9. Agenti nei sistemi socio-tecnici

di *Matteo Baldoni, Cristina Baroglio, Stefano Tedeschi*

1. Introduzione

I sistemi socio-tecnici sono da tempo oggetto di studio dell'informatica. Molti sistemi informatici hanno il fine di consentire a utilizzatori umani di collaborare e coordinarsi senza la necessità di conoscersi o di interagire direttamente. Si pensi, per esempio, ai portali di acquisto dei negozi online e ai sistemi di gestione degli appelli di esame delle università: la segreteria pubblica le date per i vari corsi, chi lo desidera si iscrive e gli elenchi sono notificati ai/alle docenti che organizzano gli esami e verbalizzano gli esiti. Allo stesso modo fornitori, clienti e spedizionieri si coordinano, senza incontrarsi nel mondo fisico, in procedure di compravendita. Inoltre la letteratura sui sistemi socio-tecnici ha ispirato il modo in cui il software stesso è organizzato. Un esempio è il modello dell'organizzazione del lavoro sviluppato negli anni '50 dal Tavistock Institute of Human Relations, che prevede fra le altre cose la distribuzione delle responsabilità e il raggruppamento degli impiegati in team (Trist, 1981). Tali principi si collegano direttamente ad alcuni importanti principi di progettazione del software tradizionali, quali separazione delle responsabilità, modularità e componibilità, ma l'idea più promettente (Sommerville *et al.*, 2012) è centrare la modellazione dei sistemi software complessi sul concetto di *interazione* fra componenti, vedendo proprio nella specifica e nella regolamentazione di tali interazioni un pilastro fondamentale del progetto.

Ecco allora che il sistema socio-tecnico potrà essere immaginato come un insieme di *agenti autonomi* – che in alcuni casi corrisponderanno a utenti umani e in altri a esecuzioni software, che utilizzano le risorse fornite dal sistema e l'interazione con gli altri agenti per perseguire *obiettivi* propri o condivisi. Questo approccio è particolarmente adatto a gestire sistemi aperti, in cui gli agenti possono entrare e uscire dinamicamente,

e dove l'esecuzione è distribuita (anche geograficamente) su più unità di processamento. Per esempio, due agenti acquirenti interessati a prenotare un posto al cinema saranno aiutati dal sistema a evitare di prenotare lo stesso posto; uno dei due potrebbe essere una persona e l'altro un agente artificiale a cui una seconda persona ha delegato il compito di effettuare la prenotazione. Anche il cinema potrebbe essere visto come un agente che offre e coordina un servizio di prenotazione. Un sistema socio-tecnico può anche aiutare a distribuire compiti in maniera più o meno esplicita. Per esempio, i docenti di un corso ricevono l'incarico di esaminare determinati studenti sulla base delle iscrizioni di questi ultimi, senza che vi sia l'operato esplicito di una segreteria né per gestire le iscrizioni né per comunicare ai docenti le liste e le aule, ma solo la presenza di un sistema software per la gestione degli esami.

Questo capitolo introduce i concetti principali sottesi dalla visione ad agenti dei sistemi socio-tecnici. Comincia con la definizione stessa di cosa si intenda per agente (Sezione 2) per poi passare all'approccio sociale alla realizzazione di sistemi ad agenti (Sezione 3). Vedremo quindi la nozione di artefatto e il suo utilizzo (Sezione 4). Parleremo di accountability (Sezione 5) per chiudere con uno sguardo all'utilizzo di intelligenza artificiale generativa e all'impatto di tali tecnologie sui sistemi socio-tecnici (Sezione 6).

2. Concetto di agente

Nell'ambito dell'intelligenza artificiale il concetto di *agente* rappresenta un'astrazione molto importante, che viene utilizzata sia nel contesto del software engineering sia per realizzare sistemi distribuiti costituiti da parti interagenti. Lo studio di cosa sia un agente, di quali strumenti formali permettano di rappresentare gli agenti e quali regole di inferenza permettano di automatizzare il ragionamento sugli agenti è noto come *Agent Theory*. Le proposte di questa disciplina sono astratte, hanno una natura formale-matematica. La definizione dei meccanismi e degli strumenti (compresi linguaggi di programmazione) che rendono operative tali teorie sono invece oggetto dell'area di ricerca nota come *Agent-Oriented Software Engineering.*

Le caratteristiche principali che definiscono un agente (Wooldridge, 2009) sono sostanzialmente quattro. Un agente è *autonomo* ovvero ha il controllo sulle proprie azioni e sul proprio stato interno. Ha delle *abilità sociali*, cioè, è in grado di interagire con altri agenti. È *reattivo* nel senso che percepisce e risponde a variazioni del contesto in cui opera (detto *ambiente*). È *proattivo*, cioè oltre a reagire alle contingenze esibisce un comportamento guidato da obiettivi, prendendo l'iniziativa ogni volta che lo

ritiene utile. Un legame indissolubile lega un agente e il suo ambiente, che può essere un ambiente fisico oppure software.

Per quanto riguarda il funzionamento, un agente può essere immaginato come un'entità che esegue all'infinito un ciclo composto di tre passi (Russell, Norvig, 2003). A ogni iterazione, l'agente *percepisce* l'informazione rilevante dell'ambiente, *delibera* se e quale azione eseguire, *applica* tale azione e quindi ricomincia. Questo altissimo livello di astrazione sottende infinite implementazioni. In alcuni casi la percezione, che è un'attività complessa che richiede l'elaborazione di un flusso di dati, può appoggiarsi a dei sensori fisici (si pensi a un sistema domotico che deve regolare l'ombreggiatura di un edificio); in altri casi corrisponde all'accesso a file o basi di dati (per esempio la lettura delle analisi cliniche di un paziente). La deliberazione può andare dall'uso di una sequenza rigida di regole *se-allora* a un motore inferenziale o una rete neurale. L'attuazione può corrispondere a un'azione nell'ambiente fisico (apertura delle tende) quanto alla produzione di una risposta o una conclusione (il paziente non presenta segni di malattia). La varietà è davvero amplissima.

L'astrazione di agente risulta utile anche nella progettazione di sistemi socio-tecnici perché con una sola astrazione risulta possibile catturare sia le componenti software sia gli utenti umani. Anche le persone percepiscono, deliberano e agiscono, sono reattive, proattive, autonome e hanno capacità sociali. Lo fanno con meccanismi diversi dai sistemi artificiali ma tali differenze non sono centrali nella realizzazione dei sistemi socio-tecnici, dove piuttosto conta quali azioni i sistemi mettono nella disponibilità di quali agenti in quali circostanze e momenti. Come per il mondo reale, lo spazio di azione degli agenti è infatti definito dalla situazione specifica considerata. A differenza del mondo reale ogni sistema socio-tecnico nasce per uno scopo, cioè per consentire e guidare lo svolgimento di determinati compiti.

Un sistema socio-tecnico non può quindi essere ridotto alla semplice giustapposizione di un insieme di agenti. Al contrario, richiede la definizione di elementi che permettono alle parti di interagire in maniera funzionale. Da un punto di vista ingegneristico, il sistema socio-tecnico ha uno scopo e le (inter)azioni che consente sono tutte finalizzate a realizzare quello scopo. Così un sistema di supporto alla didattica permetterà a un docente di inserire un voto ma solo a uno/a studente iscritto/a ad un appello valido e solo a partire dalla data in cui l'appello si svolge. Non tutti potranno iscriversi a un appello ma solo le persone che sono studenti, che hanno nel loro percorso quel particolare insegnamento e che non hanno già un voto valido per il medesimo.

L'esempio mostra come nella realizzazione di questi sistemi tornano utili concetti che caratterizzano le organizzazioni umane. Per esempio, il

concetto di ruolo, che una persona (un agente) gioca. Ogni ruolo comporta determinate possibilità di azione: il ruolo docente di accedere alla lista degli iscritti a un appello e di assegnare voti, il ruolo studente di iscriversi a un appello e leggere il proprio esito. L'esecuzione di alcune azioni potrebbe essere ristretta alle circostanze in cui valgano determinate precondizioni. Per esempio, non si può assegnare un voto a una persona non iscritta a un appello. Parleremo nella prossima sezione di questi aspetti.

Prima è importante sottolineare che, per quanto gli agenti di un sistema socio-tecnico siano guidati da obiettivi, non si richiede che tali obiettivi siano concordi anzi in generale essi potranno essere conflittuali – si pensi ad un bando di gara, i cui partecipanti competono per aggiudicarsi una commessa (FIPA, 2002). Un'ulteriore caratteristica importante è che gli obiettivi sono privati e agli agenti non è richiesto divulgarli. In generale il software che permette l'esistenza del sistema socio-tecnico ha il fine di consentire interazioni significative sia nel caso in cui gli agenti siano fra di loro "cooperanti" sia che siano "competitivi", permettendo a ciascuno di perseguire i propri obiettivi. A questo fine gli agenti, collaborativi o competitivi, dovranno rispettare le regole di interazione gestite dal sistema.

Per completezza occorre evidenziare che l'informatica offre un'ampia gamma di strumenti che abilitano, facilitano e regolamentano l'interazione, che non si basano sul concetto di agente. Si pensi per esempio ai sistemi per la rappresentazione e per la gestione dei processi aziendali, in grado di determinare e tracciare l'esecuzione di attività complesse, che coinvolgono numerosi attori e uffici, con meccanismi affini a quelli dell'esecuzione parallela di programmi scritti nei tradizionali linguaggi di programmazione imperativi, quali sequenze di attività e iterazioni, lettura e salvataggio di dati, scambio di dati. Questo capitolo non scende nel dettaglio di questo vasto settore che riscontra un ampio successo pratico in ambito aziendale. Nel seguito introdurremo invece alcuni approcci e principi studiati nella letteratura dei sistemi e delle organizzazioni multi-agente che traggono ispirazione dal modo in cui gli esseri umani si organizzano per interagire e svolgere attività complesse che nessuno di essi potrebbe svolgere da solo. Per esempio, l'introduzione di norme oppure di meccanismi commitment-trust (impegno e fiducia), di sanzioni, di accountability.

3. Approccio sociale ai sistemi socio-tecnici

Come evidenziato in Whitworth e Ahmad (2013), la realizzazione dei sistemi socio-tecnici richiede un'evoluzione del modo in cui si immagina la computazione, passando dalla computazione tradizionale a un approccio

sociale (Dalpiaz, Chopra, Lim, 2011), da un insieme di singole esecuzioni a una in cui i concetti di *struttura sociale*, *ruolo*, *norma* entrano in gioco. Dal punto di vista architetturale, un sistema socio-tecnico può essere visto come stratificato (Sommerville *et al*., 2012): lo strato più basso concerne le risorse fisiche del sistema, gli altri gestiscono livelli di astrazione via via più alti arrivando alle leggi e alle regolamentazioni che governano la società di agenti costituita sul sistema, fornendo strumenti per la loro coordinazione e per il monitoraggio della correttezza di quanto accade. In questo modo, come osservato da Singh (2014), un gruppo di agenti autonomi, che sono *disaccoppiati* per natura, che prendono decisioni autonome, che hanno ciascuno obiettivi propri, capacità e risorse differenti, riescono a realizzare una forma di auto-governo.

Nel seguito spesso non distingueremo più fra agenti artificiali e umani ma parleremo di *agenti autonomi*, poiché l'astrazione cattura entrambi i casi. Gli approcci che vedremo si dividono in *procedurali*, focalizzati sulla descrizione di precise sequenze di passi (il *come*), e *dichiarativi*, focalizzati invece sulla specifica di *cosa* le parti debbano realizzare o rispettare nella loro (inter)azione (Baldoni *et al*., 2010). Come spiegato da Baldoni, Baroglio e Capuzzimati (2014) gli approcci dichiarativi si prestano più facilmente allo sviluppo di sistemi tecnologici complessi, dove la scalabilità è un problema e i rischi di comportamenti imprevisti sono più elevati (Sommerville *et al*., 2012).

3.1. Interazione basata su protocolli

L'interazione basata su *protocolli* rappresenta un primo esempio di organizzazione sociale delle relazioni tra agenti autonomi. In termini generali, un *protocollo di interazione* definisce l'insieme di regole e convenzioni che definiscono come gli agenti comunicano tra loro per raggiungere ognuno i propri obiettivi.

Nell'ambito degli approcci procedurali, un protocollo rappresenta una specifica formale che determina la tipologia e la sequenza dei messaggi che gli agenti devono scambiarsi per portare a termine una determinata interazione. Ad esempio, supponendo che due agenti debbano negoziare un prezzo, il protocollo dovrebbe stabilire: (i) chi inizia la negoziazione, (ii) quali messaggi possono essere inviati (proposta, accettazione, rifiuto, controproposta) e da chi, (iii) in quale ordine questi messaggi devono essere scambiati, (iv) quando l'interazione termina, ed eventualmente (v) cosa accade in caso di mancata risposta o errore. Questa tipologia di protocolli deriva in parte dalla teoria degli atti linguistici (*Speech Act*

Theory) (Austin, 1975; Searle, 1979), che considera ogni messaggio come un atto dotato di forza illocutoria, cioè in grado di modificare lo stato mentale dell'agente ricevente. La specifica concreta si fonda principalmente su linguaggi standardizzati, chiamati *Agent Communication Languages* (ACL), come ad esempio FIPA-ACL[1]. L'obiettivo di questi linguaggi è definire un insieme preciso di *performative* (es. *inform*, *request*, *propose*, *accept*, *reject*) attraverso cui gli agenti possono esprimere intenzioni e obiettivi con un significato formalmente preciso e condiviso.

Dal punto di vista computazionale, i protocolli di interazione fungono da contratto sociale esplicito tra gli agenti; specificano ciò che ogni agente può o deve fare in determinate circostanze, rendendo possibile tra l'altro la verifica formale di alcune proprietà del sistema, come ad esempio correttezza (ovvero la garanzia del rispetto di proprietà di interesse), terminazione (l'interazione termina in maniera prevista), assenza di punti di stallo (assenza di *deadlock*), ecc. Inoltre, essi promuovono l'interoperabilità tra agenti sviluppati da organizzazioni o in linguaggi differenti, poiché forniscono un livello di astrazione indipendente dalle implementazioni interne dei singoli agenti.

Un grosso limite degli approcci procedurali è che tendono a essere iper-prescrittivi perché vincolano gli agenti a seguire rigidamente le sequenze previste. In pratica l'agente seguirà una sorta di procedura che dice se e come rispondere a un messaggio e se e quando essere proattivo nell'invio di un messaggio. Ciò è possibile solo realizzando un legame forte tra la logica secondo la quale si sviluppa l'interazione degli agenti e la logica decisionale con cui ciascun agente delibera le proprie azioni, riducendo la flessibilità e l'autonomia che sono invece fondamentali nel contesto di sistemi socio-tecnici, soprattutto quelli aperti e distribuiti computazionalmente e geograficamente.

Per superare tale limite, in letteratura sono stati proposti diversi *approcci dichiarativi* che sostituiscono la forma procedurale dei protocolli di interazione con *condizioni sociali* o *informative* che devono essere soddisfatte affinché l'interazione possa dirsi corretta. Gli agenti sono liberi di scegliere liberamente il proprio corso d'azione finché questo li mantiene nei confini stabiliti dalle regole sociali. In pratica i protocolli (procedurali) sono sostituiti da una rappresentazione formale delle aspettative reciproche delle parti, unita all'introduzione di meccanismi di monitoraggio e sanzione delle violazioni. Un esempio è rappresentato dai protocolli basati su commitment o *commitment-based protocols* (Castelfranchi, 1995; Singh, 1999).

[1] Foundation for Intelligent Physical Agents specifications.

Qui l'interazione viene descritta in termini di *impegni sociali* (social commitments) che nascono, cambiano stato e si estinguono nel corso dell'interazione. Un commitment rappresenta una relazione tra un agente debitore e un agente creditore, nella quale il primo si impegna a far sì che una determinata condizione (conseguente) risulti vera se valgono certe condizioni di contesto (antecedente). L'interazione tra gli agenti è governata dalla creazione, aggiornamento, soddisfazione/violazione di tali impegni. Ad esempio, si consideri uno scenario di acquisto online. Un agente acquirente invia a un agente venditore una richiesta di ordine per un prodotto. Il venditore può allora creare un commitment, secondo il quale "il venditore si impegna a spedire il prodotto all'acquirente una volta ricevuto il pagamento", dove "spedire il prodotto" è il conseguente e "ricevuto il pagamento" è l'antecedente. Quando l'acquirente effettuerà il pagamento, la condizione antecedente del commitment diventerà vera e il venditore risulterà obbligato a soddisfare l'impegno preso perché l'aspettativa sociale è che anche il conseguente diventi vero, ad esempio inviando una conferma di spedizione. Se per qualche ragione il venditore non adempie, l'impegno viene considerato violato. L'aspetto interessante è che fra antecedente e conseguente non vi è una relazione temporale, nonostante i nomi, per cui se per esempio il venditore si fida dell'acquirente, che conosce come cliente abituale, potrebbe effettuare la spedizione prima ancora di ricevere il pagamento. Ricevuto il pagamento l'impegno sarebbe automaticamente soddisfatto.

3.2. Interazione basata su norme

I *sistemi multi-agente normativi* (normative MAS) (Boella *et al.*, 2007) integrano esplicitamente la nozione di *norma* – intesa come vincolo espresso in termini di concetti deontici, quali obblighi, permessi e divieti: sono le norme che regolano il comportamento degli agenti all'interno di un ambiente condiviso, esplicitando la governance del sistema socio-tecnico nel suo insieme. Le norme specificano diritti, doveri e poteri istituzionali dei partecipanti ad un certo contesto, e definiscono le conseguenze – giuridiche, sociali o organizzative – derivanti dal rispetto o dalla violazione di tali regole. Gli agenti rimangono dei soggetti autonomi che possono decidere se conformarsi o meno alle norme che li riguardano, valutando costi, benefici e sanzioni. Questo carattere delle norme consente di rappresentare in modo realistico la complessità delle relazioni sociali, in cui l'obbedienza non è automatica, ma è il risultato di un processo deliberativo.

Dal punto di vista computazionale, le norme possono essere formalizzate attraverso logiche modali – deontiche o temporali (Jones, Carmo,

2001). L'infrastruttura del sistema è dotata di meccanismi di monitoraggio dell'applicazione delle norme che rilevano la conformità del comportamento degli agenti e le eventuali violazioni, a seguito delle quali il sistema normativo può comminare sanzioni.

Un'astrazione che si è mostrata particolarmente utile per la realizzazione di sistemi normativi è quella di *organizzazione* (Hübner, Sichman, Boissier, 2007; Coutinho, Sichman, Boissier, 2009). Nelle società umane, le organizzazioni possono essere interpretate come sistemi di regole che danno significato alle azioni coordinate di agenti autonomi sotto un quadro normativo condiviso. Analogamente, in un sistema socio-tecnico, un'organizzazione rappresenta un contesto all'interno del quale gli agenti, attraverso l'adozione di determinati *ruoli*, aderiscono a un corpus di norme condiviso e ne diventano soggetti. Oltre agli obiettivi individuali degli agenti, una struttura organizzativa include tipicamente degli obiettivi organizzativi, che rappresentano finalità a livello macro dell'intero sistema. Il raggiungimento di tali obiettivi richiede forme di coordinamento, allocazione delle risorse, pianificazione collettiva e meccanismi di decomposizione dei task in sotto-obiettivi assegnabili ai vari agenti, secondo i ruoli che essi giocano.

Un ruolo definisce un insieme di comportamenti attesi, competenze richieste, vincoli di interazione e conferisce poteri agli agenti che lo interpretano. L'assegnazione dei ruoli facilita la specializzazione funzionale dell'organizzazione e consente di modellare gerarchie, deleghe e cooperazione. Gli agenti possono poi adottare, abbandonare o negoziare ruoli in funzione dei loro obiettivi.

4. Agenti e artefatti

Da diversi decenni il mondo della ricerca ha manifestato interesse verso l'Activity Theory in relazione al mondo dei computer (hardware e software). Un esempio è una raccolta curata da Nardi (1995) di contributi che presentano l'Activity Theory come uno strumento per studiare l'interazione fra uomo e computer, dal progetto pratico allo sviluppo teorico. Nell'ambito dello sviluppo software, e in particolare dei sistemi socio-tecnici ad agenti, centrale è il modello *Agents & Artifacts* (Ricci, Viroli, Omicini, 2005). Come spiegato da Ricci, Viroli e Omicini, il modello propone di introdurre un nuovo concetto alla pari di quello di agente: l'*artefatto.* Gli artefatti, a differenza degli agenti, pur potendo avere un comportamento, non hanno obiettivi propri e quindi non sono proattivi, piuttosto sono strumenti che permettono di realizzare il contesto in cui gli agenti operano. Invece che da goal e task, gli artefatti sono caratterizzati da *usi e funzioni*

e, così come accade per gli strumenti del mondo fisico, possono essere utilizzati esclusivamente tramite un insieme di operazioni ben definite. Sono introdotti in qualità di strumenti che permettono agli agenti di perseguire i propri fini.

Gli artefatti software possono servire diversi scopi. Particolarmente interessanti sono gli artefatti che consentono agli agenti di interagire e di coordinarsi perché implementano dei protocolli di interazione oppure perché implementano una realtà organizzativa. Nel primo caso potrebbero, per esempio, modificare l'insieme delle azioni eseguibili dagli agenti a seconda dello stadio del protocollo che è stato raggiunto. Oppure potrebbero implementare il ciclo di vita dei commitment sociali, creandoli, attivando gli obblighi connessi, eccetera a seconda delle azioni degli agenti. Un agente venditore che indica un prezzo per un oggetto a cui un agente cliente è interessato (tramite un opportuno artefatto software) non sta facendo soltanto una comunicazione ma con la sua affermazione creerà un'aspettativa nell'altro agente relativa a quanto spenderà in caso di acquisto dell'oggetto e tutto questo sarà gestito dall'artefatto tramite la creazione di un commitment sociale. L'affermazione del venditore è quindi vincolante anche per il sistema socio-tecnico. Se il cliente deciderà effettivamente di procedere con l'acquisto, sarà pronto a spendere la cifra indicata. Se a questo punto il venditore dovesse cambiare le carte in tavola, chiedendo di più, il patto implicito sarebbe rotto. L'acquirente potrebbe quindi etichettare pubblicamente il venditore come poco affidabile, riducendo la fiducia di cui il venditore gode nella società degli agenti. Da notare che il venditore non è obbligato a vendere un oggetto ma se risponde a una richiesta indicando un prezzo, con quel gesto prende anche un impegno. Questo modo di interagire può essere supportato via software, tracciando l'evoluzione dei commitment con degli artefatti. Nel secondo caso gli artefatti potrebbero definire lo spazio di azione degli agenti a seconda dei ruoli da essi rivestiti e delle norme, ancora una volta attivate automaticamente a seconda della situazione e delle azioni eseguite. Per esempio, se pensiamo a un sistema automatico di assistenza alla guida in città, il diventare rosso di un semaforo fa scattare una norma che crea degli obblighi verso certi agenti automobilisti. Tali agenti possono fisicamente violare l'obbligo a stare fermi ma questa violazione sarà rilevata e il sistema procederà a imporre una sanzione.

5. Agenti e accountability

Il concetto di accountability è complesso da definire. In lingua italiana non esiste un esatto omologo e viene spesso reso con "responsabilizzazio-

ne", e un esempio di sua applicazione è il GDPR. In inglese, nel linguaggio comune, è spesso associato al biasimo: *accountable* è la persona da biasimare per un'azione o un errore che è occorso oppure che accadrà in futuro. Questa, tuttavia, è un'accezione molto parziale del concetto sempre relazionale fra due parti, secondo il quale entrambe riconoscono all'una l'autorità di fare qualcosa e all'altra l'autorità di chiedere conto in merito a quel qualcosa. Questa relazione assume innumerevoli sfumature a seconda del contesto (politico, amministrativo, produttivo, ecc.).

In ambito socio-tecnico (Baldoni, Baroglio, Micalizio, 2014) l'accountability costituisce uno strumento interessantissimo soprattutto quando molti agenti, che non hanno una visione globale di tutta l'informazione rilevante per deliberare le proprie azioni, devono collaborare con *azioni di sistema*. Può infatti capitare, per esempio, che un agente che sta svolgendo un compito per un altro incorra in una situazione anomala. Tale agente non sa a quale fine dell'altro contribuisca il suo operato; quindi, non si può pensare che trovi una soluzione al problema. Solo l'agente che ha delegato un compito è a conoscenza del contesto in cui è possibile interpretare correttamente il problema occorso. Serve di conseguenza una collaborazione fra i due, che potrebbe estendersi a terze e ulteriori parti.

Quando si parla di "chiedere conto" o "rispondere" non si parla, in contesto di accountability, di quesiti generici, tipo "come ci si iscrive a un corso di laurea?" ma di situazioni specifiche, tipo "perché la mia procedura di iscrizione è bloccata?". La gestione di situazioni anomale è un caso tipico in cui l'accountability è utile ma non è l'unico. È esperienza comune interpellare chi potrebbe avere le informazioni necessarie su un caso di interesse, quando si sente di avere il credito per farlo.

I sistemi socio-tecnici tradizionali, come pure quelli basati su commitment o su protocolli di interazione, non prevedono un supporto a questa modalità di interazione, la cui realizzazione richiede di agire lungo due dimensioni: una dimensione *normativa*, che cattura i diritti e i doveri delle parti, e una dimensione *strutturale*, che cattura chi può rispondere su cosa e a chi, creando dei percorsi lungo i quali l'informazione corre.

Per intuire la differenza vediamo come lo stesso imprevisto sarebbe gestito sia in un contesto normativo senza accountability sia in un contesto con accountability. Supponiamo che un medico attenda gli esiti di un'analisi, commissionata al laboratorio dell'ospedale. In un contesto normativo ci si aspetta che gli agenti rispettino i loro obblighi. Supponiamo che il laboratorio ritardi la consegna: il suo obbligo risulterà violato. In questo caso l'unico strumento che il sistema ha a disposizione per cercare di risolvere le cose è far scattare una sanzione. L'assunto implicito è che il laboratorio di analisi abbia violato l'obbligo consapevolmente, avendo sia le capaci-

tà sia i mezzi per fare quanto stabilito (per inefficienza o perché ha dato priorità ad altri compiti). La sanzione intende essere uno stimolo affinché il laboratorio si comporti correttamente (rispetti gli obblighi) in futuro. Supponiamo però che il laboratorio di analisi non abbia restituito il referto in tempo per via di un ritardo nella consegna dei reagenti necessari da parte di un fornitore. La sanzione sarà stata inutile in quanto né permette al laboratorio di assolvere al compito assegnato né al medico di trovare una strategia alternativa. Riattivare successivamente l'obbligo al laboratorio di analisi, come alcuni sistemi fanno, sarebbe altrettanto inutile se non cambiano le condizioni al contorno. In un sistema con accountability, il medico avrebbe avuto non solo il diritto ma anche gli strumenti per chiedere conto al laboratorio di analisi in merito al ritardo e il laboratorio avrebbe condiviso l'informazione sulla mancanza di reagenti. Questa azione non sarebbe finalizzata solo a giustificarsi per evitare una sanzione, permetterebbe al medico di interpretare queste nuove informazioni nel contesto delle ragioni per cui aveva chiesto le analisi, valutando se attendere ulteriormente o se rivolgersi ad un altro laboratorio.

6. Intelligenza artificiale generativa e agentic AI

Negli ultimi anni hanno avuto grande impulso modalità di interazione con il computer che non richiedono conoscenza degli specifici strumenti ma hanno piuttosto una forma colloquiale. Ciò è stato possibile grazie allo sviluppo di tecnologie avanzatissime per la comprensione e l'elaborazione del cosiddetto *linguaggio naturale*, termine utilizzato in ambito informatico in contrapposizione a linguaggio di programmazione, per riferirsi al linguaggio parlato dagli esseri umani. Le nuove modalità di interazione permettono da un lato di esprimere richieste in forma di frasi, che possono essere molto articolate e comprendere esempi, note come *prompt*. Dall'altro permettono ai sistemi software di produrre sintesi, traduzioni o altri tipi di documenti e media (inclusi programmi) per conto dei richiedenti.

L'ambito scientifico che studia e sviluppa questo tipo di soluzioni è noto come *intelligenza artificiale generativa* (Banh, Strobel, 2023; Sengar *et al.,* 2025) e rappresenta uno degli sviluppi più recenti dell'apprendimento automatico. L'apprendimento automatico (in inglese *machine learning*) a sua volta è quella disciplina dell'intelligenza artificiale che si occupa di sviluppare algoritmi e modelli che consentono ai computer di apprendere dai dati e di fare previsioni, migliorando la loro performance con l'esperienza. Esistono diversi modelli di apprendimento automatico, ognuno adatto a compiti specifici. Nel caso dell'IA generativa i modelli sono

progettati per creare autonomamente una vasta gamma di media, inclusi testi, immagini, musica, voce, video e persino strutture molecolari. Tali capacità, che possono trarre in inganno soprattutto chi è digiuno di queste tecnologie, non sono scevre di limiti, come ben sottolineato in Hicks, Humphries e Slater (2024). L'IA generativa si basa su meccanismi *probabilistici* producendo, nel caso del testo, il proseguimento più probabile del prompt sulla scorta dei documenti elaborati in fase di apprendimento (per altre tipologie di media vale un discorso simile). Lo scopo è produrre una risposta *verosimile*. Tuttavia, chi pone un quesito allo strumento si aspetta una risposta *vera*, esattamente come se stesse facendo un'estrazione di dati da una banca dati o come se calcolasse il valore di una funzione complessa su certi input. La veridicità delle risposte richiede delle verifiche come, più in generale, anche la qualità delle risposte richiede verifiche e spesso rielaborazioni. Nel contesto dei sistemi socio-tecnici passare da strumenti deterministici a strumenti probabilistici costituisce un'enorme differenza e affinché l'uso che se ne fa sia corretto è necessaria un'adeguata formazione di chi lo utilizza.

L'agentic AI (Acharya, Kuppan, Divya, 2025; Huang, 2025; Murugesan, 2025) si inserisce in questo panorama come evoluzione dell'IA generativa verso modelli capaci non solo di produrre contenuti, ma anche di *agire in modo autonomo o semi-autonomo* perseguendo obiettivi complessi. Se i modelli generativi forniscono le capacità di elaborazione e generazione linguistica, l'agentic AI aggiunge un livello ulteriore: la possibilità per questi modelli di interagire tra di loro e con l'ambiente, prendere decisioni, coordinare strumenti esterni e adattare il proprio comportamento in base al contesto. Per esempio, diventa possibile chiedere a un agente di prenotare per noi un tavolo a un ristorante. Per raggiungere l'obiettivo l'agente dovrà compiere delle azioni che non si limitano alla produzione di un testo o un'immagine ma che consistono nell'interazione col sistema di prenotazione del ristorante, adattandosi a ciò che trova. Nel caso sia fornito un numero di telefono dovrà fare una telefonata e dialogare con chi sarà all'altro capo. Inoltre, dovrà fornire un feedback al richiedente, soprattutto nel caso in cui in quel ristorante non vi sia posto. In questo senso, l'agentic AI si riconduce naturalmente ai concetti di agenti autonomi, che però operano seguendo prompt linguistici che esprimono obiettivi ad alto livello, e di sistemi multi-agente, in cui più agenti collaborano o si specializzano per risolvere problemi articolati.

Tuttavia, è importante sottolineare che ciò che oggi viene definito come agentic AI si traduce, nella pratica, soprattutto in un'orchestrazione di workflow composti da chiamate multiple a modelli generativi, generalmente espresse in linguaggio naturale. Questi workflow consistono in sequenze

di prompt che il sistema utilizza per pianificare, scomporre ed eseguire un compito, spesso delegando parti del lavoro a modelli diversi e specializzati per ciascuna fase. Questa prospettiva, pur efficace e flessibile, non realizza pienamente il concetto classico di agente autonomo come inteso nell'*agent-oriented software engineering*. Qui, infatti, l'autonomia è concepita in modo più rigoroso: gli agenti dispongono di obiettivi interni, modelli espliciti di credenze e intenzioni, capacità di interazione, e sono progettati come entità software dotate di cicli deliberativi complessi. In confronto, l'agentic AI attuale sfrutta la potenza dei modelli generativi per ottenere comportamenti che appaiono "agentici", ma senza una struttura interna altrettanto formalizzata. L'autonomia emergente deriva dall'abilità dei modelli di generare e interpretare testo o altri media, più che da un'architettura concettuale modellata in modo esplicito.

Bibliografia

Acharya D.B., Kuppan K., Divya B. (2025), *Agentic AI: Autonomous intelligence for complex goals – a comprehensive survey*, «IEEE Access».

Austin J.L. (1975), *How to do things with words*, Harvard University Press, Cambridge (MA).

Baldoni M., Baroglio C., Capuzzimati F. (2014), *A commitment-based infrastructure for programming socio-technical systems*, «ACM Transactions on Internet Technology», 14(4), pp. 23:1-23:23.

Baldoni M., Baroglio C., Mascardi V., Omicini A., Torroni P. (2010), "Agents, multi-agent systems and declarative programming: What, when, where, why, who, how?", in Dovier A., Pontelli E. (a cura di), *A 25-Year Perspective on Logic Programming: Achievements of the Italian Association for Logic Programming, GULP*, Springer, pp. 204-230.

Baldoni M., Baroglio C., Micalizio R., Tedeschi S. (2021), *Reimagining robust distributed systems through accountable MAS*, «IEEE Internet Computing», 25(6), pp. 7-14.

Banh L., Strobel G. (2023), *Generative artificial intelligence*, «Electronic Markets», 33(1), p. 63.

Boella G., van der Torre L.W.N., Verhagen H. (2007), "Introduction to normative multiagent systems", in *Normative Multi-agent Systems*, Dagstuhl Seminar Proceedings, vol. 07122.

Castelfranchi C. (1995), "Commitments: From individual intentions to groups and organizations", in *Proceedings of ICMAS*, vol. 95, pp. 41-48.

Coutinho L.R., Sichman J.S., Boissier O. (2009), "Modelling dimensions for agent organizations", in *Handbook of research on multi-agent systems: Semantics and dynamics of organizational models*, IGI Global, pp. 18-50.

Dalpiaz F., Chopra A.K., Lim S.L. (2011), *Proceedings of the 1st International Workshop on Requirements Engineering for Social Computing*, IEEE, Trento.

Foundation for Intelligent Physical Agents (s.d.), *FIPA specifications*, FIPA Specifications.
Hicks M.T., Humphries J., Slater J. (2024), *ChatGPT is bullshit*, «Ethics and Information Technology», 26.
Huang K. (2025), *Agentic AI*, Springer.
Hübner J.F., Sichman J.S., Boissier O. (2007), *Developing organised multiagent systems using the MOISE+ model: Programming issues at the system and agent levels*, «International Journal of Agent-Oriented Software Engineering», 1(3/4), pp. 370-395.
Jones A.J.I., Carmo J. (2001), "Deontic logic and contrary-to-duties", in Gabbay D. (a cura di), *Handbook of Philosophical Logic*, Kluwer, pp. 203-279.
Murugesan S. (2025), *The rise of agentic AI: implications, concerns, and the path forward*, «IEEE Intelligent Systems», 40(2), pp. 8-14.
Nardi B.A. (a cura di) (1995), *Context and Consciousness: Activity Theory and Human-Computer Interaction*, The MIT Press.
Ricci A., Viroli M., Omicini A. (2005), "Programming MAS with artifacts", in Bordini R.H., Dastani M., Dix J., El Fallah Seghrouchni A. (a cura di), *Programming Multi-Agent Systems*, Springer, pp. 206-221.
Russell S., Norvig P. (2003), *Artificial Intelligence: A Modern Approach*, Prentice Hall.
Searle J.R. (1979), *Expression and meaning: Studies in the theory of speech acts*, Cambridge University Press.
Sengar S.S., Hasan A.B., Kumar S., Carroll F. (2025), *Generative artificial intelligence: a systematic review and applications*, «Multimedia Tools and Applications», 84(21), pp. 23661-23700.
Singh M.P. (1999), *An ontology for commitments in multiagent systems*, «Artificial Intelligence and Law», 7(1), pp. 97-113.
Singh M.P. (2011), "Information-driven interaction-oriented programming: BSPL, the blindingly simple protocol language", in *Proceedings of the 10th International Conference on Autonomous Agents and Multiagent Systems (AAMAS '11)*, International Foundation for Autonomous Agents and Multiagent Systems, pp. 491-498.
Singh M.P. (2014), *Norms As a Basis for Governing Sociotechnical Systems*, 5(1), pp. 21:1-21:23.
Sommerville I., Cliff D., Calinescu R., Keen J., Kwiatkowska M.Z., McDermid J.A., Paige R.F. (2012), *Large-scale complex IT systems*, «Communications of the ACM», 55(7), pp. 71-77.
The Foundation For Intelligent Physical Agents (2002), *Interaction protocols*.
Trist E. (1981), *The Evolution of Socio-technical Systems: A Conceptual Framework and an Action Research Program*, *Issues in the Quality of Working Life*, 2.
Whitworth B., Ahmad A. (2013), "Socio-Technical System Design", in Soegaard M., Friis Dam R. (a cura di), *The Encyclopedia of Human-Computer Interaction*, 2nd ed., The Interaction Design Foundation, Aarhus, Denmark, chapter 24.
Wooldridge M. (2009), *An introduction to multiagent systems*, John Wiley & Sons.