Informatica e diritto, XXXIX annata, Vol. XXII, 2013, n. 2, pp. 71-86

Legge sul *software* e materiali preparatori: spunti per una riflessione giuridica e informatica nell'ambito delle architetture *software*

MICHELE ALBERTI, ENRICO CANTONI*

SOMMARIO: *1. La legge sul* software *– 1.1. Una "alchimia giuridica" – 1.2. Verso un* copyright *senza copie? – 1.3. Il materiale preparatorio: una definizione giuridica – 2. I materiali preparatori nel* software *– 2.1. Un insieme di artefatti – 2.2. I documenti di specifica – 2.3. Architettura* software *– 2.4. L'importanza di una notazione formale – 2.5. Il contesto – 2.6. Perché tutelarli? – 3. Conclusione*

1. LA LEGGE SUL SOFTWARE

1.1. Una "alchimia giuridica"

Come ben noto, la tutela della proprietà intellettuale viene generalmente perseguita attraverso tre strumenti giuridici differenti: i marchi, i brevetti e il diritto d'autore. Ciascuno di questi strumenti nasce per tutelare una specifica categoria di opere considerate suscettibili di proprietà intellettuale.

Con riferimento al *software*, appare generalmente indubbio il potersi qualificare come oggetto di proprietà intellettuale. È tuttavia difficile riuscire a stabilire senza alcuna ambiguità se esso debba rientrare nel regime di tutela del diritto d'autore (che tradizionalmente protegge le opere letterarie, scientifiche e artistiche), o di quello delle invenzioni industriali.

Con il d.lgs. 518/92, conformandosi a quanto era ormai diventata convinzione sempre più diffusa in dottrina, "anche il nostro ordinamento ha accolto l'impostazione secondo la quale i programmi per elaboratore elettronico debbono considerarsi un particolare esempio di opera dell'ingegno e, più specificatamente, di opera letteraria, e, conseguentemente, possono godere della tutela apprestata dalla legge sul diritto d'autore"[1].

* M. Alberti è PhD student presso Aix-Marseille Université, Institut de Mathématiques de Luminy (michele.alberti@univ-amu.fr). E. Cantoni è PhD student presso il Massachusetts Institute of Technology, Department of Economics (cantoni@mit.edu). Ringraziamo tutti i docenti e i partecipanti del corso "Proprietà intellettuale" del Collegio Superiore dell'Università di Bologna (a.a. 2009/2010). Un particolare ringraziamento va ad Alberto Musso e Anna Guagnini per l'insostituibile supporto che ci hanno fornito nella redazione del presente lavoro. Ringraziamo inoltre due referees anonimi per gli ottimi suggerimenti.

[1] G. CAVANI, *Oggetto della tutela*, in Ubertazzi L.C. (a cura di), "La legge sul software – commentario sistematico", Milano, Giuffrè, 1994, p. 1.

Come dimostrato dalla storia successiva al suddetto decreto (che, d'ora in avanti, chiameremo per semplicità "legge sul software"), questo intervento non ha posto la parola fine alla lunga diatriba che ha visto contrapporsi i sostenitori di due diverse (ma non necessariamente "incompatibili") concezioni circa la proteggibilità del *software*.

Come è stato giustamente osservato da Cavani[2], alcuni peroravano (e tuttora sostengono) la tesi secondo cui il *software* rappresenta espressione di innovazione tecnologico-produttiva e, di conseguenza, dovrebbe essere protetto come invenzione industriale (cioè tramite brevetto), purché ne sussistano i requisiti di novità, originalità e industrialità. Di diversa opinione erano invece coloro che, in virtù della natura testuale del *software* e del suo essere frutto di un'attività creativa, sostenevano la necessità di qualificare il *software* come opera dell'ingegno. Nonostante sia ormai ovunque prevalsa l'impostazione che vuole il diritto d'autore lo strumento più adeguato per proteggere il *software* (spesso però affiancato da altri strumenti di tutela), sono assai numerose le critiche rivolte a questo approccio.

In particolare:

1. È venuto rapidamente meno il "pregiudizio" secondo cui un programma per elaboratore mancherebbe del requisito di industrialità e, dunque, non paiono oggi esistere sufficienti ragioni per credere che il *software* sia sempre e comunque non proteggibile tramite brevetto.
2. In aggiunta a ciò, la particolare natura del *software* (cioè di "macchina il cui mezzo di costruzione è in forma testuale"[3]), rende il ricorso al diritto d'autore una sorta di "alchimia giuridica", sulla cui efficacia è legittimo sollevare diversi dubbi. In particolare, mentre il diritto d'autore è tradizionalmente rivolto a proteggere la *forma espressiva* di un'opera e non il suo *contenuto*, nel *software* è proprio quest'ultimo che più necessita di protezione. Perché la tutela del *software* sia efficace, è pertanto necessario che il diritto d'autore venga impiegato al di là dei propri confini tradizionali, andando così a proteggere (almeno parzialmente) anche il contenuto. Questo è soltanto il primo degli innumerevoli effetti collaterali che derivano da una progressiva *misappropriation* del diritto d'autore, cioè dell'impiego di questo strumento a nuove e diverse forme espressive ritenute meritevoli di tutela.

[2] *Ibidem*.

[3] P. SAMUELSON, R. DAVIS, M.D. KAPOR, J.H. REICHMAN, *A Manifesto Concerning the Legal Protection of Computer Programs*, in "Columbia Law Review", 1994, n. 8, pp. 2308-2431.

3. È altresì indubbio che, qualora venga accettata la sola protezione della forma espressiva di un'opera, il diritto d'autore fornisce una forma di tutela qualitativamente più debole rispetto a quella offerta dal brevetto. Ciononostante, l'assenza di un requisito di "registrazione" e un livello di originalità sensibilmente più attenuato rispetto a quello previsto per i brevetti fanno sì che l'area di *software* monopolizzabile appaia destinata a crescere di pari passo con la dimensione dell'industria a esso relativa.
4. È infine possibile che, in virtù della maggiore durata temporale assicurata al diritto d'autore, un singolo soggetto diventi titolare dei diritti di numerosi programmi per elaboratore intimamente connessi l'uno con l'altro, così creando forti barriere all'entrata di nuovi concorrenti. In aggiunta a ciò, l'assenza di un istituto atto a regolamentare le eventuali modificazioni o elaborazioni migliorative di un programma (quale è la *licenza obbligatoria* nei brevetti) rende virtualmente possibile un parziale blocco della tecnologia. Anche qualora quest'ultimo effetto non dovesse verificarsi, è fuor di dubbio che, a causa di quanto detto in precedenza, la *misappropriation* del diritto d'autore nei programmi per elaboratore rischia di generare un notevole grado di ambiguità, così contraddicendo quello che dovrebbe in teoria essere lo spirito "chiarificatore" di uno strumento di tutela della proprietà intellettuale.

1.2. Problemi di coordinamento con la tutela brevettuale

Stante la peculiare natura del *software* e la sua "fittizia" inclusione nel regime del diritto d'autore, era inevitabile che si verificassero problemi di coordinamento con altre forme di tutela, in particolar modo con il regime brevettuale. Mentre l'articolo 52 della Convenzione Europea dei Brevetti sancisce che non è possibile brevettare programmi per elaboratore in quanto tali (*"as such"*), tale disposizione non preclude infatti l'opportunità di ricorrere al brevetto per proteggere una particolare inedita applicazione tecnica del *software*. È proprio l'eventuale assenza di uno specifico problema di natura tecnica che dovrebbe consentire di discriminare tra un programma per elaboratore "in quanto tale" e un *software* brevettabile (già comunque beneficiario del diritto d'autore). Negli Stati Uniti, per esempio, i brevetti sui programmi per elaboratore non vennero concessi fino al 1981, quando la Corte Suprema risolse il celebre caso *Diamond v. Diehr*[4] affermando che:

[4] *Diamond v. Diehr*, 450 U.S. 175 (1981).

> "[il software è brevettabile se contiene] una formula matematica [e] implementa o applica tale formula in una struttura o processo tale che, considerato nel suo insieme, assolve una funzione che le leggi vigenti in materia di brevetto sono volte a proteggere."

Sono esempi di una particolare applicazione tecnica "un programma che regoli una apparecchiatura radiologica e che consenta di ottenere la giusta esposizione ai raggi col minor carico di radiazioni nocive per il paziente; o un programma che coordini la gestione di una pluralità di elaboratori tra loro connessi in un'unica rete"[5]. Non è perciò difficile prefigurare l'ipotesi di concorso di tutele: quella del diritto d'autore (di maggiore durata e rivolta – in teoria – alla sola forma espressiva di un programma) e quella brevettuale (di più breve durata e contraddistinta da un più difficile accesso, a causa dei requisiti di industrialità, novità ed originalità e della specifica applicazione tecnica). La possibile sovrapposizione – e i relativi problemi di coordinamento – di questi due strumenti di tutela (e di quella eventualmente presente nel caso del c.d. *firmware*[6]) è una conseguenza quasi naturale dell'applicazione degli istituti convenzionali di tutela della proprietà intellettuale a domini innovativi e non immediatamente assimilabili a quelli delle opere dell'ingegno (diritto d'autore) o delle invenzioni industriali (brevetti).

Pertanto, prima di procedere con la nostra disamina della "legge sul software", ci soffermiamo ora a porci una domanda: alla luce di quel fenomeno che altri autori hanno efficacemente chiamato "dilemma digitale"[7], e di cui il *software* è senza dubbio uno dei principali protagonisti, siamo certi che il concetto di "diritto di copia" rimanga oggi la scelta più opportuna come elemento fondante il diritto d'autore?

1.3. Verso un copyright *senza copie?*

Il "controllo" sulla possibilità di creare copie di un'opera è, nel diritto d'autore, un mezzo nel perseguimento di uno scopo, laddove lo scopo è dato dalla promozione delle scienze e delle arti liberali attraverso la fornitura di incentivi economici agli autori e agli inventori.

[5] G. CAVANI, *op. cit.*, p. 9.

[6] Il *firmware*, cioè quel tipo particolare di *software* integrato direttamente in un supporto a semiconduttori, gode nel nostro ordinamento, in attuazione della Direttiva CEE 16 dicembre 1986, n. 54, di una specifica protezione di tipo brevettuale.

[7] P. SAMUELSON, R. DAVIS, M.D. KAPOR, J.H. REICHMAN, *op. cit.*

Il fatto che si creino copie illegali di un'opera non rappresenta né una condizione necessaria né una ragione sufficiente affinché venga danneggiato l'incentivo di un autore alla creazione di un'opera. Se anche si fotocopiasse 500 volte un libro e si usassero tali fotocopie per alimentare il fuoco di un camino[8], ciò evidentemente non sortirebbe alcun effetto sugli interessi dell'autore dell'opera copiata, né sul suo incentivo a creare ulteriori opere. Il punto è che, fino ad oggi, "copiare" rappresentava un buon "predittore" di una successiva violazione del diritto d'autore. Per vendere una copia illegale di un libro, era necessario copiarla su un nuovo supporto fisico (le fotocopie); per acquistare un quadro contraffatto, era evidentemente necessario che tale opera artistica venisse riprodotta manualmente o meccanicamente su un nuovo supporto. Per questo motivo, la copia rappresentava il risultato di un gesto "intenzionale" (è difficile sostenere di avere "casualmente" fotocopiato un intero libro altrui) destinato a una successiva azione "illegale" (la distribuzione o il consumo privato della copia pirata). Fintanto che si tratta di supporti fisici, il controllo sulle copie è dunque un buon predittore di successive azioni che possono arrecare danno al detentore dei diritti. In definitiva: per distribuire copie pirata, è prima necessario crearle; controllare che non vengano create copie è dunque un buon mezzo per evitare che le stesse vengano distribuite e, infine, "copiare" non è un'azione che fa normalmente parte del processo di consumo.

Tutto ciò cessa però di essere vero nel mondo dell'informatica. Per funzionare, i computer generano automaticamente copie locali del *software* che viene caricato; in informatica creare copie è non solo un processo molto comune, ma spesso tecnicamente fondamentale ai fini della fruizione di un'opera (e ciò è particolarmente vero nel caso dei programmi per elaboratore). Data l'elevata complessità tecnico-giuridica della questione, anche il tentativo di definire la "liceità" di una copia digitale attraverso la creazione di uno spartiacque tra "copie temporanee" e "copie permanenti" rischia di ridursi a un mero esercizio di discrezionalità.

La continua crescita del ruolo svolto dalle tecnologie digitali nella nostra vita quotidiana ha messo fortemente in discussione il ruolo delle "copie" come chiave di volta del sistema del diritto d'autore: la creazione di copie è tecnicamente necessaria per il funzionamento dei personal computer e non per-

[8] Esempio tratto da NATIONAL RESEARCH COUNCIL – COMMITTEE ON INTELLECTUAL PROPERTY RIGHTS AND THE EMERGING INFORMATION STRUCTURE, *The Digital Dilemma: Intellectual Property in the Information Age*, Washington D.C., National Academy Press, 2000.

mette inoltre di prevedere se si verificheranno delle successive azioni capaci di arrecare danno ai detentori dei diritti d'autore.

In conclusione: se lo scopo è quello di garantire incentivi agli autori e ai creatori di contenuti, è oggi difficile essere sicuri che il controllo sulle copie rimanga il mezzo più idoneo per farlo. Ciononostante, la dottrina giuridica, tecnica ed economica è ancora ben lungi dal potere offrire delle alternative concrete capaci di supplire alle debolezze di un sistema di diritto d'autore basato sul concetto di "copia".

1.4. Il materiale preparatorio: una definizione giuridica

All'articolo 2, n. 8, l. 22 aprile 1941, n. 633 come modificato dalla "legge sul software", il legislatore specifica che sono tutelati

> "i programmi per elaboratore, in qualsiasi forma espressi purché originali quale risultato di creazione intellettuale dell'autore. Restano esclusi dalla tutela accordata dalla presente legge le idee e i principi che stanno alla base di qualsiasi elemento di un programma, compresi quelli alla base delle sue interfacce. Il termine programma comprende anche il materiale preparatorio per la progettazione del programma stesso".

È curioso che si specifichi l'esclusione dalla tutela delle idee e dei principi alla base di qualsiasi elemento del programma: se così non fosse, ci troveremmo evidentemente al di fuori del dominio del diritto d'autore, che, come ben sappiamo, è volto a tutelare la sola forma espressiva di un'opera.

Il senso della suddetta (pleonastica) espressione è probabilmente quello di definire un limite all'estensione della tutela offerta dal diritto d'autore al materiale preparatorio. In deroga alla disciplina generale del diritto d'autore, infatti, nel *software* il materiale preparatorio è protetto, purché però dotato di una propria "compiutezza" sul piano dell'organizzazione e della comunicazione delle informazioni in esso contenute[9]. Si tratta di una definizione dai confini estremamente labili, che lascia ancora una volta trasparire i sintomi dell'insufficienza del regime del diritto d'autore nella protezione dei programmi per elaboratore.

Pertanto, ciò che qualifica il materiale preparatorio ai fini della sua tutela sul piano del diritto d'autore è l'esistenza di un "vincolo di compiutezza":

[9] Si veda G. CAVANI, *op. cit*.

nelle parole della Direttiva CEE[10], il materiale preparatorio deve essere "di natura tale da consentire la realizzazione di un programma per elaboratore in una fase successiva".

Tuttavia, vale la pena osservare che, a differenza delle opere dell'ingegno "tradizionali", il *software* manca di una regola di facile applicazione che permetta di discriminare tra un'opera già compiuta e una semplice idea elaborata. Per le opere dell'ingegno "tradizionali", infatti, si considera suscettibile di tutela quel materiale che è capace di generare "quelle reazioni emotive, sul piano della comunicazione, cui è subordinata, secondo la tradizionale accezione, la nascita stessa di un'opera dell'ingegno"[11]. Nel caso del *software* è altresì difficile sostenere che un programma per elaboratore, magari ancora incompiuto, possa suscitare reazioni emotive analoghe a quelle che può, per esempio, sperimentare un compositore leggendo lo spartito (quand'anche incompiuto) di una composizione altrui.

Di conseguenza, l'unico criterio discretivo che rimane per distinguere tra materiale preparatorio tutelabile dal diritto d'autore e "mera idea" è la "compiutezza" sul piano dell'organizzazione e della comunicazione delle informazioni, tale da consentire la realizzazione di un programma per elaboratore o, quanto meno, il successivo perfezionamento di un'opera già esistente.

Secondo G. Cavani[12], "rientra probabilmente fra i materiali tutelati il c.d. 'diagramma di flusso'; il quale rappresenta graficamente la sequenza delle operazioni che il programma deve compiere e che, dunque, ne esprime, seppure in forma sintetica, l'intera costruzione; nonché, a maggior ragione, il c.d. 'diagramma a blocchi', che del precedente ne costituisce lo sviluppo". Samuelson et al.[13] osservano altresì che la tutela dei materiali preparatori nasce per analogia alla protezione di cui tipicamente gode la struttura dettagliata di opere letterarie convenzionali come i romanzi.

A fronte della difficoltà di distinguere tra forma espressiva e contenuto del *software*, la giurisprudenza italiana ha mantenuto un atteggiamento cauto nell'identificare che cosa debba intendersi per materiale preparatorio e nell'affermarne una tutela autonoma rispetto al programma successivamente

[10] Settimo *considerando* della Direttiva CEE n. 91/250.

[11] G. CAVANI, *op. cit.*, p. 16.

[12] *Ivi*, pp. 16-17.

[13] P. SAMUELSON, T. VINJE, W. CORNISH, *Does Copyright Protection Under the EU Software Directive Extend to Computer Program Behaviour, Languages and Interfaces?*, in "European Intellectual Property Review", 2012, n. 3, pp. 158-166.

creato. In particolare, il Tribunale di Roma[14] ha valutato che la mera individuazione di una serie di esigenze dell'utente e delle funzioni del programma destinate a soddisfarle rientra nell'ambito delle "idee" non proteggibili dal diritto d'autore, ancorché le stesse fossero contenute in materiali preparatori. Come nota inoltre Zincone[15] "[...] solo in programmi di una certa complessità potrà distinguersi chiaramente già nella struttura dei diagrammi di flusso una particolare forma espressiva compiuta ed originale, come tale tutelabile nell'ambito del materiale preparatorio". Le difficoltà e l'atteggiamento cauto della giurisprudenza dimostrano pertanto la necessità di definire in maniera chiara e priva di ambiguità la natura tecnica del materiale preparatorio.

2. I MATERIALI PREPARATORI NEL SOFTWARE

Nelle pagine che seguono, ci domanderemo che cosa si possa intendere, sotto il profilo tecnico, con "materiale preparatorio". A tale scopo, ci concentreremo su un processo di sviluppo *software* noto come *modello waterfall*. Pur nella sua semplicità, esso presenta un'ampia gamma di caratteristiche e di risvolti di notevole interesse ai fini della presente analisi.

2.1. Un insieme di artefatti

Sebbene egli stesso non abbia mai utilizzato tale nome per riferirsi alla propria opera, il *modello waterfall* è stato formalmente descritto per la prima volta da Winston W. Royce[16]. Il termine *waterfall* è volto a indicare il modo di procedere sequenziale precipuo dello sviluppo *software* di questo tipo di processo. Tipicamente, questo si articola lungo le seguenti fasi:

1. Analisi dei requisiti. È la fase preliminare del processo: in questo stadio vengono raccolte e analizzate tutte quelle informazioni che definiscono nel dettaglio cosa il cliente si aspetta dal sistema *software* una volta terminato.
2. Progettazione. Prima di iniziare la vera e propria scrittura del sistema *software* da realizzare, ne viene creata la struttura concettuale. Que-

[14] Trib. Roma, 20 dicembre 1993, in "AIDA", 1995, con commento di G. Ghidini.

[15] A. ZINCONE, *Nuovi spunti di riflessione sui limiti di protezione del software e del materiale preparatorio*, in "Il Diritto di Autore", 2000, n. 2, p. 290.

[16] W.W. ROYCE, *Managing the Development of Large Software Systems*, in "Proceedings of IEEE WESCON 26", 1970, pp. 1-9, disponibile on line in www.cs.umd.edu/class/spring2003/cmsc838p/Process/waterfall.pdf.

sta è la fase in cui viene dato avvio alla definizione dell'architettura dell'intero sistema.

3. Implementazione. È la fase di sviluppo per eccellenza: il lavoro viene diviso in moduli (o unità) e la scrittura del *software* ha inizio. Questa fase può comprendere o meno l'integrazione del *software* sinora realizzato.
4. *Testing*. Tutte le funzionalità del programma scritto vengono messe alla prova e viene quindi controllata la conformità del sistema alle specifiche compendiate durante l'analisi dei requisiti.
5. Rilascio e manutenzione. Questa fase, in teoria, può protrarsi illimitatamente. Una volta rilasciato, il sistema *software* entra infatti in uno stato di continua manutenzione, che prevede la risoluzione dei problemi eventualmente riscontrati durante il suo utilizzo.

Il processo di sviluppo in esame prevede che ogni fase possa iniziare soltanto alla conclusione della precedente. Ciascuna di esse termina con la produzione di un insieme di modelli, ossia di artefatti utilizzati sia all'interno del gruppo di sviluppo sia nelle interazioni con il cliente. Gli artefatti possono essere interpretati come il "collante" tra le diverse fasi del processo di sviluppo, ovvero l'oggetto che ciascuna di esse riceve dalla precedente e da cui essa parte per sviluppare il proprio materiale.

In ogni processo di sviluppo *software* i seguenti sono, rispettivamente, i tipici artefatti risultanti da ciascuna fase di lavoro:

1. Documenti di specifica dei requisiti.
2. Documenti di specifica dell'architettura.
3. Sorgente del sistema *software*.
4. Sorgente dei programmi di controllo.
5. Manuale utente e "patch"[17] al sorgente del sistema *software*.

Questi artefatti possono essere visti come i singoli tasselli che, tutti assieme, compongono il mosaico "sistema software". Ognuno di loro è fondamentale per la buona riuscita non solo della fase di sviluppo immediatamente seguente, ma dell'intero prodotto finale.

Come già ricordato in precedenza, la vigente disciplina italiana in materia di diritto d'autore prevede già la protezione della forma "testuale" dei programmi per elaboratore e, dunque, anche di quel codice sorgente dedicato alla verifica del comportamento del sistema *software* nel suo complesso

[17] Con il termine *patch* si indica una porzione di codice sorgente da integrare al sistema *software* e che, tipicamente, risolve un qualche problema dello stesso.

e alla correzione di eventuali difetti. Lo stesso dicasi, per ovvie ragioni, del "manuale utente".

2.2. I documenti di specifica

Gli artefatti appartenenti alla prima categoria prendono il generico nome di "documenti di specifica dei requisiti". Essi non vanno tuttavia immaginati come semplici annotazioni delle richieste e dei vincoli imposti dai clienti. Tali richieste ed esigenze vengono difatti approfonditamente analizzate e rielaborate in modo da rendere il documento finale di effettivo supporto alla seguente fase di progettazione. Solitamente, il documento così realizzato comprende molteplici informazioni di varia natura, tecnica e non. Oltre agli artefatti di tipo tecnico, vengono costruiti anche quelli pensati per coinvolgere i clienti nel processo di sviluppo. Mantenendo quest'ultimi costantemente aggiornati sui progressi del sistema, si cerca di evitare di commettere errori concettuali di base, i quali minerebbero dalle fondamenta la buona riuscita del prodotto finale.

Prendendo in esame il problema della tutela degli artefatti appartenenti ai documenti di specifica dei requisiti, sembra corretto considerare tutelabili soltanto quelli aventi caratteristiche tecniche. Infatti, solo questi presentano quella peculiarità di "compiutezza" di informazioni (che li rende cioè necessari ai fini del perfezionamento di una successiva fase di sviluppo del programma) che la legge impone ai materiali preparatori.

Analogamente ai documenti di specifica dei requisiti, il documento di specifica caratterizzante la struttura generale del sistema *software* finale subisce continui aggiustamenti e integrazioni, nel continuo tentativo di realizzare un sistema il più conforme possibile alle richieste del cliente.

In generale, entrambi i tipi di documenti presi in esame in questa sezione (quelli di specifica dei requisiti e quelli di specifica dell'architettura *software* nel suo complesso) comprendono diverse informazioni, sintetizzate in *modelli*[18] attinenti a diversi livelli di astrazione. Questi documenti sono pertanto degli insiemi di artefatti. Anche se i diversi tipi di documenti di specifica hanno origini differenti perché ideati in fasi di sviluppo distinte, il loro contenuto informativo va considerato un *unicum*. Perfino nei processi di sviluppo meno articolati, come nel caso del *modello waterfall*, le prime

[18] Con il termine modello ci si riferisce ad una rappresentazione semplificata di un sistema (o di una sua parte). Per definizione, i modelli sono artefatti.

due fasi non sono propriamente distinte ed è infatti assente un netto confine logico-sequenziale che le separi.

Nonostante sia ineluttabile che il maggior contributo derivi dagli artefatti realizzati durante la fase di progettazione, tutti quelli finora prodotti vanno a costituire le basi dell'architettura del sistema.

Risulta perciò necesessario analizzare più nel dettaglio il concetto di architettura *software* e cercare quindi di capire se, ed entro quali limiti, gli artefatti che la compongono costituiscano o meno materiale preparatorio.

2.3. Architettura software

Con riferimento ai sistemi *software*, il termine *architettura* indica l'insieme delle decisioni concernenti la progettazione del sistema. Di conseguenza, ogni *software* che si rispetti ha definita una propria architettura.

Come accennato precedentemente, mentre la creazione dell'architettura di un programma per elaboratore può avere particolare rilevanza all'interno di una singola specifica fase, le attività di aggiunta e di perfezionamento pervadono l'intero processo di sviluppo.

Una volta ultimata, l'architettura di un programma per elaboratore comprende tutti quegli artefatti (documenti di specifica, codice sorgente, manuale utente, etc.) che vengono aggiunti e affinati di pari passo al continuo avvicendamento delle diverse fasi del processo di sviluppo.

2.3.1. Stili architetturali

Nel risolvere problemi legati ad aspetti architetturali dei sistemi *software* destinati ai più disparati domini d'applicazione, i progettisti hanno osservato che specifiche scelte progettuali tendono a generare soluzioni di qualità superiore. Confrontate con potenziali soluzioni alternative, queste si contraddistinguono per un più elevato grado di eleganza, efficienza e scalabilità.

Tipicamente, tali soluzioni ottimali si adattano a particolari classi di sistemi e a contesti ben definiti, per i quali forniscono schemi di soluzioni efficienti, rispettando al contempo stringenti vincoli architetturali. Questi insiemi di decisioni vanno sotto il nome di *stili architetturali*.

2.3.2. *Pattern* architetturali

Gli stili architetturali forniscono schemi di soluzioni generiche che, per essere effettivamente applicate, necessitano spesso di essere affinate in de-

cisioni architetturali dal contenuto più specifico. Al contrario, i "pattern architetturali" forniscono degli schemi di soluzioni dal contenuto più mirato e risultano dunque efficienti per approcciare la progettazione di classi di problemi, o addirittura di specifiche sottoclassi all'interno di determinati contesti applicativi. In definitiva, un *pattern* architetturale è una collezione di specifiche decisioni architetturali che sono applicabili a un determinato e ricorrente problema di progettazione e che, parametrizzate in modo opportuno, sono utilizzabili in tutti quei contesti di progettazione architetturale dove tale problema affiora.

2.3.3. Il problema dell'astrazione

L'incessante richiesta di sistemi *software* di elevata qualità e dalla facile manutenzione ha trovato soluzione nel processo di astrazione, ovvero in quel processo di creazione di modelli che astraggono dai dettagli dei sistemi reali per descriverne i concetti e le proprietà principali.

Gli stili e i *pattern* architetturali non sono altro che astrazioni, in quanto sono insiemi di modelli che forniscono, in modo efficiente ed elegante, soluzioni di massima a diversi problemi di progettazione. Infatti, entrambi sono il risultato dell'opera di razionalizzazione e fattorizzazione, in modelli astratti, di anni di studio e di esperienze nel campo della progettazione architetturale del *software*.

Ragionando per livelli di astrazione, gli effettivi modelli utilizzati nel corso dello sviluppo di un sistema sono *istanze* di stili e di *pattern* architetturali. Pertanto, questi ultimi possono essere definiti "meta-modelli"[19].

Quanto affermato sinora ci porta a concludere che, seppur definiti mediante modelli e altri artefatti, gli stili e i *pattern* architetturali non possono essere oggetto di protezione nel senso inteso dalla legge sul *software*. In particolare, esistono valide ragioni per credere che essi non siano direttamente categorizzabili come materiale preparatorio. Il problema fondamentale risiede proprio nella loro principale caratteristica: l'astrazione. Essi sono infatti troppo "distanti" dagli artefatti costituenti l'architettura di un sistema *software* reale. In altri termini, essi sono tecnicamente e concettualmente più assimilabili alle "idee e principi generali" che la legge sul *software* espressamente esclude dalla tutela del diritto d'autore.

[19] Un modello che descrive un modello è detto meta-modello. I modelli creati in conformità alle "regole" definite nel meta-modello si dicono istanze del meta-modello stesso.

Se il problema degli stili e dei *pattern* architetturali è l'elevato grado di astrattezza, lo stesso non si può tuttavia dire dei modelli utilizzati regolarmente in ogni processo di sviluppo *software*. Questi sono istanze di *pattern* e di stili architetturali e, come tali, ne rispecchiano la struttura. Ciononostante, si ricordi che stili e *pattern* architetturali presentano soluzioni generiche che necessitano solitamente dei giusti adeguamenti dettati dalle esigenze specifiche dei differenti contesti applicativi. Di conseguenza, la realizzazione di appropriate istanze richiede spiccate doti di creatività, deduzione e capacità critica, abilità alla base di quel processo di creazione intellettuale che la legge italiana sul diritto d'autore mira a tutelare e che, pertanto, rendono questi modelli candidati naturali per la definizione di materiale preparatorio.

Individuati gli artefatti potenzialmente tutelabili, appare però opportuno condizionare la loro protezione alla sussistenza di un requisito fondamentale: la forma.

2.4. L'importanza di una notazione formale

Per meglio comprendere l'importanza della presenza di una notazione formale e standardizzata, si pensi a un'intera progettazione architetturale in artefatti espressi in linguaggio naturale (per esempio, la lingua italiana). Quest'ultimo risulta decisamente espressivo, ma, allo stesso tempo, tende ad essere ambiguo, non rigoroso e piuttosto informale.

Poiché uno stesso insieme di artefatti privi di codifica formale può essere oggetto di un numero assai elevato di interpretazioni potenzialmente diverse l'una dall'altra, pare ragionevole escluderli dalla categoria dei materiali preparatori. In altre parole, in assenza di una notazione formale, tali artefatti non sono in grado di fornire un insieme di indicazioni e/o informazioni sufficientemente univoco per poter realizzare un programma per elaboratore.

Di conseguenza, una condizione necessaria per qualificare tali modelli come materiale preparatorio è la loro espressione in una notazione formale, che renda impossibile, con riferimento a uno stesso modello, l'insorgere di diverse interpretazioni.

2.5. Il contesto

Ai fini della tutelabilità da parte del diritto d'autore, oltre al requisito della descrizione formale, è necessario riflettere sul contesto in cui i modelli

vengono elaborati. "Il vincolo di compiutezza", così come è stato definito sopra (par. 1.4), fa sì che gli artefatti debbano essere funzionalmente utili allo sviluppo di un sistema *software*. In altre parole, essi devono far parte di un più ampio contesto nel quale fornire un apporto concreto e, inoltre, la stessa architettura di cui fanno parte dovrebbe esibire un alto grado di completezza logico-strutturale.

Di conseguenza, i modelli che non sono parte di una ben determinata architettura *software*, logicamente e strutturalmente autonoma (cioè in grado di essere direttamente sfruttata per la realizzazione di un programma per elaboratore), dovrebbero rimanere esclusi dalla definizione di materiali preparatori protetti dalla legge sul *software*.

2.6. Perché tutelarli?

Una volta individuati gli artefatti meritevoli di tutela (cioè quelle istanze di stili o *pattern* architetturali espressi in notazione formale e dotati di un elevato grado di autonomia logico-strutturale), viene da chiedersi perché il materiale preparatorio debba essere oggetto di protezione. Dopo tutto, il codice sorgente dei programmi per elaboratore è già tutelato e, senza alcuna ombra di dubbio, esso rappresenta l'artefatto in assoluto più "prezioso".

In realtà, quest'ultima affermazione è solo parzialmente corretta. È chiaro che disporre del codice sorgente è il modo più efficiente e immediato per appropriarsi delle funzionalità proprie di un sistema *software*. Eppure, è evidente che il contenuto informativo degli artefatti sinora esaminati non è nullo. Infatti, oltre a essere il "cuore" di ogni architettura *software*, i modelli stanno conquistando sempre più importanza nel modo di sviluppare programmi per elaboratore. Facendo leva sul concetto di astrazione e di trasformazione tra modelli[20], un intero ramo di ricerca nell'ambito dell'ingegneria del *software* sta studiando e mettendo a punto la possibilità di ottenere il codice sorgente a partire direttamente dai modelli.

Risulta perciò naturale pensare a una loro tutela. Dopotutto, la loro creazione è un lavoro non particolarmente dissimile da quei processi creativi le cui opere sono già protette dal diritto d'autore.

[20] I concetti di astrazione e di modello vanno di pari passo in informatica. Lo stesso codice sorgente può essere visto come un modello, cioè un'astrazione delle operazioni svolte dall'hardware del calcolatore in una codifica di più facile comprensione all'essere umano. In breve, la trasformazione tra modelli ha l'obiettivo di ottenere il codice sorgente da modelli più astratti.

D'altro canto, quello dei modelli che descrivono architetture *software* è un caso parzialmente diverso. Infatti, questi ultimi codificano in una particolare notazione un'*idea* di soluzione (per quanto specifica e circostanziata essa possa essere) a un dato problema di progettazione *software*. Di conseguenza, non è del tutto fuori luogo domandarsi se, e in quali specifiche circostanze, il diritto d'autore sia lo strumento di tutela più adeguato al caso.

3. Conclusione

La tutela del materiale preparatorio è, in definitiva, uno dei segni manifesti dell'inadeguatezza del diritto d'autore nel mondo del *software*: grazie a essa traspare infatti la volontà di proteggere non solo la mera forma espressiva di un'opera-*software*, ma anche (almeno in parte) l'idea ad essa sottostante. A causa di una definizione necessariamente molto generica e poco circostanziata di materiale preparatorio, si rischia infatti di far rientrare nel dominio del diritto d'autore anche le "idee elaborate" (che tradizionalmente godono di altre forme di protezione, come la tutela delle informazioni riservate), purché dotate di un minimo grado di "compiutezza" organizzativa e comunicativa. Se così fosse, la tutela fornita dal diritto d'autore ai programmi per elaboratore risulterebbe essere assai più incisiva (leggi: *monopolistica*) di quanto si potrebbe altrimenti pensare.

La definizione di materiale preparatorio apre a diversi spunti di riflessione sugli artefatti prodotti durante lo sviluppo di un sistema *software*. Innanzitutto, è estremamente difficile determinare quali artefatti rientrino nell'ambiguo novero del materiale preparatorio. Il codice sorgente (compresi test e *patch*) e il "manuale utente" sono, per via della loro particolare natura, artefatti tutelati, ma va sottolineato che lo sarebbero a prescindere da cosa si possa concretamente intendere per materiali preparatori.

La legge, dunque, dovrebbe dare una chiara definizione *ex ante* della protezione (o non protezione) di tutti quegli artefatti che vengono realizzati nel corso del ciclo di sviluppo, ma che non sono visibili una volta che il sistema *software* è ultimato. Ci si riferisce, in particolare, ai modelli prodotti durante la fase di analisi delle specifiche e, più in generale, a quelli costituenti l'architettura del sistema. Purtroppo, però, la definizione di materiale preparatorio fornito dalla legge sul *software* risulta, proprio in questi casi, di difficilissima applicazione.

Si è perciò proposto in questo contributo di considerare come materiale preparatorio l'insieme di quei modelli che, espressi in modo formale, con-

tribuiscono direttamente e in maniera consistente allo sviluppo del sistema *software* finale, che ne tracciano le caratteristiche (strutturali o comportamentali) o che ne determinano specifiche proprietà.