Em002-2 Istruzioni sulla pubblicazione di software a codice sorgente aperto

Raccomandazione sull'informatica nell'Amministrazione federale[1]

Il presente documento è un allegato indipendente del documento principale Em002.

Classificazione[2]:	Non classificato
Carattere vincolante[3]:	Raccomandazione
Settore di pianificazione[4]:	TIC dell'Amministrazione federale
Versione attuale:	1.0
Sostituisce versione:	Nuovo documento
Stato:	Bozza
Data di approvazione (versione attuale):	14.02.2025
Approvato da, base legale:	Approvato dal delegato alla trasformazione digitale e alla governance delle TIC (delegato TDT) in virtù dell'articolo 17 capoverso 1 dell'ordinanza del 25 novembre 2020 sulla trasformazione digitale e l'informatica (OTDI), RS 172.010.58
Lingue:	Tedesco (originale), francese, italiano e inglese (traduzioni)
Licenza	CC0 1.0 Universal By marking the work with a CC0 public domain dedication, the creator is giving up their copyright and allowing reusers to distribute, remix, adapt, and build upon the material in any medium or format, even for commercial purposes.

[1] «Empfehlung zur Bundesinformatik» secondo [P035], n. 4.6

[2] Riguardo alla classificazione AD USO INTERNO e CONFIDENZIALE cfr. ordinanza dell'8 novembre 2023 sulla sicurezza delle informazioni (OSIn, RS **128.1**)

[3] Si veda la nota 1.

[4] Settori di pianificazione secondo la Strategia TIC della Confederazione 2020–2023 del 3 aprile 2020 (SB000)

Indice

1 Scopo 3

2 Panoramica 4

3 Procedura dei chiarimenti preliminari 5

3.1 Rilascio secondo la LMeCA 5
3.2 Eccezione: diritti di terzi 7
3.3 Eccezione: motivi importanti in materia di sicurezza 10
3.4 Chiarimento della pubblicazione 12
3.5 Riduzione al mimino dell'onere 13
3.6 Scelta della lingua di pubblicazione 13

4 Procedura di analisi e preparazione 14

4.1 Analisi del codice sorgente 14
4.2 Scelta della licenza 15
4.3 Documentazione specifica open source 15

5 Procedura di pubblicazione e comunicazione 17

5.1 Scelta della piattaforma 17
5.2 Pubblicazione del software 18
5.3 Comunicazione 18
5.4 Creazione e cura della community 19

Appendice 20

A. Modifiche rispetto alla versione precedente 20
B. Riferimenti 20
C. Abbreviazioni 20
D. Documentazione di accompagnamento della EM002-2.2 «Lista di controllo Analisi e preparazione» – Documentazione 21
E. Documentazione di accompagnamento della «Lista di controllo Analisi e preparazione» – Codice sorgente 24
F. Altre fonti e licenze 27

1 Scopo

La presente guida descrive la procedura di pubblicazione del codice sorgente di software che le autorità federali sviluppano o fanno sviluppare per l'adempimento dei propri compiti secondo l'articolo 9 LMeCA[5]. Fanno eccezione eventuali diritti di terzi o motivi importanti in materia di sicurezza che escludano o limitino tale attività. In seguito alla pubblicazione, chiunque può utilizzare, sviluppare ulteriormente e trasmettere il software. A tal fine non si impone alcun pagamento di diritti di licenza.

La presente guida è rivolta alle persone responsabili della pubblicazione del codice sorgente e che si occupano di eseguirla.

La tematica si suddivide in tre parti principali:

- nella prima parte (n. 3) si trattano i **chiarimenti preliminari** e le eccezioni secondo la LMeCA;
- nella seconda parte sull'**analisi e la preparazione** (n. 4) si esamina il codice sorgente già esistente e lo si rielabora, se necessario.
  In seguito avviene la scelta della licenza.
- La terza parte (n. 5) sulla **pubblicazione e la comunicazione** descrive la pubblicazione vera e propria e le ulteriori misure, come la comunicazione e la creazione di una community adeguata.

La presente guida comprende anche **tre liste di controllo** che accompagnano e documentano la procedura.

L'appendice riporta informazioni aggiuntive di tipo tecnico.

[5] Legge federale *[LMeCA2023]*

2 Panoramica

Quando si parla di rilascio di software a codice sorgente aperto, occorre distinguere fra il **contributo di codice sorgente e documentazione apportato a un software a codice sorgente aperto già esistente** e la **pubblicazione di un proprio progetto open source**.

Nel primo caso, tipicamente si tratta di correzioni di errori (bug fixes) e ampliamenti delle funzionalità (nuove caratteristiche). Il codice sorgente interessato deve o può, a seconda del tipo di licenza e dell'impiego del software, essere rilasciato con la licenza esistente del progetto open source. Può eventualmente essere necessario rispettare un accordo sul rilascio.

Nel secondo caso, ovvero l'avvio di un nuovo progetto open source, la governance e la licenza possono in linea di principio essere scelte liberamente. Si deve solo prestare attenzione a quale licenza sono sottoposti eventuali elementi software che sono integrati nel nuovo progetto. Ulteriori dettagli sulla scelta della licenza sono esposti nel documento «*Em002-3 Guida Licenze OSS*». Se si può creare un valore aggiunto per l'Amministrazione federale a partire da una community o se l'Amministrazione federale vuole creare un ecosistema, si deve compilare anche il documento «*Em002-4.1 Lista di controllo Community OSS*» secondo il documento «*Em002-4 Guida Community*».

Il grafico seguente illustra le istruzioni e diverse caratteristiche.

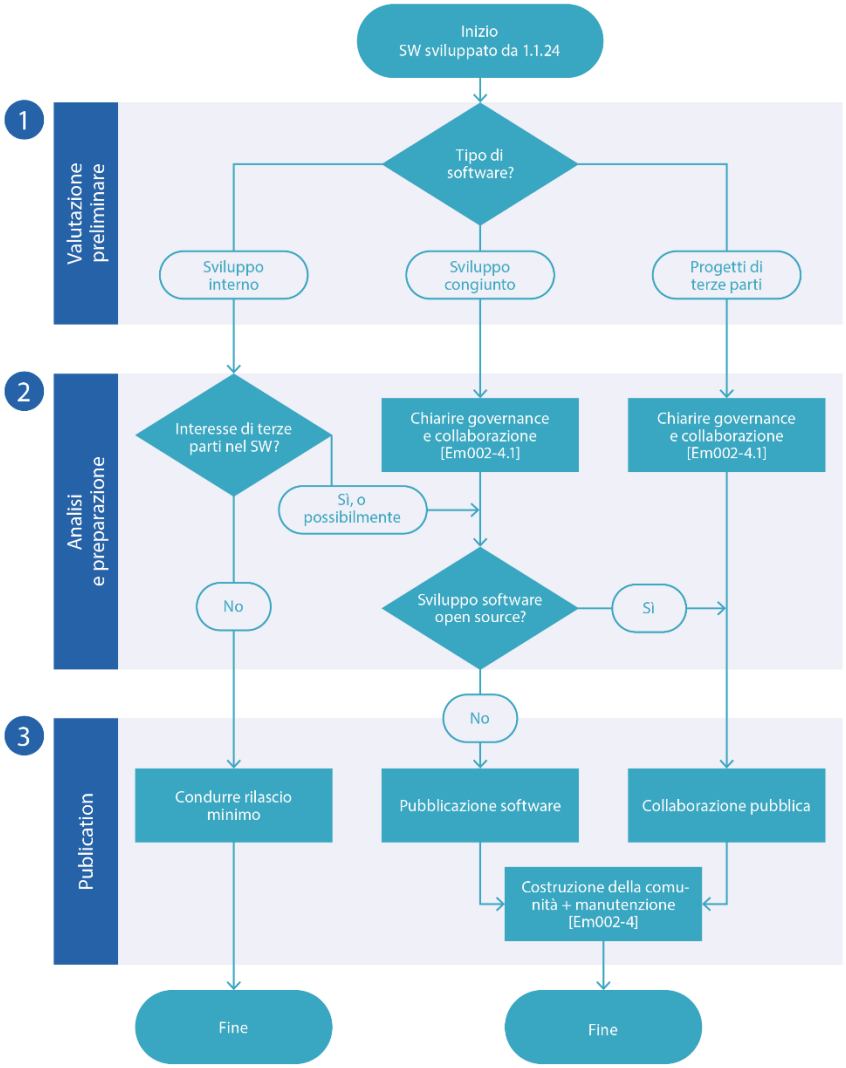

Figura 1 – Albero decisionale relativo al rilascio di software

3 Procedura dei chiarimenti preliminari

Scopo: vengono chiariti i vantaggi e le possibili conseguenze di una pubblicazione. Decidere se il software può essere pubblicato e, in caso affermativo, se è utile costituire una community attiva. Attenzione: si dovrebbe compilare il documento «*Em002-2.1 Lista di controllo Chiarimenti preliminari*» il più presto possibile nel progetto poiché può influire sul tipo di sviluppo del software.

3.1 Rilascio secondo la LMeCA

Secondo l'articolo 9 della legge federale del 17 marzo 2023 [*LMeCA2023*] concernente l'impiego di mezzi elettronici per l'adempimento dei compiti delle autorità (LMeCA), l'Amministrazione federale (art. 2 cpv. 1) è tenuta a pubblicare il codice sorgente dei software che sviluppa o fa sviluppare per l'adempimento dei propri compiti. Fanno eccezione eventuali diritti di terzi o motivi importanti in materia di sicurezza che escludano o limitino tale attività.

Se si tratta di un software individuale creato su incarico della Confederazione, si applicano in linea di massima le CGC della Confederazione [*UFCL-CGC*], che al punto 25.1 stabiliscono che la proprietà del codice sorgente e della documentazione passano al beneficiario della prestazione (la Confederazione). Se il software è stato acquistato congiuntamente da più organizzazioni o altre istituzioni, allora nella maggior parte dei casi la proprietà è di un'associazione fondata a tale scopo.

Il mero utilizzo di software standard non è contemplato dall'articolo 9 LMeCA e, inoltre, un rilascio in questo senso non sarebbe nemmeno possibile in quanto i diritti di software standard spettano spesso al produttore (CGC della Confederazione sull'acquisto di software standard). Ampliamenti individuali di software standard sono invece adatti anche alla pubblicazione e normalmente sono di proprietà della Confederazione. Sono meno adatte alla pubblicazione le semplici modifiche di configurazione.

3.1.1 Software in riferimento alla LMeCA

La norma ISO/IEC 24765 attuale contiene tre definizioni del concetto di software:

1. Un software è un programma o un insieme di programmi che servono a far funzionare un computer.
2. I software sono programmi e la relativa documentazione.
3. I software sono programmi ed eventualmente la relativa documentazione e ulteriori dati necessari al funzionamento di un computer.

Sulla base di questa definizione, si devono classificare come software anche piccoli script, macro o «infrastructure as code» (IaC). Nell'articolo 9 capoverso 1 la LMeCA fa riferimento al codice sorgente. Secondo questa interpretazione sarebbe sufficiente una pubblicazione secondo la definizione 1.
Tuttavia, una pubblicazione priva di relativa documentazione avrebbe poco senso poiché il software non può essere utilizzato secondo la definizione 2.

Per poter ottenere sinergie con terzi, occorre basarsi sulla terza definizione. Con «dati» si intendono in particolare i dati di base e le configurazioni di base (p. es. valori di elenchi e dati di base che dovrebbero essere messi a disposizione anche come open source).

Se ritenuto opportuno dall'autorità federale, possono essere pubblicati insieme in un unico «repository» anche progetti minori, script o esempi di codici. In tal caso, la presente guida e le relative liste di controllo possono essere compilate una volta sola.

Si deve porre attenzione alla proporzionalità e agli oneri riguardo a cosa si pubblica e in che modo.

3.1.2 Tipo di software

Secondo la Figura 1 si distingue fra sviluppo proprio, sviluppo congiunto e progetti di terzi. In base alla situazione, le conseguenze possono essere diverse.
Indipendentemente dal modo in cui viene sviluppato il software, questo rientra nell'articolo 9 LMeCA.

Nello sviluppo in proprio, la Confederazione stessa crea il software o assegna tale incarico, scegliendo la governance e mantenendo i diritti sul codice sorgente.

Nello sviluppo congiunto, la Confederazione crea il software insieme ad altre organizzazioni. In questo caso, la governance e i diritti sul software si devono disciplinare per mezzo della forma organizzativa selezionata.

Il codice sorgente rientra nell'ambito d'applicazione dell'articolo 9 LMeCA anche se la Confederazione contribuisce a software di terzi. In questo caso occorre prestare attenziona al fatto che questo avvenga secondo le intenzioni dell'autorità federale, che la partecipazione sia conforme ai requisiti del progetto e che l'Amministrazione federale si attenga alla governance.

A tal fine si applicano i documenti «Em002-2.1 *Lista di controllo Chiarimenti preliminari*» e «*Em002-4.1 Lista di controllo Community OSS*».

3.1.3 Legacy code

Nel caso di vecchi software (**software legacy**), il dispendio a posteriori per la pubblicazione è maggiore rispetto a quando si pianifica un rilascio fin dall'inizio. Per i software legacy è ragionevole dare avvio all'attività soltanto nel momento in cui un potenziale utente abbia effettivamente intenzione di utilizzarlo. A prescindere da questo, si dovrebbe compilare la «*Em002-2.1 Lista di controllo Chiarimenti preliminari*» per documentare le rispettive decisioni.

Le applicazioni che le autorità federali hanno iniziato a sviluppare a partire dal 1° gennaio 2024 o quelle che sono state sviluppate su incarico della Confederazione sulla base di un contratto stipulato dopo tale data non sono considerate legacy e rientrano in ogni caso nell'obbligo di pubblicazione secondo l'articolo 9 capoverso 1 LMeCA.

L'obbligo di pubblicazione secondo la LMeCA non dipende dall'utilizzo di terzi.

3.1.4 Librerie, plug-in e add-on

In determinate circostanze un software non è un'applicazione indipendente, ma di parti di essa. La LMeCA e l'ordinanza non definiscono nel dettaglio il termine «codice sorgente». La presente guida vale anche per librerie separate, plug-in e add-on, che rientrano nel campo d'applicazione della LMeCA.

Attività:

- reperire le informazioni dalla «*Em002-5 Scheda informativa LMeCA e OSS*», che contiene informazioni sia generali sull'argomento dell'open source sia specifiche sul rapporto con la LMeCA;
- verificare che siano presenti i presupposti stabiliti dalla LMeCA;
- compilare la «*Em002-2.1 Lista di controllo Chiarimenti preliminari*». Di norma è opportuno anche coinvolgere direttamente le persone di contatto del fornitore o dello sviluppatore del software;
- se necessario, compilare la «*Em002-4.1 Lista di controllo Community OSS*».

Decisione: si deve pubblicare il software e come deve verificarsi questo a grandi linee?

3.2 Eccezione: diritti di terzi

Se la pubblicazione lede i diritti di terzi, si deve rinunciare a eseguirla. Se il software è stato sviluppato da collaboratori della Confederazione, i diritti spettano al datore di lavoro[6]. Nei contratti per la fornitura di personale e di servizi IT, in genere i diritti vengono ceduti alla Confederazione e dovrebbero essere reclamati.

Se un software è oppure è stato sviluppato sulla base delle Condizioni generali di contratto della Confederazione (stato 2024)[7], i diritti di protezione sono di proprietà del committente, salvo disposizioni contrattuali contrarie.

Un'applicazione è tipicamente formata da numerosi elementi e componenti singoli. In alcuni tipi di questi è un problema se i componenti stessi non sono open source o se in quanto parti di un'applicazione vengono a loro volta pubblicati con una licenza open source.

Librerie/library

Le librerie (in inglese «library»), in base al linguaggio di programmazione, vengono compilate direttamente insieme all'applicazione, inserite come link o consegnate insieme come pacchetti. Esse costituiscono parte integrante dell'applicazione.

Se l'applicazione contiene librerie proprietarie o con obbligo di licenza, ricorrere all'open source è più difficile. In linea di massima il codice sorgente del resto dell'applicazione può essere pubblicato, ma i potenziali utenti e sviluppatori dovrebbero acquisire anche la rispettiva libreria prima che l'applicazione diventi utilizzabile o ampliabile.

Se il proprietario della libreria problematica è il fornitore dell'applicazione, vale la pena verificare con un accertamento giuridico se alla Confederazione non spetti un diritto trasferibile a terzi su tale libreria.

Se possibile, nello sviluppo proprio andrebbero evitate le librerie proprietarie e con obbligo di licenza oppure andrebbero sostituite prima del rilascio. Considerando lo scopo dell'articolo 9 LMeCA, allora si dovrebbe pubblicare quanta più funzionalità possibile fra quella utilizzata dalle amministrazioni pubbliche. Pertanto, singole librerie che impediscono la pubblicazione del codice rappresentano una limitazione tecnica dell'architettura che va documentata e, in determinate circostanze, eliminata quando se ne presenta l'opportunità.

Banche dati e altri supporti di memoria

L'impiego da parte dell'applicazione di supporti di memoria proprietari con obbligo di licenza, come per esempio Microsoft SQL Server, non rappresenta un ostacolo alla pubblicazione come open source.

Si dovrebbe tuttavia verificare se in aggiunta si possono per esempio supportare anche banche dati aperte come PostgreSQL. Ciò consente di risparmiare costi d'esercizio e ridurre gli ostacoli iniziali per i potenziali utenti.

Application server e sistemi operativi

Se l'applicazione necessita di sistemi operativi o application server con obbligo di licenza (p. es. Windows Server di Microsoft o RedHat JBoss), non si ha alcun un ostacolo alla pubblicazione come open source.

---

[6] Art. 332 CO in combinato disposto con art. 6 cpv. 2 LPers.

[7] Condizioni generali per i contratti d'appalto nel settore informatico e la manutenzione di software individuali, in particolare punto 25.1 - https://www.bkb.admin.ch/it/condizioni-generali-di-contratto-della-confederazione

Si raccomanda di porre al produttore di software la seguente domanda: «Come elemento dell'applicazione si utilizzano librerie con obbligo di licenza di produttori terzi? Se sì, quali?»

Se il software è stato acquistato congiuntamente da più organizzazioni o altre istituzioni, la proprietà è perlopiù di un'associazione fondata a tale scopo o di un'altra persona giuridica.

Altre proprietà intellettuali

I diritti di terzi non comprendono solo i diritti d'autore, bensì anche altre proprietà intellettuali (marchi, brevetti). In generale i brevetti non ostacolano la pubblicazione come open source, ma possono essere d'intralcio all'utilizzo o all'ampliamento del software. I brevetti sono piuttosto inusuali, almeno in Svizzera; tuttavia, è utile chiedere brevemente al produttore di software se è già stata effettuata una verifica.

Acquisizione dei diritti necessari

Per adempiere la disposizione di legge della LMeCA, l'autorità federale ha la possibilità, in quanto committente, di garantire per sé stessa i diritti sui risultati del lavoro. Questa condizione per la pubblicazione può essere definita come criterio di aggiudicazione già al momento dell'acquisto. Acquisire successivamente il diritto necessario sul software può diventare più oneroso.

Se l'autorità federale non vuole pubblicare il software da sé, può delegare questo compito per esempio al mandatario oppure può cedere di conseguenza i diritti per la pubblicazione. In tal caso, trasferirà i diritti necessari (e gli obblighi) al fornitore e/o a terzi incaricandoli della pubblicazione. Ciò significa che il diritto viene effettivamente trasferito a terzi a condizione del rilascio/supporto del software corrispondente.

Procedura

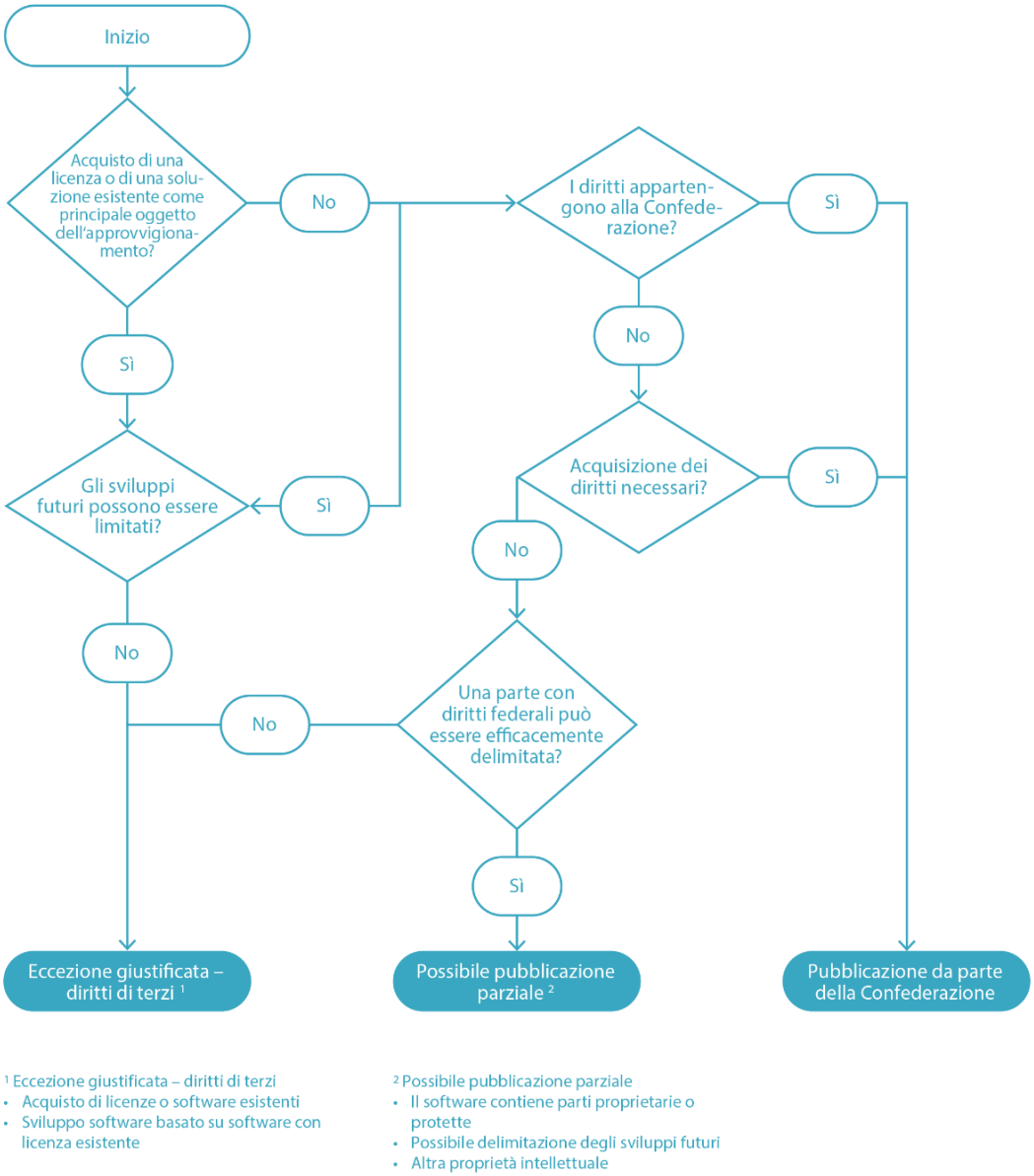

Figura 2 – Eccezione: diritti di terzi

Attività:

- verificare se l'organizzazione possiede i diritti di protezione corrispondenti sul software;
- verificare se si possono acquisire i diritti;
- se necessario, ottenere l'approvazione scritta del titolare dei diritti;
- verificare che l'applicazione non contenga parti proprietarie o protette;
- verificare che non sussistano ostacoli dovuti a una protezione brevettuale (per quanto possibile e ragionevole);
- verificare se il rilascio deve avvenire a opera di terzi (come anticipazione del n. 3.1);
- verificare se lo sviluppo deve avvenire secondo l'«open source software development» (con il documento «*Em002-4 Guida Community*» cfr. anche n. 3.4).

Decisione: il software può essere pubblicato senza ledere i diritti di terzi?

3.3 Eccezione: motivi importanti in materia di sicurezza

È escluso dalla pubblicazione un software che non può essere pubblicato per motivi importanti in materia di sicurezza.

I dati veri e propri di un software non sono interessati dalla pubblicazione del codice sorgente. Questi dati sono in genere degni di protezione e non vengono pubblicati. Può però succedere che, oltre ai dati contenuti nel software, non debbano diventare pubblici nemmeno determinati algoritmi e procedure visibili nel codice sorgente (p. es. capacità informatiche offensive, dettagli sulla scoperta di frodi). La sicurezza può fare riferimento alla confidenzialità, all'integrità, alla disponibilità o alla tracciabilità.

Raccomandazione

Il software dovrebbe essere consapevolmente sviluppato in modo da non generare alcun rischio aggiuntivo anche al momento della pubblicazione del codice sorgente. L'ipotesi secondo cui evitando la pubblicazione ci si protegga da attacchi che vanno a buon fine è ingannevole («security through obscurity»).

Procedura

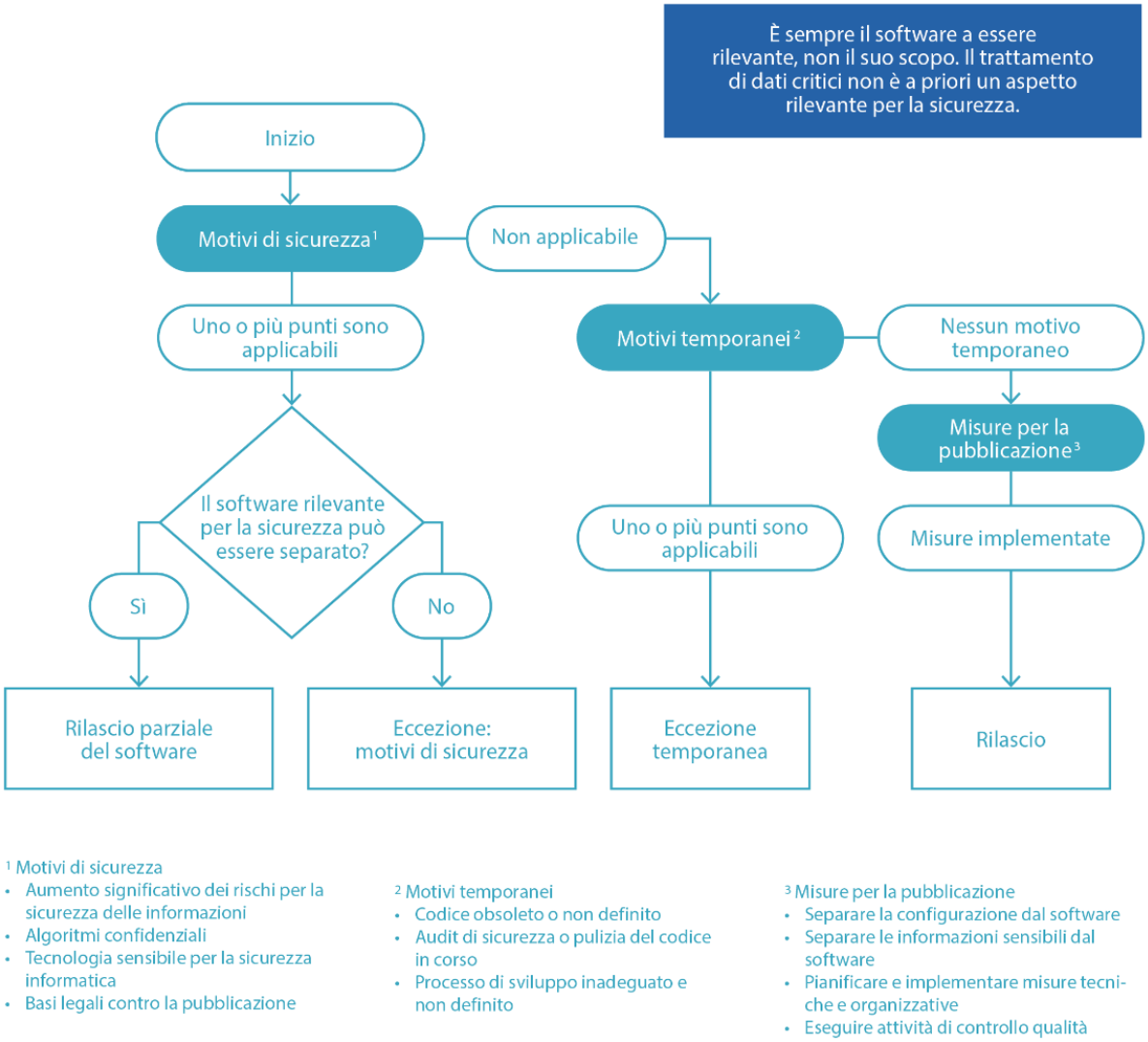

Figura 3 – Eccezione: motivi importanti in materia di sicurezza

In ogni singolo progetto è possibile far valere come importanti in materia di sicurezza i seguenti motivi:

- con la pubblicazione di un software, il rischio legato alla sicurezza delle informazioni aumenta notevolmente[8];
- nel caso degli algoritmi per la lotta alle frodi e simili è possibile, in determinate circostanze, rinunciare alla pubblicazione;
- tecnologie di cibersicurezza particolarmente degne di protezione;
- software centrali per l'esercizio di infrastrutture critiche;
- nel caso in cui la pubblicazione violerebbe altre basi legali.

Anche i seguenti motivi temporanei possono impedire una pubblicazione, ma grazie a opportune misure è possibile procedere successivamente:

- non è ancora stato possibile rettificare il legacy code, che può contenere problemi rilevanti per la sicurezza;
- le verifiche di sicurezza del codice non sono ancora state eseguite;
- il processo di sviluppo non è ancora abbastanza maturo per poter essere pubblicato in sicurezza. Se il processo di sviluppo non è ancora sufficientemente consolidato, è possibile pubblicare parti non destinate a terzi. Per esempio, le linee guida per i commenti sul codice e la generazione di dati di test devono essere adattati alla nuova situazione iniziale con la pubblicazione e funzionare in maniera affidabile.

Attuando le seguenti misure, è possibile pubblicare il software nonostante motivi rilevanti per la sicurezza:

- separare le configurazioni (p. es. tipo di cifratura impiegata) dal software;
- trasferire in configurazioni le informazioni sensibili dal punto di vista tecnico (p. es. processi, calcoli) che non sono pubbliche;
- pianificare adeguate misure organizzative, tecniche e in materia di personale, implementarle e esercitarsi regolarmente;
- al momento della pubblicazione, programmare ed eseguire adeguate attività relative alla qualità, cfr. il capitolo seguente «Procedura di analisi e preparazione».

Attività:

- verificare se il software presenta motivi importanti in materia di sicurezza che potrebbero impedire la pubblicazione;
- valutare l'onere per l'eliminazione dei motivi temporanei;
- pianificare misure volte alla separazione delle informazioni rilevanti per la sicurezza dal codice sorgente del software;
- assicurarsi che l'applicazione non contenga processi che non devono essere pubblicati.

Decisione: sono presenti motivi vincolanti importanti in materia di sicurezza che rendono impossibile la pubblicazione?

[8] Legge sulla sicurezza delle informazioni, LSIn, art. 6 Sicurezza delle informazioni https://www.fedlex.admin.ch/eli/cc/2022/232/it#art_6

3.4 Chiarimento della pubblicazione

In questa fase si chiarisce chi pubblica il software. Se il software è stato creato da un fornitore esterno, è possibile incaricare lo stesso fornitore della pubblicazione. In tal caso occorre stabilire dove pubblicare il codice sorgente; in questa fase è importante disciplinare di conseguenza la governance.

Se il software è stato sviluppato internamente alla Confederazione ma quest'ultima non può o non vuole occuparsi della pubblicazione, è possibile un incarico corrispondente a un fornitore o cedere i diritti (trasferimento degli obblighi a terzi).

Rilascio minimo e supporto

La LMeCA non comprende prescrizioni in merito alla pubblicazione; pertanto, un rilascio minimo del codice sorgente soddisfa le disposizioni di legge. Non si presuppongono ulteriori attività. Alle domande se sia necessario offrire supporto, in che misura e quanto attivamente sia aggiornato il progetto è possibile rispondere basandosi sul documento «*Em002-4 Guida community OSS*».

Open source software development (OSSD)

Con «open source software development» si intende lo svolgimento pubblico dell'intero processo di sviluppo, in aggiunta alla pubblicazione del codice sorgente. Tutto il processo, dai requisiti (issue) fino al codice sorgente, è trasparente. L'OSSD è supportato da strumenti di comunicazione pubblici (mailing list, forum ecc.), un sistema di controllo delle versioni (Git), elenchi di bug e delle caratteristiche, roadmap e strumenti per gli sviluppatori. La modalità di lavoro trasparente va a vantaggio dell'intera community e consente una collaborazione al di là dei confini dell'organizzazione. Grazie a questa procedura, con la chiusura del progetto non sono nemmeno necessarie ulteriori attività relative alla pubblicazione, essendo questa stata considerata già all'inizio dello stesso.

Attività:

- redigere una valutazione sommaria degli oneri per la fase di analisi. In base alle dimensioni e alla complessità dell'applicazione, sono necessari da tre giorni a due settimane;
- rilascio delle risorse per i passi successivi, soprattutto per l'analisi;
- consenso di fondo per la pubblicazione dell'applicazione come open source.

Decisione: sussistono il consenso di fondo e la fattibilità per pubblicare l'applicazione come open source.

3.5 Riduzione al mimino dell'onere

Le liste di controllo e le istruzioni sono strutturate in modo tale da poter ridurre al minimo l'onere per il rilascio. Un'opportuna pianificazione anticipata del rilascio già all'inizio del progetto e uno sviluppo già orientato al rilascio riducono al minimo l'onere. Determinati oneri e costi amministrativi si verificheranno in qualsiasi caso [Le2023]. In determinate circostanze, questi si possono recuperare grazie a community con suddivisione dei costi per lo sviluppo (cfr. «*Em002-4 Guida community OSS*»).

3.6 Scelta della lingua di pubblicazione

Nella scelta della lingua si devono considerare, da un lato, i potenziali destinatari e, dall'altro, la modalità di lavoro del team. Fortunatamente, grazie ai nuovi strumenti, oggi è più facile svolgere le traduzioni. Le lingue prese in considerazione sono quelle ufficiali[9] della Confederazione e l'inglese. Se il progetto ha rilevanza internazionale è bene orientarsi sull'inglese. In caso di pubblicazione a posteriori è evidentemente poco ragionevole cambiare la lingua.

Si dovrebbe rendere disponibile in più lingue almeno il README.md così che le persone interessate possano riconoscere subito se il progetto è rilevante per loro.

In caso di comunicazione pubblica si devono innanzitutto considerare i propri destinatari. Occorre tenere inoltre presente che si tratta di comunicazione pubblica della Confederazione.

Nota: con la pubblicazione, l'Amministrazione federale si presenta al pubblico con un nuovo canale. Se ciò non avviene in maniera accurata e professionale, la reputazione della Confederazione ne risente rapidamente.

[9] https://www.fedlex.admin.ch/eli/cc/1999/404/it#art_70

4 Procedura di analisi e preparazione

Obiettivo: i lavori necessari alla pubblicazione come open source sono svolti.
Le decisioni sulla licenza e sul tipo di community sono prese. Questo avviene con l'ausilio della «*Em002-2.2 Lista di controllo Analisi e preparazione*».

4.1 Analisi del codice sorgente

Il codice sorgente da pubblicare e ulteriori documenti correlati vengono analizzati prima della pubblicazione. Quest'analisi serve a garantire che non vengano pubblicate informazioni degne di protezione.

Se si programma di pubblicare un codice sorgente esteso, per le attività seguenti si raccomanda di utilizzare gli opportuni strumenti automatizzati. I dettagli sono definiti alla lettera E.

Attività:

- verificare in che modo sono state applicate la qualità del codice (code quality) e le direttive esistenti
  (p. es. ISO/IEC 25010:2023[10]);
- assicurarsi che non siano contenuti dati sensibili (di test), per esempio dati basati su persone o casi reali;
- aggiornare la documentazione del software (v. appendice);
- eliminare file non più necessari o documenti e dati obsoleti;
- verificare la presenza di segreti/credenziali nel codice sorgente[11];
- svolgere test di sicurezza mirati e in seguito stabilire un programma «bug bounty» (pubblico)[12];
- la verifica di tutte le librerie impiegate in riferimento alle licenze e le liste corrispondenti («attribution») sono redatte;
- descrizione del deployment (pipeline CI/CD) ed eventualmente creazione di una versione demo.

Queste attività sono indipendenti dalla pubblicazione e rappresentano anzi i presupposti per la buona qualità del software e il rispetto della compliance. L'osservanza di queste misure semplifica la collaborazione, accresce la qualità dei contributi esterni e migliora l'immagine del software e dell'autorità.

[10] ISO/IEC 25010:2023 - Systems and software Quality Requirements and Evaluation (SQuaRE) https://www.iso.org/standard/78176.html

[11] https://www.ncsc.admin.ch/ncsc/it/home/aktuell/im-fokus/2022/git.html

[12] https://www.ncsc.admin.ch/ncsc/it/home/infos-fuer/infos-it-spezialisten/themen/bug-bounty-programme.html

4.2 Scelta della licenza

Utilizzare l'ausilio «*Em002-3 Guida Licenze OSS*»

Per quanto possibile e opportuno, si devono utilizzare testi di licenze affermati a livello internazionale. Deve essere esclusa qualsiasi pretesa in materia di responsabilità da parte dei licenziatari, sempre che ciò sia consentito dal profilo giuridico.

Attività:

- verificare eventuali dipendenze da licenze esistenti;
- in caso di codice di terzi, verificare la compatibilità delle condizioni di utilizzo con la licenza prevista e se necessario sostituire il codice;
- selezionare la licenza adatta.

Come ausilio, la compatibilità delle licenze può essere verificata, per esempio, su Black Duck[13], FOSSA[14] o FOSSology[15]. TODO Group[16] offre una panoramica aggiornata degli strumenti.

Decisione: stabilire e documentare con quale licenza si pubblica il software.

4.3 Documentazione specifica open source

Insieme alla pubblicazione, in genere ci si attende anche un set minimo di documenti specifici. Nell'ambiente open source si sono ormai consolidati alcuni documenti. In questo modo, il visitatore può farsi rapidamente un'idea del software e del tipo di licenza.

Di seguito i documenti auspicati:

Documento	**Descrizione / contenuto**
README(.md)	Serve come documento introduttivo e offre una rapida panoramica su scopo, entità, stato, licenza e gruppi di destinatari del progetto.
LICENSE	Descrive la licenza scelta nel progetto.
CONTRIBUTING(.md)	Descrive il processo secondo cui si può contribuire al progetto.
CODE_OF_CONDUCT(.md)	Il «code of conduct» stabilisce le regole sociali che ci si aspetta nel quadro del progetto.
CHANGELOG(.md)	Presenta un elenco delle modifiche alla versione del software.
THIRD-PARTY-LICENSES(.md)	Descrive le licenze di parti terze o dei componenti utilizzati.
Getting started	Breve guida su come si può utilizzare e installare il software.

[13] https://www.blackducksoftware.com
[14] https://www.fossa.com
[15] https://www.fossology.org
[16] https://landscape.todogroup.org/

Gitignore	Descrive i documenti che non sono pubblicati. Può trattarsi per esempio di configurazioni, dati di test o contenuti generati. Le informazioni sensibili e degne di protezione dovrebbero essere gestite all'esterno del progetto in strumenti e memorie appositamente previsti.

Tabella 1 – Documenti specifici open source

Ulteriori dettagli tecnici su questi documenti sono descritti in appendice.

Attività:

- redigere la documentazione standard;
- dotare tutti i file di indicazioni sul copyright e sulla licenza nonché sull'esclusione della responsabilità;
- a seconda della piattaforma di pubblicazione si possono indicare ulteriori descrizioni.

Decisione: decisione definitiva sulla pubblicazione dell'applicazione come open source.

5 Procedura di pubblicazione e comunicazione

Obiettivo: l'applicazione è pubblicata come open source ed è facilmente reperibile per terzi. Tutti i componenti sono pronti per la pubblicazione e la community è costituita o fondata. Questo avviene con l'ausilio della «*Em002-2.3 Lista di controllo Rilascio e pubblicazione*». In parte si tratta anche di ricapitolare le decisioni già prese.

5.1 Scelta della piattaforma

Esistono diversi modi di pubblicare un codice sorgente. La tabella seguente offre una panoramica di possibili «source code management» (SCM) e piattaforme.

Prima dell'effettiva pubblicazione occorre selezionare la piattaforma adatta. Alcune autorità federali sono già presenti su GitHub[17]. In questo caso è ragionevole ricorrere a piattaforme già consolidate.

Raccomandazione

Attualmente, per pubblicare un codice sorgente per un'autorità federale raccomandiamo di creare un'organizzazione su GitHub. In tale organizzazione si possono creare i progetti corrispondenti (repository) e gestire gli utenti con le rispettive autorizzazioni. In futuro si dovrebbe creare una piattaforma federale centralizzata. Per garantire una gestione ottimale degli utenti e mantenere la reputazione della Confederazione Svizzera è importante utilizzare pochi repository e assicurarsi che siano ben gestiti.

Piattaforma	Caratteristica
GitHub[18]	Piattaforma online di Microsoft largamente diffusa. Ospita un gran numero di software a codice sorgente aperto. Oltre alla funzione di SCM, offre anche molti altri strumenti. È disponibile sia nella variante «free» che in quella «enterprise» (abbonamento).
GitLab[19]	Piattaforma online dell'omonima azienda GitLab, a sua volta pubblicata con una licenza open source. Oltre alla funzione di SCM, offre anche molti altri strumenti. Può funzionare anche in locale e offre perciò una certa sovranità. È disponibile sia nella variante «free» che in quella «enterprise» (abbonamento).
Bitbucket[20]	Piattaforma online di Atlassian, integrata soprattutto nel rispettivo ecosistema. È disponibile sia nella variante «free» che in quella «enterprise» (abbonamento).
Sistema di SCM interno	Se si mette a disposizione una propria piattaforma SCM, questa può anche pubblicare e rendere accessibili certi repository/progetti.

[17] Il sito web https://ossbenchmark.com presenta una panoramica delle organizzazioni e autorità attive su GitHub.

[18] https://github.com/about/

[19] https://about.gitlab.com/

[20] https://bitbucket.org/product/it

	Bisogna a tal fine garantire il funzionamento.
Piattaforma della Confederazione	Allo stato attuale non esiste ancora una piattaforma centralizzata della Confederazione per la pubblicazione di software. Il documento «*Em002 Guida strategica Software a codice sorgente aperto nell'Amministrazione federale*» propone una misura in questo senso.
Sito web, server FTP pubblici ecc.	La LMeCA non definisce in che modo il software debba essere pubblicato. Si può eseguire un rilascio minimo secondo il numero 3.4 anche su un sito web o altre risorse accessibili al pubblico. Tuttavia, questa modalità non è adatta a una collaborazione sostenibile nell'ottica dell'open source.

Tabella 2 – Piattaforme per la pubblicazione di software a codice sorgente aperto

Se il software viene pubblicato insieme a un partner esterno o a un'organizzazione, si deve prestare attenzione al fatto che siano presenti le necessarie autorizzazioni per l'accesso al codice sorgente.

Attività:

- valutare la piattaforma adatta incl. la rispettiva organizzazione/il rispettivo progetto;
- se non ancora chiarito, definire la denominazione del progetto/del repository;
- stabilire e verificare l'autorizzazione per il repository;
- garantire la governance adeguata e l'eventuale replicazione del codice sorgente.

Decisione: la piattaforma per la pubblicazione è stata scelta.

5.2 Pubblicazione del software

Se i presupposti sono stati chiariti, il software può essere pubblicato.

Attività:

- verificare i presupposti secondo la lista di controllo sul rilascio e sulla pubblicazione (Em002-2.3);
- l'accesso alla piattaforma è disponibile e la governance è chiarita;
- lo sviluppatore pubblica il codice sorgente e la documentazione.

5.3 Comunicazione

In seguito all'effettiva pubblicazione si possono avviare ulteriori attività di comunicazione al di fuori e all'interno dell'autorità federale interessata. Tali attività contribuiscono alla diffusione e apportano il valore aggiunto auspicato dalla pubblicazione.

Attività:

- inserire una descrizione nel repository hosting, impostazioni di sicurezza ecc.
- occuparsi della comunicazione verso l'interno e l'esterno;
- redigere una comunicazione per la community;
- rendere nota l'applicazione all'interno dell'organizzazione e negli organi specializzati, pubblicizzare l'impiego e i contributi.

5.4 Creazione e cura della community

Con la pubblicazione di software può svilupparsi una community che può collaborare al software tramite contributi (pull request), domande, proposte di documentazione ecc. Per disporre di una community attiva, è necessario uno sforzo adeguato per crearla e curarla costantemente. Una community che funziona bene riduce al minimo le scissioni indesiderate (fork) del software. In questo contesto l'obiettivo è quello di attivare il valore aggiunto della community attraverso attività mirate. Anche in caso di una pubblicazione minima si dovrebbe definire come gestire i riscontri (feedback) e le domande sul software.

Attività:

- chiarire chi sono i potenziali utenti o interessati;
- basandosi sulla Em002-4 «*Guida Community OSS*», determinare la forma adatta per l'applicazione concreta;
- creare la community;
- programmare attività di community building.

Appendice

A. Modifiche rispetto alla versione precedente

Documento nuovo.

B. Riferimenti

Cfr. «*Em002 Guida strategica Software a codice sorgente aperto nell'Amministrazione federale*».

C. Abbreviazioni

Cfr. «*Em002 Guida strategica Software a codice sorgente aperto nell'Amministrazione federale*» e «*Em002-6 FAQ OSS e articolo 9 LMeCA*».

D. Documentazione di accompagnamento della EM002-2.2 «Lista di controllo Analisi e preparazione» – Documentazione

D. 1. Documentazione del codice sorgente

La documentazione dovrebbe essere parte del codice sorgente del progetto, in modo che anche le modifiche alla documentazione siano conservate correttamente e la versione della documentazione sia analoga a quella del progetto. Ciò garantisce automaticamente che la documentazione, purché aggiornata come opportuno, non diverga dallo stato di sviluppo del progetto e che sia possibile consultare in ogni momento anche la documentazione delle versioni più vecchie del progetto.

Inoltre, in questo modo il progetto dà la possibilità a terzi di integrare agevolmente modifiche alla documentazione sotto forma di contribuiti.

Come formato di testo si deve usare Markdown come markup language, essendo questo facile da scrivere, ampiamente diffuso e leggibile a macchina. In aggiunta, i documenti in Markdown sono rappresentati in un formato di destinazione ben leggibile da tutte le piattaforme di uso comune. Indipendentemente dalla piattaforma utilizzata, i documenti Markdown possono essere rappresentati in un formato di destinazione qualsiasi con generatori di siti statici come Jekyll, MkDocs, Gatsby o Sphinx, per esempio con le direttive di corporate identity di un'amministrazione.

D. 2. Destinatari della documentazione e struttura

La documentazione del progetto deve rivolgersi sia all'utente finale sia agli specialisti tecnici, perciò non deve focalizzarsi esclusivamente sugli aspetti che riguardano questi ultimi. Per quanto riguarda la scelta della lingua si rimanda al numero 3.6. Questo agevola la diffusione del software.

Un file `README.md` funge da documento introduttivo. I file con questo nome sono riconosciuti e rappresentati da tutte le piattaforme di uso comune come punto di partenza per la documentazione[21].

`README.md` deve offrire una rapida panoramica su scopo, entità, stato, licenza e gruppi di destinatari del progetto. In concreto, `README.md` deve comprendere i seguenti punti:

- nome del software
- breve descrizione di senso e scopo del software; entità delle funzioni e possibili limitazioni
- istruzioni di installazione
- utilizzo del software
- rimando a un'eventuale versione demo
- supporto e informazioni di contatto
- in che modo si può contribuire al software a codice sorgente aperto (contributing)
- licenza utilizzata (license)
- stato del progetto / stato dello sviluppo

[21] Ulteriori informazioni sulla redazione di un «readme»: https://www.makeareadme.com/

D. 3. Messa in evidenza dell'open source e archiviazione della licenza

Immediatamente dopo la dichiarazione d'intenti si deve descrivere in modo chiaro e inequivocabile che il progetto è un software a codice sorgente aperto. La licenza utilizzata dal progetto deve essere indicata con l'identificatore SPDX[22] e nel file LICENSE si deve inserire un collegamento al testo concreto della licenza. I modelli di licenza in genere contengono una parte variabile (p. es. nome del progetto, anno di copyright, nome dell'autore) che deve essere adattata nel file LICENSE.

La maggior parte delle piattaforme riconoscono le licenze in un file `LICENSE` e offrono informazioni ordinate sulla licenza, per esempio un breve riepilogo delle rispettive condizioni.

Si raccomanda di aggiungere a ogni file sorgente che si crea un header minimo di licenza e copyright. Gli header presenti nei file esistenti non dovrebbero essere modificati. Se si contribuisce a un progetto che presenta chiari standard per le intestazioni, questi devono essere rispettati.

Per ulteriori dettagli cfr. «Producing Open Source Software, State That the Project is Free»[23].

D. 4. Stato attuale dello sviluppo

Lo stato attuale dello sviluppo del progetto deve essere documentato in README.md affinché risulti rapidamente chiaro. Per il lettore è infatti importante sapere se il progetto si trova in uno stato avanzato oppure all'inizio dello sviluppo, se viene aggiornato attivamente e con quale frequenza vengono pubblicate nuove versioni.

In questa sezione si può descrivere quale supporto da parte di terzi è più importante per il progetto in un determinato momento. Può trattarsi per esempio di uno sviluppatore con determinate conoscenze tecnologiche o di una persona che rielabora la documentazione del progetto.

Per ulteriori dettagli cfr. «Producing Open Source Software, Development Status»[24].

D. 5. Versione demo

Nel caso delle applicazioni si raccomanda di mettere a disposizione una versione demo affinché l'applicazione possa essere testata senza onere di installazione. A seconda dell'applicazione, la versione demo può essere l'installazione produttiva o una fase di test. È importante che il lettore possa avere accesso per quanto possibile in maniera autonoma e immediata a tale versione.

In alternativa, in molti progetti open source vengono messi a disposizione ausili su come avviare facilmente l'applicazione. Ciò può avvenire per esempio per mezzo di container, applicazioni portabili (portable) o script di installazione.

D. 6. Persone di contatto specializzate e proprietari del progetto

Il file README.md deve indicare il nome della principale persona di riferimento specializzata, affinché altre amministrazioni e organizzazioni interessate possano mettersi direttamente in contatto.

---

[22] Identificatore SPDX - https://spdx.org/about

[23] https://producingoss.com/en/getting-started.html#state-freedom

[24] https://producingoss.com/en/getting-started.html#development-status

Non andrebbero utilizzati indirizzi e-mail personali.

Nel file `README.md` si deve indicare brevemente anche quale organizzazione è proprietaria del progetto, soprattutto se a tal fine è stata creata un'associazione.

D. 7. Documentazione d'installazione

Per le applicazioni, nel file README.md si dovrebbe rimandare a una documentazione d'installazione che descriva i requisiti hardware e software, quali componenti infrastrutturali (p. es. banche di dati) vengono utilizzate e come installare e utilizzare l'applicazione.

D. 8. Developer guidelines

Il punto di partenza delle «developer guidelines» deve essere archiviato in un file Markdown CONTRIBUTING.md che a sua volta deve essere inserito come link nel README.md. Le «developer guidelines» fanno parte della documentazione approfondita destinata agli sviluppatori che vogliono contribuire al progetto; l'accento è posto innanzitutto sulla collaborazione e sulla comunicazione all'interno del progetto e non sugli aspetti tecnici.

Le «developer guidelines» devono quindi contenere i punti seguenti ed eventualmente rimandare ai documenti corrispondenti:

- un rimando al «code of conduct» del progetto. Il «code of conduct» stabilisce le regole sociali che ci si aspetta nel quadro del progetto;
- come «code of conduct» si raccomanda di scegliere il Contributor Covenant Code of Conduct sottoposto alla licenza CC-BY-4.0[25] oppure di basarsi su di esso[26];
- una nota sul fatto che il progetto esegue le revisioni del codice sotto forma di «pull request» di GitHub, con un rimando alla documentazione su queste ultime.

Si deve infine rimandare alla «developer documentation».

D. 9. Developer documentation

Mentre le «developer guidelines» descrivono la collaborazione e le norme sociali del progetto, la «developer documentation» si concentra sugli aspetti tecnici rilevanti per la partecipazione allo sviluppo del progetto. Essa deve descrivere almeno i punti seguenti:

- quali dipendenze tecniche ha il progetto e quali strumenti sono necessari per il suo sviluppo;
- quali regole formali valgono per il codice sorgente del progetto, per esempio riguardo alla formattazione dello stesso;
- come è organizzato il codice sorgente del progetto e come possono essere testate dal punto di vista tecnico le funzioni;
- uno schizzo dell'architettura che la descriva a grandi linee e che per quanto possibile includa una delimitazione del contesto e una panoramica sommaria dei componenti[27];
- per le applicazioni: come si può avviare l'applicazione per scopi di sviluppo o debug e quali componenti infrastrutturali sono necessari.

[25] https://creativecommons.org/licenses/by/4.0/

[26] https://www.contributor-covenant.org/version/1/4/code-of-conduct.html

[27] Framework dell'architettura MMB (metodica di modellizzazione della Confederazione) oppure arc42. http://arc42.org/

E. Documentazione di accompagnamento della «Lista di controllo Analisi e preparazione» – Codice sorgente

E. 1. Cronologia del codice sorgente

Il codice sorgente di un progetto è di norma archiviato in un sistema di source code management (SCM), come per esempio GitHub. Questi sistemi non memorizzano soltanto lo stato attuale del codice sorgente ma anche le modifiche che vi sono state apportate nel corso del tempo. Fra queste rientrano in particolare anche elementi cancellati dal codice sorgente, compresa la cancellazione di file.

Inoltre, l'SCM contiene metainformazioni sulle modifiche al codice sorgente, come per esempio il nome e l'indirizzo e-mail dell'autore o le firme PGP per i commit e i tag firmati.

E. 2. Dati sensibili, degni di protezione o confidenziali

Il codice sorgente del progetto e la sua cronologia possono contenere dati pubblicamente disponibili, come per esempio un elenco di numeri postali di avviamento, dati generati casualmente o dati sintetici.

Il codice sorgente e la sua cronologia non possono invece contenere dati (personali) sensibili, degni di protezione o confidenziali né informazioni secondo la legge sulla protezione dei dati (LPD) e la legge sulla sicurezza delle informazioni (LSIn). Fra questi rientrano in particolare:

- nomi utenti e password o altre informazioni riservate come chiavi private, token di accesso o certificati;
- nomi DNS interni, URL, nomi di host, indirizzi IP, strutture di rete o denominazioni di drive;
- nomi, indirizzi, immagini o altri dati personali senza il consenso della persona interessata;
- dati anonimizzati o pseudonimizzati (perché in caso di anonimizzazione o pseudonimizzazione di dati rimane comunque una possibilità di ripristinare i dati).

Anche se lo stato attuale del codice sorgente è privo di dati sensibili, degni di protezione o confidenziali, questi dati o informazioni possono comunque essere consultati tramite la cronologia del sistema SCM se in passato hanno fatto parte del codice sorgente o delle metainformazioni SCM. Perciò, una pulizia dello stato attuale del codice sorgente non rimuove completamente questo tipo di informazioni.

In caso di dubbio, si raccomanda di cancellare la cronologia del codice sorgente dopo averlo pulito e prima della pubblicazione.

I nomi e gli indirizzi e-mail degli autori del codice sorgente sono di solito presenti direttamente nel codice sorgente o nelle metainformazioni SCM, per esempio nei commit o nei tag annotati di GitHub. Da un punto di vista legale, ciò non rappresenta un problema poiché l'azienda sviluppatrice ha la responsabilità di disciplinare la pubblicazione di queste informazioni. È tuttavia consigliabile far notare all'azienda sviluppatrice o anche ai collaboratori interni alla Confederazione che con la pubblicazione di un software potrebbero essere pubblicati anche i nomi e gli indirizzi e-mail degli autori del codice sorgente.

E. 3. Rimozione di informazioni sugli autori

La cronologia di GitHub può essere modificata in modo da rendere illeggibili le informazioni sugli autori. Quest'operazione può essere automatizzata ricorrendo ad appositi strumenti[28].

Rimuovendo le informazioni sugli autori si perdono l'attribuzione, la tracciabilità e le informazioni di contatto. In base all'entità del codice sorgente non si deve sottovalutare l'onere che questa operazione comporta. Per queste motivazioni, la rimozione degli autori è un'operazione sconsigliata.

E. 4. Considerazione delle licenze delle librerie

Praticamente ogni progetto utilizza librerie (librerie di software, dipendenze) di terzi per poter attingere a funzioni generalmente applicabili senza doverle programmare da sé. Spesso queste librerie dipendono a loro volta da altre librerie e questo porta a dipendenze su più livelli (dipendenze transitive). A seconda del linguaggio di programmazione, specialmente nei progetti più piccoli, centinaia di dipendenze costituiscono la norma.

Il risultato di questo chiarimento deve essere considerato nella scelta della licenza secondo la Em 002-3 «*Guida Licenze OSS*» e la Em002-2.3 «*Lista di controllo Rilascio e pubblicazione*».

Per ognuna di queste dipendenze occorre assicurarsi che la loro licenza sia compatibile con quella del progetto. Sono particolarmente problematiche in tal senso le dipendenze che non sono sottoposte a una licenza open source, che non hanno dichiarato alcuna licenza o che ne utilizzano una virale che non coincide con quella del progetto. Alcune librerie sono sottoposte a più licenze e quindi consentono all'utente di sceglierne una.

- In tutti i linguaggi di programmazione di uso comune le dipendenze del progetto (incluse quelle transitive) nonché le licenze delle dipendenze sono elencate automaticamente. Fra queste rientrano:
- il «Project Info Reports Plugin» del sistema di gestione pacchetti Apache Maven per progetti Java, che tramite il report delle dipendenze crea in formato HTLM tutte le librerie utilizzate, incluse le licenze dichiarate[29];
- il license checker NPM per il sistema di gestione pacchetti NPM di progetti JavaScript, che è in grado di emettere in diversi formati (p. es. CSV) tutte le librerie utilizzate, incluse le licenze dichiarate[30];
- il «license finder» di Pivotal, che supporta diversi sistemi di gestione pacchetti e linguaggi di programmazione, per esempio anche Ruby Gems, Python Eggs o Godeps[31];

Utilizzo di librerie con licenze proprietarie

In generale, le librerie sottoposte a una licenza proprietaria sono problematiche per i progetti open source e non possono perciò essere utilizzate, soprattutto se il codice sorgente viene pubblicato con una licenza virale come la AGPL.

Insieme al servizio giuridico del proprio ufficio occorre mettersi in contatto con il produttore per chiarire se e a quali condizioni la libreria proprietaria può essere impiegata nel progetto in questione.

---

[28] https://www.adamdehaven.com/blog/update-commit-history-author-information-for-git-repository/

[29] Informazioni sul plug-in https://maven.apache.org/plugins/maven-project-info-reports-plugin/plugin-info.html

[30] «License checker» NPM https://github.com/davglass/license-checker

[31] «License finder» di Pivotal https://github.com/pivotal/LicenseFinder

E. 5. Utilizzo di sistemi di gestione pacchetti

Praticamente tutti i linguaggi di programmazione comuni offrono sistemi di gestione pacchetti (package manager) per gestire le proprie dipendenze; per esempio Apache Maven per i progetti Java, NPM per i progetti JavaScript o NuGet per i progetti .NET.

Quando si utilizza uno di questi sistemi, le dipendenze del progetto non sono più direttamente parte del codice sorgente del progetto stesso, per esempio quando si copia il codice sorgente della dipendenza nel progetto oppure copiando le dipendenze nel progetto in forma binaria. Invece, il progetto ottiene tutte le dipendenze tramite dichiarazioni corrispondenti del sistema di gestione pacchetti.

In questo modo è possibile elencare tutte le dipendenze e farvi correttamente riferimento. Inoltre, si impedisce che vengano apportate inconsapevolmente modifiche al codice delle dipendenze. Se sono necessarie modifiche, queste devono essere apportate direttamente alla rispettiva fonte della dipendenza.

E. 6. Dichiarazione della licenza del progetto nel descrittore del sistema di gestione pacchetti

Il progetto dovrebbe dichiarare la propria licenza nel rispettivo descrittore del sistema di gestione pacchetti con l'identificatore SPDX[32] affinché la licenza del progetto possa essere emessa tramite il sistema di gestione pacchetti. Questo è particolarmente importante per le librerie affinché i loro utenti possano consultare automaticamente la licenza tramite il sistema di gestione pacchetti.

E. 7. «Attribution» e note sul diritto d'autore

Molte librerie sono sottoposte a una licenza che richiede a chi le utilizza di riportare una nota sul diritto d'autore (original copyright notice) o di indicare un riconoscimento (attribution) dell'utilizzo della libreria.

Fra queste rientrano le seguenti licenze (estratto):

Apache 2.0, ASL 1.1 (Apache 1.1), BSD, BSD 3-Clause, Creative Commons «Attribution» (CC BY), MIT, ISC

Per altre licenze si veda l'elenco di quelle accreditate OSI[33].

In pratica, ne sono interessate quasi tutte le librerie utilizzate. Poiché già nei progetti di piccole dimensioni il numero di librerie utilizzato è molto elevato, il fatto di inserire una nota sul diritto d'autore che sia impeccabile dal punto di vista giuridico o citare l'attribuzione necessaria risulta molto complicato.

Pertanto si raccomanda di procedere come segue:

- creare un elenco delle librerie utilizzate grazie agli appositi meccanismi del sistema di gestione pacchetti o degli strumenti come il license checker NPM o il license finder di Pivotal;
- elaborare manualmente l'elenco in modo che contenga le seguenti informazioni:

[32] Identificatore SPDX - https://spdx.org/

[33] Open source licenses are licenses that comply with the Open Source Definition – in brief, they allow software to be freely used, modified, and shared. https://opensource.org/licenses

nome ufficiale del progetto della libreria, homepage del progetto della libreria, licenza utilizzata della libreria come identificatore SPDX con un link al testo della licenza;
- archiviare l'elenco nel codice sorgente del progetto, per esempio in un file `THIRD-PARTY-LICENSES.md`;
- per le applicazioni: inserire un link in un «About» o in una finestra di dialogo «Su quest'applicazione» che rimandi al file `THIRD-PARTY-LICENSES.md`;
- l'utilizzo della finestra di dialogo «About» è molto diffuso a questo scopo, per esempio da Google Chrome (nella finestra «About Chrome»);
- per le librerie: nel file `README.md` rimandare al file `THIRD-PARTY-LICENSES.md`;
- come parte del processo di revisione del codice, assicurarsi che tutte le modifiche alle librerie siano aggiunte nel file `THIRD-PARTY-LICENSES.md`.

Esempio di blocco di codice «THIRD-PARTY-LICENSES.md»

```
Questo progetto utilizza software a codice sorgente aperto:

* Spring Boot, http://projects.spring.io/spring-boot/, sotto licenza [Apache-
2.0](http://www.apache.org/licenses/LICENSE-2.0)
* caniuse-db, https://github.com/Fyrd/caniuse, sotto licenza [CC-BY-
4.0](https://creativecommons.org/licenses/by/4.0/)
```

F. Altre fonti e licenze

La lista di controllo, la documentazione di accompagnamento e i modelli corrispondenti si basano in parte sulle seguenti fonti:

- Google Open Source Docs von Google LLC, sotto licenza CC-BY-4.0[34]
- Producing Open Source Software, How to run a Successful Free Software Project di Karl Fogel, sotto licenza CC-BY-SA-4.0[35]
- GitHub Healthy Contributions[36]
- Standard for Public Code[37]

[34] Google Open Source Docs - https://opensource.google.com/docs/

[35] Producing Open Source Software, How to run a Successful Free Software Project - https://producingoss.com/

[36] GitHub Healthy Contributions

[37] Standard for Public Code - https://standard.publiccode.net/