il 20 luglio il LEM della missione Apollo 11 della NASA posava i suoi quattro piedi sulla superficie della Luna, permettendo a due esseri umani, Neil Armstrong e Buzz Aldrin, di lasciare le loro impronte sul satellite e ritornare sani e salvi dal collega Michael Collins, rimasto in orbita lunare ad aspettarli. Ma ciò che pochi sanno è che Il 29 ottobre avvenne qualcosa che passò del tutto inosservato, ma che cambierà il nostro mondo e le nostre abitudini in modo molto più incisivo dell'allunaggio: un messaggio digitale viaggiò da un computer della University of California a Los Angeles (UCLA) a un altro dello Stanford Research Institute (SRI), mettendoli in collegamento. Era nata ARPANET, una rete dati di nuova concezione che aprirà la strada per Internet.

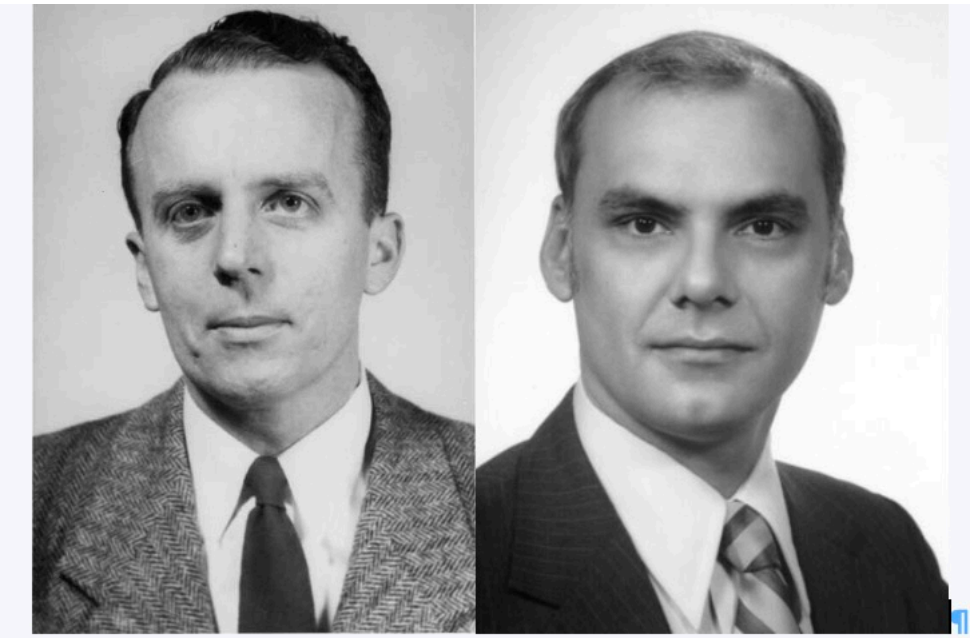

Figura 2
J.C.R. Licklider (sinistra) e Robert Taylor (destra). (history-computer.com)

Anche ARPANET, come il circuito integrato, fu frutto della Guerra fredda, come testimonia il suo nome. ARPA (oggi DARPA) è infatti l'acronimo di Advanced Research Project Agency (Agenzia per i progetti di ricerca avanzata), un ente statunitense voluto dal presidente Dwight D. Eisenhower nel 1958 per incentivare la ricerca nei settori di punta della scienza e della tecnologia, a fini potenzialmente militari. Ciò che spinse Eisenhower fu, anche qui, la preoccupazione che l'Unione Sovietica disponesse di risorse tecnologiche più avanzate degli USA. Tra i settori di intervento dell'agenzia vi era infatti anche quello spaziale, che fu subito dopo riassegnato alla neonata NASA. Il motto di ARPA era "Cercare persone con idee, finanziarle in modo generoso, assumersi il rischio di risultati imprevedibili, fare tutto il necessario, ma farlo!". Tra le sezioni dell'ARPA vi era lo IPTO (Information Processing Techniques Office) che nel decennio successivo finanzierà molti progetti informatici con contributi che saliranno da 9 a 23 milioni di dollari all'anno (160 milioni di oggi). I fondi erano concessi con grande liberalità e scarso controllo e ai ricercatori era lasciata una notevole libertà di iniziativa. Tra i settori di intervento vi furono lo sviluppo dei sistemi in *time-sharing*, dell'intelligenza artificiale, delle interfacce uomo-macchina, della *computer-graphic* e, appunto, delle reti di calcolatori. A capo dello IPTO fu posto uno psicologo convertitosi all'informatica, Joseph C. R. LIcklider, che nel 1963 propose la sua visione di una rete informatica in uno scritto intitolato "Memorandum per i membri e gli affiliati della rete intergalattica" dove proponeva un sistema di collegamenti che permettesse di utilizzare tutte le risorse di calcolo dislocate sul territorio americano, utilizzando un linguaggio comune. Quattro anni dopo, il nuovo direttore dello IPTO, Robert Taylor, ottenne

un finanziamento di un milione di dollari per "[…] soddisfare le esigenze di comando e controllo contro le minacce nucleari e ottenere un controllo delle forze nucleari statunitensi in grado di sopravvivere a un attacco […]."[3] L'obiettivo sembra piuttosto diverso da quelli di Licklider e di Taylor, ma serviva a convincere il Ministero della difesa a concedere le risorse necessarie.

Nello IPTO (ora diretto da Robert Taylor) fu creato un gruppo di lavoro guidato da Larry Roberts e composto da Paul Baran, Leonard Kleinrock e Wesley Clark. Si decise di ricorrere, per i collegamenti, alla tecnologia a 'commutazione di pacchetto', proposta da Paul Baran negli USA e, indipendentemente, da Donald W. Davies in Inghilterra. La realizzazione pratica della rete fu affidata alla Bolt, Beranek and Newman Inc., un'azienda specializzata in psicoacustica, già fornitore delle forze armate. Il progetto decollò e, in soli due anni, fu pronto un piccolo prototipo. Il 29 ottobre 1969, alle ore 22:30 (ora locale), il dottorando Charley Kline provò a trasmettere un comando dalla tastiera del computer SDS Sigma 7 del'UCLA diretto allo SDS 940 dello SRI, a 500 Km di distanza. Il comando avrebbe dovuto essere il banale "LOGIN", ma dopo le prime due lettere il Sigma 7 andò in *crash*.[4] Charley ci riprovò un'ora più tardi, questa volta con successo. Il primo collegamento permanente fu stabilito il 21 novembre e pochi giorni dopo l'intera rete era in funzione. All'epoca i nodi di ARPANET erano solo quattro, oltre a UCLA e SRI vi erano l'Università della California a Santa Barbara (UCSB) e l'Università dello Utah. Nel 1975 ARPANET fu dichiarata operativa, con 57 nodi che connettevano molti centri di ricerca americani, un nodo in Norvegia e uno a Londra, questi ultimi connessi via satellite. L'uso della rete, diversamente dalle intenzioni per cui fu creata, consistette quasi solo di messaggi (*email*) e scambio di documenti (*file transfer*), che resteranno le *killer application* anche di Internet. ARPANET non può, a rigore, essere considerata una versione prototipale di Internet — quest'ultima non è una rete, ma una rete di reti, un concetto all'epoca ancora embrionale, e per ARPANET era inimmaginabile un uso esteso al pubblico — ma permise di sperimentare un concetto, acquisire esperienza e formare una rete di competenze senza i quali Internet non sarebbe esistita.

Bibliografia

1.M. Riordan, L. Hoddeson , *Crystal Fire*, Norton, 1998.

2.D. Brock, D. Laws, "Early History of Microcircuitry, An Overview", *IEEE Ann. Hist. Comp*. 34(1), 2012.

3.J. C. R. Licklider, R. W. Taylor, "The Computer as a Communication Device", *Science and Technology,* April, 1968.

4.J. Abbate, *Inventing the Internet*, Inside Technology, 1999.