Schweizerische Eidgenossenschaft
Confédération suisse
Confederazione Svizzera
Confederaziun svizra

Cancelleria federale CaF

Trasformazione digitale e governance delle TIC TDT

14.09.2026

Em002-2 Istruzioni per la pubblicazione di OSS

Versione 2.0-31789e3

Indice

Avvertenza: Il presente documento è una bozza in continua evoluzione e fa parte degli strumenti destinati a sostenere l'Amministrazione federale nella pubblicazione di codice open source.[1][2][3][4] Per maggiori informazioni si veda il README principale.

1. Riassunto per la direzione

Le presenti istruzioni descrivono la procedura per la pubblicazione del codice sorgente del software che le autorità federali sviluppano o fanno sviluppare per adempiere i loro compiti, conformemente all'art. 9 LMeCA.[5] Sono previste eccezioni se diritti di terzi o motivi rilevanti per la sicurezza escludono o limitano tale pubblicazione. Pubblicare significa che chiunque può utilizzare, sviluppare e diffondere il software. Non sono riscosse tasse di licenza.

Le presenti istruzioni si rivolgono alle persone responsabili della pubblicazione del codice sorgente e incaricate della sua attuazione.

La materia è suddivisa in tre parti principali:

- La prima parte (capitolo 3) tratta le **valutazioni preliminari** e le eccezioni secondo la LMeCA.
- La seconda parte (capitolo 4), **Analisi e preparazione**, esamina il codice sorgente e lo prepara secondo le necessità. Vi si effettua inoltre la scelta della licenza.

1. Raccomandazione per l'informatica dell'Amministrazione federale secondo [P035] *capitolo 4.6*
2. Per le definizioni delle classificazioni INTERNO e CONFIDENZIALE si veda l'*ordinanza dell'8 novembre 2023 sulla sicurezza delle informazioni in seno alla Confederazione e all'esercito (OSIC; RS 128.1)*
3. Cfr. nota a piè di pagina 1
4. Ambiti di pianificazione secondo la *strategia TIC della Confederazione 2020-2023 del 3 aprile 2020 (SB000)*
5. Legge federale *[EMOTA2023]*

- La terza parte (capitolo 5), **Pubblicazione e annuncio**, descrive la pubblicazione vera e propria e ulteriori misure quali la comunicazione e la costituzione di una community adeguata.

Tre liste di controllo accompagnano e documentano il processo.

Complementi tecnici figurano nell'allegato.

2. Introduzione

Nella pubblicazione di software open source occorre distinguere tra il **contributo di codice sorgente e documentazione a software open source esistente** e la **pubblicazione come progetto open source indipendente**.

Il primo caso comprende tipicamente correzioni di errori ed estensioni funzionali. A seconda del tipo di licenza e dell'impiego del software, il codice sorgente deve o può essere pubblicato sotto la licenza esistente del progetto open source. Può essere necessario rispettare un accordo di pubblicazione.

Nel secondo caso, l'avvio di un nuovo progetto open source, la governance e la licenza possono in linea di principio essere scelte liberamente. L'unico elemento da considerare è la licenza sotto la quale sono pubblicati gli eventuali elementi software integrati nel nuovo progetto. Ulteriori dettagli sulla scelta della licenza figurano nel documento Em002-3 Guida alle licenze OSS. Se da una community può derivare un valore aggiunto per l'Amministrazione federale o se questa intende creare un ecosistema, va inoltre compilata la lista di controllo Em002-4.1 Community OSS conformemente alla Em002-4 Guida alle community OSS.

La pubblicazione di software open source è più di una semplice misura tecnica. Significa anche instaurare una cultura dell'apertura corrispondente e coinvolgere le persone. Una cultura open source di successo è una combinazione di apertura, eccellenza tecnica, forte interazione sociale e impegno condiviso.

Il grafico seguente descrive le istruzioni e le diverse configurazioni.

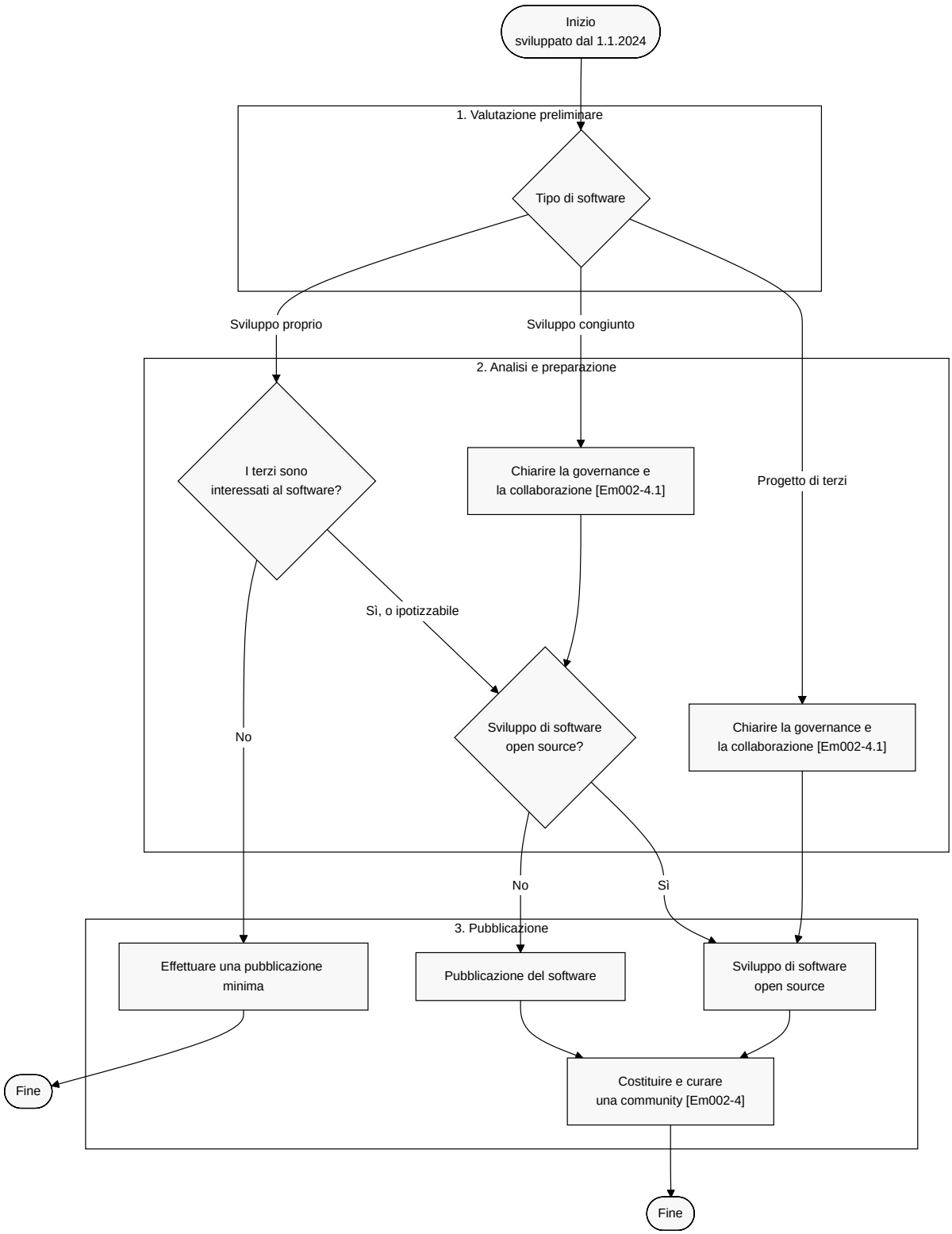

Figura 1 – Albero decisionale per la pubblicazione di software

3. Processo di valutazione preliminare

Obiettivo: comprendere i vantaggi e le possibili conseguenze di una pubblicazione del software. Decidere se il software può essere pubblicato e, in caso affermativo, se la costituzione di una community attiva ne vale la pena. Nota: la lista di controllo Em002-2.1 Valutazione preliminare OSS dovrebbe essere compilata il prima possibile nel progetto, poiché ciò può influenzare il modo in cui il software viene sviluppato.

3.1. Pubblicazione secondo la LMeCA

Secondo l’art. 9 della legge federale del 17 marzo 2023 [EMOTA2023] sull’impiego di mezzi elettronici per l’adempimento dei compiti delle autorità (LMeCA), l’Amministrazione federale (concretamente l’art. 2 cpv. 1) deve pubblicare il codice sorgente del software che sviluppa o fa sviluppare per adempiere i suoi compiti. Si applicano eccezioni se diritti di terzi o motivi rilevanti per la sicurezza lo escludono o lo limitano.

Per il software su misura realizzato su mandato della Confederazione si applicano in linea di principio le condizioni generali della Confederazione [BBL-AGB]. Secondo il loro numero 25.1, la proprietà del codice sorgente e della documentazione passa al servizio acquirente (la Confederazione). Se il software è stato acquistato congiuntamente da più organizzazioni o altre istituzioni, occorre verificare chi detiene i diritti sul codice. La proprietà può per esempio spettare a un’associazione costituita a tale scopo.

Il semplice utilizzo di software standard non rientra nell’art. 9 LMeCA e una pubblicazione non sarebbe comunque di regola possibile, poiché i diritti sul software standard restano spesso al fornitore (CG della Confederazione per l’acquisto di software standard). Le estensioni su misura di software standard si prestano invece pienamente a una pubblicazione. Esse appartengono normalmente alla Confederazione. Meri adattamenti di configurazione si prestano meno a una pubblicazione. Il tema è trattato in Em002-7 Aspetti strategici degli acquisti e del software open source nonché nei documenti [KKB-MB] e [BBL-WL]. In linea di principio le estensioni sottostanno sempre all’art. 9 LMeCA.

3.1.1. Il software in rapporto alla LMeCA

L’attuale norma ISO/IEC 24765 contiene tre definizioni di software:

1. un programma o un insieme di programmi utilizzati per far funzionare i computer
2. i programmi e la relativa documentazione
3. i programmi e, ove applicabile, la documentazione associata nonché altri dati necessari al funzionamento del computer.

Sulla base di questa definizione anche script, macro o Infrastructure as Code (IaC) di minore entità sono classificati come software. La LMeCA si riferisce al codice sorgente all’articolo 9 capoverso 1. Secondo tale interpretazione una pubblicazione ai sensi della definizione 1 sarebbe sufficiente. Una pubblicazione priva della documentazione associata ha tuttavia scarso valore, poiché il software non può essere utilizzato ai sensi della definizione 2.

Per realizzare sinergie con terzi occorre basarsi sulla definizione 3. Riguardo ai dati ciò significa concretamente i dati di base e le configurazioni di base (per esempio valori di enumerazione e dati di base che dovrebbero parimenti essere resi disponibili come open source).

Progetti, script o esempi di codice di minore entità possono anche essere pubblicati congiuntamente in un unico repository, se l’autorità federale lo ritiene opportuno. In tal caso le presenti istruzioni e le corrispondenti liste di controllo possono essere compilate una sola volta.

Nel decidere che cosa pubblicare e come, occorre considerare la proporzionalità e l’onere.

3.1.2. Tipo di software

Come mostra la figura 1, si distingue tra sviluppo proprio, sviluppo congiunto e progetti di terzi. Ogni costellazione comporta implicazioni diverse. Indipendentemente da come il software viene sviluppato, esso rientra nell’art. 9 LMeCA.

Nello sviluppo proprio la Confederazione crea essa stessa il software o lo fa creare di conseguenza. Sceglie la governance e conserva i diritti sul codice sorgente.

Nello sviluppo congiunto la Confederazione crea il software insieme ad altre organizzazioni. La governance e i diritti sul software devono essere disciplinati mediante la forma organizzativa scelta.

Se la Confederazione contribuisce direttamente a software di terzi, anche tale codice sorgente rientra nell'art. 9 LMeCA. Occorre assicurare che ciò avvenga nell'interesse delle autorità federali, che la partecipazione soddisfi i requisiti del progetto e che l'Amministrazione federale rispetti la governance.

Ciò avviene mediante la lista di controllo Em002-2.1 Valutazione preliminare OSS e la lista di controllo Em002-4.1 Community OSS.

3.1.3. Codice preesistente

Per il software datato (**software preesistente**) l'onere successivo per una pubblicazione è maggiore rispetto al caso in cui essa fosse stata pianificata sin dall'inizio. Per il software preesistente vale la pena sostenere tale onere soltanto se un potenziale utente intende utilizzare il software. Indipendentemente da ciò, la lista di controllo Em002-2.1 Valutazione preliminare OSS dovrebbe essere compilata per documentare le decisioni determinanti.

Le applicazioni il cui sviluppo è stato avviato dalle autorità federali il 1° gennaio 2024 o dopo tale data, oppure che sono sviluppate su mandato della Confederazione sulla base di un contratto concluso dopo il 1° gennaio 2024, non sono considerate preesistenti e sottostanno in ogni caso all'obbligo di pubblicazione secondo l'art. 9 cpv. 1 LMeCA.

L'obbligo di pubblicazione secondo la LMeCA non dipende dall'utilità per i terzi.

3.1.4. Librerie, plug-in e add-on

In taluni casi il software non è un'applicazione autonoma, bensì soltanto una parte di essa. La LMeCA e l'ordinanza non definiscono ulteriormente il codice sorgente. Le istruzioni si applicano anche a librerie, plug-in e add-on disaccoppiati, che rientrano nella LMeCA.

Compiti:

- Raccogliere le informazioni contenute in Em002-5 Promemoria Strumenti OSS. Esso descrive sia informazioni generali sul software open source sia informazioni specifiche sull'applicazione della LMeCA.
- Verificare se le condizioni previste dalla LMeCA sono soddisfatte.
- Compilare la lista di controllo Em002-2.1 Valutazione preliminare OSS. Di regola è opportuno coinvolgere direttamente le persone di contatto del fornitore del software o del team di sviluppo.
- Se necessario, compilare la lista di controllo Em002-4.1 Community OSS.

Decisione: il software deve essere pubblicato e in che modo?

3.2. Eccezione: diritti di terzi

Occorre rinunciare a una pubblicazione se essa violerebbe diritti di terzi. Se il software è stato sviluppato da impiegati della Confederazione, i diritti spettano al datore di lavoro.[6] Nei contratti di fornitura di personale a prestito e di prestazioni informatiche i diritti sono di norma trasferiti alla Confederazione e dovrebbero essere fatti valere.

6. Art. 332 CO in combinato disposto con l'art. 6 cpv. 2 LPers

Se il software è stato o viene sviluppato sulla base delle condizioni generali della Confederazione Svizzera (stato 2024),[7] i diritti di proprietà intellettuale spettano al committente, salvo diversa pattuizione contrattuale.

Un'applicazione è tipicamente composta da numerosi singoli componenti ed elementi. Per alcuni tipi ciò diventa problematico se essi stessi non sono open source o non possono essere pubblicati sotto una licenza open source nell'ambito dell'applicazione.

Librerie

A seconda del linguaggio di programmazione, le librerie sono compilate direttamente con l'applicazione, collegate oppure fornite come pacchetti. Esse costituiscono parte integrante dell'applicazione.

Se l'applicazione contiene librerie proprietarie o soggette a licenza, l'open sourcing diventa più difficile. In linea di principio il codice sorgente del resto dell'applicazione può essere pubblicato, ma i potenziali utenti o sviluppatori dovrebbero acquistare la libreria in questione prima che l'applicazione diventi utilizzabile o estendibile.

Se la proprietà della libreria problematica spetta al fornitore dell'applicazione, vale la pena chiedere un chiarimento giuridico per stabilire se la Confederazione disponga di diritti trasferibili su tali librerie.

Ove possibile, le librerie proprietarie e soggette a licenza dovrebbero essere evitate negli sviluppi propri o sostituite se possibile prima della pubblicazione. Considerato l'obiettivo dell'articolo 9 LMeCA, occorre pubblicare il maggior numero possibile di funzionalità utilizzate dalle amministrazioni pubbliche. Singole librerie che impediscono la pubblicazione del codice rappresentano pertanto un debito tecnico che dovrebbe essere documentato e, all'occasione, eliminato.

Banche dati e altri sistemi di archiviazione

Se l'applicazione utilizza sistemi di archiviazione proprietari e soggetti a licenza quali Microsoft SQL Server, ciò non costituisce un ostacolo a una pubblicazione open source.

Occorre tuttavia esaminare se possano essere supportate anche banche dati aperte quali PostgreSQL. Ciò può ridurre i costi d'esercizio e abbassare le barriere d'accesso per i potenziali utenti.

Server applicativi e sistemi operativi

Se l'applicazione richiede sistemi operativi o server applicativi sotto licenza (p. es. Microsoft Windows Server o RedHat JBoss), ciò non costituisce un ostacolo a una pubblicazione open source.

Raccomandiamo di chiedere al fornitore del software se e quali librerie di terzi sotto licenza siano utilizzate come componenti dell'applicazione.

Se la realizzazione del software è stata acquistata congiuntamente da più organizzazioni o altre istituzioni, la proprietà spetta di regola a un'associazione o a un'altra persona giuridica costituita a tale scopo.

Altra proprietà intellettuale

Tra i diritti di terzi rientrano non soltanto i diritti d'autore, ma anche altra proprietà intellettuale (marchi, brevetti). I brevetti non impediscono fondamentalmente che un software sia pubblicato come open

7. Condizioni generali della Confederazione Svizzera, in particolare il numero 25.1 – https://www.bkb.admin.ch/bkb/de/home/themen/agb.html#accordion_21321111141713247349255

source, ma possono impedirne l’utilizzo o l’estensione. Sebbene i brevetti siano piuttosto rari in Svizzera, di regola vale la pena chiedere brevemente al fornitore del software se sia già stato effettuato un esame.

Acquisizione dei diritti necessari

Per soddisfare le prescrizioni legali della LMeCA, l’autorità federale ha, in qualità di committente, la possibilità di assicurarsi i diritti sui risultati dei lavori. Tale presupposto per una pubblicazione può essere definito come criterio in sede di acquisto. L’acquisizione successiva dei diritti necessari sul software può risultare più complessa.

Se l’autorità federale non intende pubblicare essa stessa il software, può per esempio delegare tale compito al mandatario o trasferire di conseguenza i diritti di pubblicazione. In tal caso essa trasferisce i diritti (e gli obblighi) necessari al fornitore e/o a terzi e li incarica della pubblicazione. Il diritto viene così di fatto trasferito a terzi a condizione della pubblicazione o del supporto del software corrispondente.

Anche in questo caso [KKB-MB] contiene indicazioni importanti.

Procedura

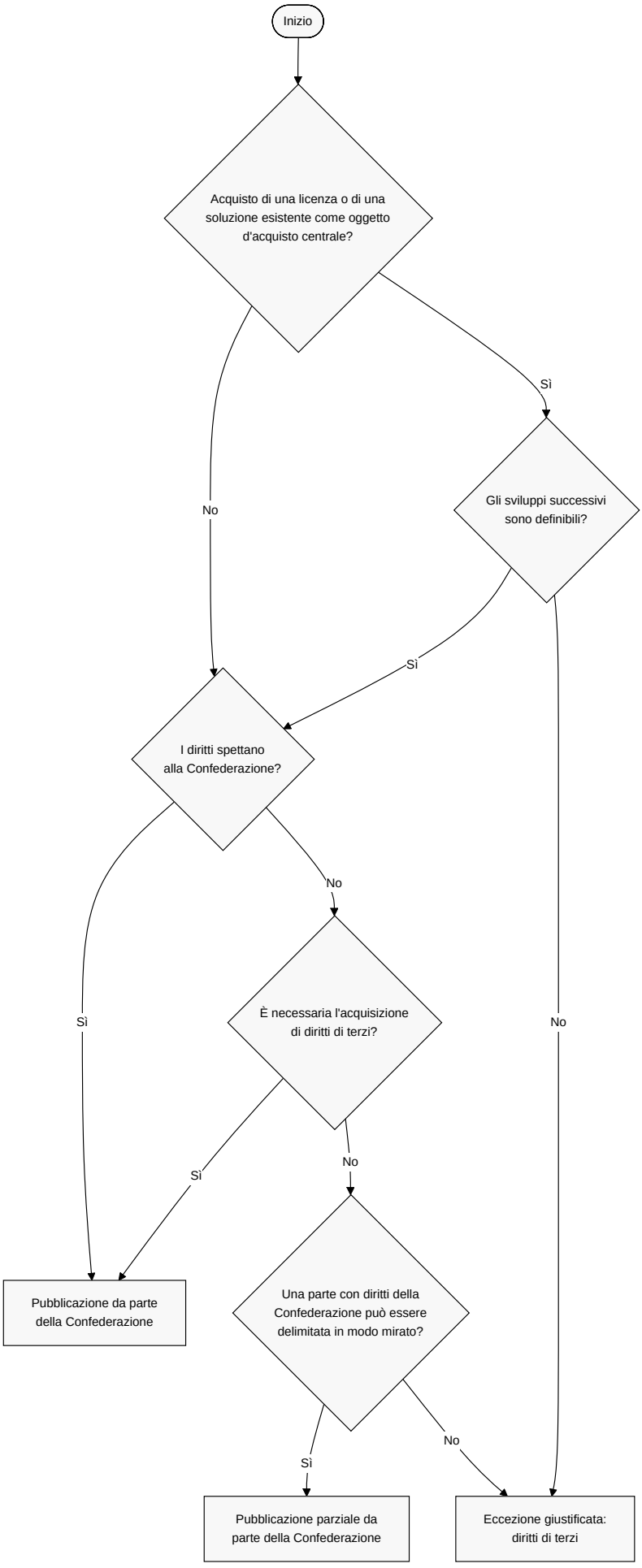
Inizio
Acquisto di una licenza o di una soluzione esistente come oggetto d'acquisto centrale?
Sì
No
Gli sviluppi successivi sono definibili?
Sì
No
I diritti spettano alla Confederazione?
Sì
No
È necessaria l'acquisizione di diritti di terzi?
Sì
No
Una parte con diritti della Confederazione può essere delimitata in modo mirato?
Sì
No
Pubblicazione da parte della Confederazione
Pubblicazione parziale da parte della Confederazione
Eccezione giustificata: diritti di terzi

Figura 2 – Eccezione: diritti di terzi (considerare sempre anche [KKB-MB] e Em002-7 Aspetti strategici degli acquisti e del software open source)

Compiti:

- Verificare se l'organizzazione detiene i diritti di protezione determinanti sul software.
- Verificare se i diritti possono essere acquisiti.
- Se necessario, ottenere l'autorizzazione scritta dei titolari dei diritti.
- Assicurarsi che l'applicazione non contenga parti proprietarie o protette.
- Verificare che non sussistano ostacoli dovuti alla protezione brevettuale (per quanto possibile e ragionevole).
- Verificare se la pubblicazione debba essere effettuata da terzi.
- Verificare se lo sviluppo debba basarsi sullo sviluppo di software open source (mediante la Em002-4 Guida alle community OSS).

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

3.3. Eccezione: motivi rilevanti per la sicurezza

Il software che non può essere pubblicato per motivi rilevanti per la sicurezza è esentato dalla pubblicazione.

I dati veri e propri di un software non sono interessati dalla pubblicazione del codice sorgente. Tali dati sono di regola sensibili e non vengono pubblicati. Può tuttavia accadere che, oltre ai dati di contenuto, determinati algoritmi e procedimenti visibili nel codice sorgente non debbano diventare pubblici (p. es. capacità cibernetiche offensive, dettagli del rilevamento delle frodi). La sicurezza può riferirsi alla riservatezza, all'integrità, alla disponibilità o alla tracciabilità.

Raccomandazione:

Il software dovrebbe essere sviluppato deliberatamente in modo che non sorga alcun rischio supplementare nemmeno in caso di pubblicazione del codice sorgente. L'ipotesi secondo cui la mancata pubblicazione protegge da attacchi riusciti è ingannevole («security through obscurity»).

Procedura:

> **NOTA**
>
> Determinante è sempre il software e non la sua finalità. Il trattamento di dati critici non costituisce a priori un aspetto rilevante per la sicurezza.

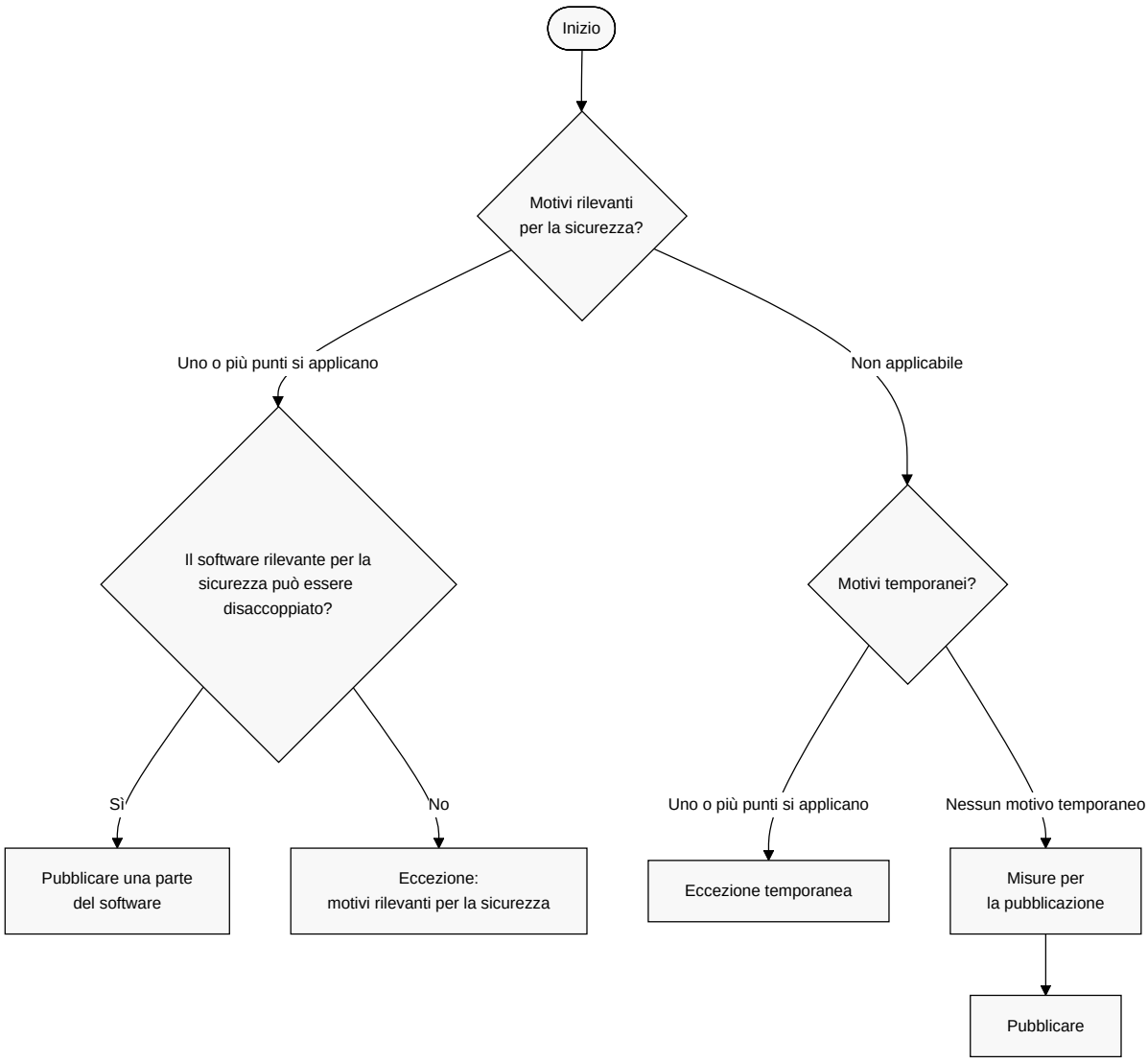

Figura 3 – Eccezione: motivi rilevanti per la sicurezza

I motivi seguenti possono essere considerati rilevanti per la sicurezza a seconda del progetto:

- La pubblicazione del software aumenterebbe sensibilmente un rischio per la sicurezza delle informazioni.[8]
- Per gli algoritmi di prevenzione delle frodi e analoghi si può se del caso rinunciare a una pubblicazione.
- Tecnologie di cibersicurezza particolarmente sensibili.
- Software essenziale per l’esercizio di infrastrutture critiche.
- Quando la pubblicazione violerebbe altre basi legali.

I seguenti motivi temporanei possono ancora impedire una pubblicazione. Quest’ultima sarà tuttavia resa possibile in seguito mediante misure adeguate.

- Il codice preesistente non ha ancora potuto essere ripulito e contiene eventualmente problemi rilevanti per la sicurezza.

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

- Gli audit di sicurezza del codice non sono ancora stati svolti.
- Il processo di sviluppo non è ancora sufficientemente maturo per consentire una pubblicazione sicura. Se il processo di sviluppo non è sufficientemente consolidato, è possibile che vengano pubblicate parti non destinate a terzi. Per esempio le direttive sui commenti al codice e la generazione dei dati di test devono essere adattate al nuovo scenario di pubblicazione e funzionare in modo affidabile.

Mediante le misure seguenti il software può essere pubblicato nonostante i motivi rilevanti per la sicurezza:

- Separare dal software le configurazioni (p. es. tipo di cifratura utilizzata).
- Esternalizzare in configurazioni o impostazioni le informazioni tecnicamente sensibili e non pubbliche (p. es. processi, calcoli).
- Pianificare, attuare ed esercitare regolarmente misure organizzative, personali e tecniche adeguate.
- Pianificare e svolgere attività di qualità adeguate al momento della pubblicazione. Cfr. il capitolo seguente «Analisi e preparazione».

Compiti:

- Esaminare il software quanto ai motivi rilevanti per la sicurezza che osterebbero a una pubblicazione.
- Stimare l’onere necessario per eliminare i motivi temporanei.
- Pianificare misure volte a separare le informazioni rilevanti per la sicurezza dal codice sorgente del software.
- Assicurarsi che l’applicazione non contenga procedimenti che non devono diventare pubblici.

Decisione:

Sussistono motivi imperativi rilevanti per la sicurezza che rendono impossibile la pubblicazione?

3.4. Chiarimento della pubblicazione

Questa fase chiarisce chi pubblicherà il software. Se il software è stato realizzato da un fornitore esterno, la pubblicazione può essere affidata anche a tale fornitore. In questa fase è importante stabilire dove sarà pubblicato il codice sorgente e definire una governance corrispondente.

Se il software è stato sviluppato internamente alla Confederazione e questa non può o non intende pubblicarlo essa stessa, può essere conferito un mandato di pubblicazione a un fornitore oppure i diritti possono essere ceduti (trasferimento degli obblighi a terzi).

Pubblicazione minima e supporto

La LMeCA non prevede requisiti in materia di pubblicazione. Una pubblicazione minima del codice sorgente soddisfa la prescrizione legale. Non sono richieste ulteriori attività. La questione se e in quale misura sia offerto un supporto e con quale intensità il progetto sia curato può essere affrontata mediante la Em002-4 Guida alle community OSS.

Sviluppo di software open source (OSSD)

Nello sviluppo di software open source (OSSD), oltre alla pubblicazione del codice sorgente, l'intero processo di sviluppo si svolge pubblicamente. Tutto, dai requisiti (issue) al codice sorgente, è trasparente. L'OSSD è supportato da strumenti di comunicazione pubblici (mailing list, forum ecc.), da un sistema di gestione delle versioni (git), da elenchi di errori e funzionalità, da una roadmap e da strumenti per sviluppatori. Grazie a questo metodo di lavoro trasparente ne beneficia l'intera community e diventa possibile la collaborazione oltre i confini organizzativi. Con questo approccio, al termine del progetto non è necessaria alcuna ulteriore attività di pubblicazione, poiché ciò è già stato considerato all'inizio del progetto.

Compiti

- Elaborare una stima approssimativa dell'onere per la fase di analisi. A seconda delle dimensioni e della complessità dell'applicazione, l'onere varia tra tre giorni e due settimane.
- Liberare risorse per le fasi successive, in particolare per l'analisi.
- Approvare in linea di principio la pubblicazione dell'applicazione come open source.

Decisione:

L'accordo di principio e la fattibilità per pubblicare l'applicazione come open source sono dati.

3.5. Ridurre al minimo l'onere

Le liste di controllo e le istruzioni sono strutturate in modo da ridurre al minimo l'onere legato alle pubblicazioni. Una pianificazione anticipata e coerente della pubblicazione all'inizio del progetto e uno sviluppo orientato alla pubblicazione riducono al minimo l'onere. Un certo onere amministrativo e determinati costi sono inevitabili [Le2023]. Essi possono eventualmente essere recuperati mediante community con costi di sviluppo condivisi (cfr. Em002-4 Guida alle community OSS).

3.6. Scelta della lingua di pubblicazione

La scelta della lingua deve tenere conto sia del potenziale pubblico target sia del metodo di lavoro del team. Fortunatamente le traduzioni sono diventate molto più semplici grazie ai nuovi strumenti. Entrano in considerazione le lingue ufficiali della Confederazione[9] e l'inglese.

Se il progetto ha rilevanza internazionale, l'inglese dovrebbe essere la lingua target. Per una pubblicazione successiva un cambio di lingua ha naturalmente poco senso.

Come minimo il file README.md dovrebbe essere disponibile in più lingue, affinché gli interessati possano vedere subito se il progetto è rilevante per loro.

La comunicazione pubblica dovrebbe considerare in primo luogo il pubblico target. Occorre inoltre tenere presente che si tratta di comunicazione pubblica della Confederazione.

Nota: con la pubblicazione l'Amministrazione federale accede a un nuovo canale di comunicazione pubblica. Se ciò non avviene con cura e professionalità, l'immagine pubblica della Confederazione può rapidamente risentirne.

4. Analisi e preparazione

Obiettivo: i lavori necessari a una pubblicazione open source sono conclusi. Le decisioni relative alla licenza e al tipo di community sono state prese. Ciò avviene mediante la lista di controllo Em002-2.2 Analisi e preparazione OSS.

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

4.1. Analisi del codice sorgente

Il codice sorgente da pubblicare e gli altri documenti correlati sono analizzati prima della pubblicazione. Tale analisi garantisce che non vengano pubblicate informazioni confidenziali.

Se è prevista la pubblicazione di un ampio volume di codice sorgente, si raccomanda di utilizzare strumenti automatizzati adeguati per le attività seguenti.

I dettagli sono definiti nella sezione E.

Compiti

- Verificare come sono state attuate la qualità del codice esistente e le direttive (p. es. ISO/IEC 25010:2023[10]).
- Assicurarsi che non siano contenuti dati (di test) sensibili, p. es. dati basati su persone o casi reali.
- Aggiornare la documentazione del software (cfr. allegato).
- Cancellare i file superflui o i documenti e dati obsoleti.
- Verificare il codice sorgente alla ricerca di segreti o credenziali.[11]
- Svolgere test di sicurezza mirati e istituire in seguito un programma di bug bounty (pubblico).[12]
- Verificare tutte le librerie utilizzate quanto alle licenze e allestire i relativi elenchi (attribuzioni). Se una libreria è disponibile sotto più licenze, ciò va indicato di conseguenza negli elenchi.
- Descrivere il deployment (pipeline CI/CD) ed eventualmente allestire un'istanza dimostrativa.

Queste attività sono indipendenti dalla pubblicazione, ma costituiscono presupposti per una buona qualità del software e per garantire la conformità. L'osservanza di tali misure semplifica la collaborazione, accresce la qualità dei contributi esterni e migliora l'immagine del software o dell'autorità.

4.2. Scelta della licenza

Cfr. Em002-3 Guida alle licenze OSS.

Andrebbero utilizzati testi di licenza affermati a livello internazionale, ove ciò sia possibile e opportuno. Le pretese di responsabilità dei licenziatari vanno escluse nella misura consentita dal diritto.

Compiti

- Verificare eventuali dipendenze da licenze esistenti.
- Per il codice di terzi, verificare la compatibilità delle condizioni d'uso con la licenza prevista e, se necessario, sostituire il codice.[13]
- Scegliere la licenza adeguata.

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
13. Cfr. Em002-3 riguardo alla scelta di codice o librerie di terzi per uno stack tecnologico e riguardo alla compatibilità della licenza.

La compatibilità delle licenze può essere verificata con strumenti quali ORT Toolkit,[14] Black Duck,[15] FOSSA[16] o FOSSology.[17] Il ToDo Group [18] offre una panoramica aggiornata degli strumenti.

Decisione:

- Stabilire e documentare sotto quale licenza il software sarà pubblicato.

4.3. Documentazione open source

Con la pubblicazione si attende un minimo di documentazione. Nell'ecosistema open source si sono affermati diversi documenti. Essi offrono ai visitatori una rapida panoramica del software, licenza compresa.

Sono attesi i documenti seguenti:

Documento	Descrizione / contenuto:
README(.md)	Funge da documento introduttivo e offre una rapida panoramica di scopo, portata, stato, licenza e gruppi target del progetto.
LICENSE	Descrizione della licenza scelta nel progetto
CONTRIBUTING(.md)	Descrive il processo per contribuire al progetto.
CODE_OF_CONDUCT(.md)	Il codice di condotta stabilisce le norme sociali attese all'interno del progetto.
CHANGELOG(.md)	Tiene un elenco delle modifiche nelle versioni del software.
THIRD-PARTY-LICENSES(.md)	Descrizione delle licenze di terzi o dei componenti utilizzati.
Getting started	Breve guida all'installazione e all'utilizzo del software.

14. https://oss-review-toolkit.org/
15. https://www.blackducksoftware.com
16. https://www.fossa.com
17. https://www.fossology.org
18. https://landscape.todogroup.org/

Documento	Descrizione / contenuto:
Create Gitignore	Descrive i documenti che non saranno pubblicati, p. es. configurazioni, dati di test o contenuti generati. Le informazioni protette e sensibili dovrebbero essere gestite al di fuori del progetto, in strumenti e archivi appositi e adeguati.
publiccode.yml	Metadati strutturati per la descrizione del software, sulla base dello standard https://yml.publiccode.tools/. Il formato definito rende la pubblicazione più facilmente reperibile e riutilizzabile. Ulteriori informazioni nell'allegato F.

Tabella 1 – Documenti open source

Ulteriori dettagli tecnici su questi documenti sono descritti nell'allegato.

Compiti

- Allestire la documentazione standard.
- Riportare copyright, indicazioni di licenza e clausola di esclusione della responsabilità in tutti i file.
- A seconda della piattaforma di pubblicazione possono essere aggiunte descrizioni supplementari.

Decisione:

- Decisione definitiva di pubblicare l'applicazione come open source.

5. Pubblicazione e annuncio

Obiettivo: l'applicazione è pubblicata come open source e facilmente reperibile da terzi. Tutti i componenti sono pronti per la pubblicazione e la community è costituita o consolidata. Ciò avviene mediante la lista di controllo Em002-2.3 Rilascio e pubblicazione OSS. In una certa misura si tratta di ricapitolare decisioni già prese.

5.1. Scelta della piattaforma

Esistono diversi modi per pubblicare codice sorgente. La tabella seguente offre una panoramica dei possibili sistemi di gestione del codice sorgente (SCM) e delle piattaforme.

La piattaforma adeguata va scelta prima della pubblicazione vera e propria. Alcune autorità federali sono già presenti su GitHub.[19] In tal caso è opportuno utilizzare piattaforme già consolidate.

Raccomandazione:

Attualmente raccomandiamo di istituire un'organizzazione su GitHub per ogni autorità federale ai fini della pubblicazione del codice sorgente. All'interno di tale organizzazione possono essere creati i

19. Il sito https://ossbenchmark.com offre una panoramica delle organizzazioni e autorità attive su GitHub.

progetti (repository) corrispondenti e gestiti gli utenti con le relative autorizzazioni. In futuro dovrebbe essere istituita una piattaforma federale centrale. Per garantire una gestione ottimale degli utenti e preservare la reputazione della Confederazione Svizzera, è importante utilizzare un numero minore di repository, ma meglio gestiti.

Piattaforma	Caratteristiche
GitHub[20]	Piattaforma online di Microsoft, ampiamente diffusa. Una grande quantità di software open source è ospitata su GitHub. Offre molti strumenti supplementari oltre all'SCM. Disponibile in versione Free ed Enterprise (abbonamento).
GitLab[21]	Piattaforma online dell'omonima impresa GitLab, essa stessa pubblicata sotto una licenza open source. Offre molti strumenti supplementari oltre all'SCM. Può anche essere gestita localmente, offrendo così un certo grado di sovranità. Disponibile in versione Free ed Enterprise (abbonamento).
Bitbucket[22]	Piattaforma online di Atlassian, specificamente integrata nel suo ecosistema. Disponibile in versione Free ed Enterprise (abbonamento).
Sistema SCM interno	Se è messa a disposizione una piattaforma SCM propria, anch'essa può pubblicare e rendere disponibili determinati repository o progetti. L'esercizio deve essere assicurato.
Piattaforma federale	Attualmente non esiste una piattaforma federale centrale per la pubblicazione di software. Una misura in tal senso è proposta in Em002 Linee guida strategiche per il software open source nell'Amministrazione federale.

20. https://github.com/about/
21. https://about.gitlab.com/
22. https://bitbucket.org/product/en

Piattaforma	Caratteristiche
Sito web, server FTP pubblico ecc.	La LMeCA non definisce come il software debba essere pubblicato. Una pubblicazione minima[23] può avvenire anche su un sito web o altre risorse accessibili al pubblico. Ciò non è tuttavia adatto a una collaborazione duratura nel senso dell'open source.

Tabella 2 – Piattaforme per la pubblicazione di software open source

Se il software è pubblicato congiuntamente a un partner o a un'organizzazione esterna, occorre assicurarsi che siano disponibili le autorizzazioni e gli accessi adeguati al codice sorgente.

Compiti

- Valutare la piattaforma adeguata, compresa l'organizzazione o il progetto corrispondente.
- Se non ancora chiarito, stabilire la denominazione del progetto o del repository.
- Definire e verificare le autorizzazioni sul repository.
- Assicurare una governance adeguata e un'eventuale replica del codice sorgente.

Decisione:

- La piattaforma per la pubblicazione è stata scelta in modo appropriato.

5.2. Pubblicazione del software

Una volta chiariti i presupposti, il software può essere pubblicato.

Compiti

- Verificare i presupposti mediante la lista di controllo Em002-2.3 Rilascio e pubblicazione OSS.
- L'accesso alla piattaforma è disponibile e la governance è chiarita.
- Pubblicazione del codice sorgente e della documentazione da parte del team di sviluppo.

5.3. Comunicazione

Dopo la pubblicazione vera e propria, ulteriori attività di comunicazione possono essere avviate dall'autorità federale interessata e al suo interno. Esse favoriscono la diffusione e apportano il valore aggiunto desiderato dalla pubblicazione.

Compiti

- Descrizione nell'hosting del repository, impostazioni di sicurezza ecc.
- Comunicazione interna ed eventualmente esterna
- Redigere un messaggio destinato alla community.

23. Una pubblicazione minima comprende soltanto la pubblicazione del codice sorgente e di altri artefatti minimi. Questo tipo di pubblicazione soddisfa le prescrizioni della LMeCA. Con essa non si consegue tuttavia alcun valore aggiunto.

- Far conoscere l’applicazione all’interno dell’organizzazione e negli organi specializzati, promuoverne l’utilizzo e i contributi.
- Creare un file publiccode.yml[24] nel repository per il catalogo OSS dell’Amministrazione federale.[25]

5.4. Costituzione e cura della community

Pubblicando un software può nascere una community in grado di contribuire allo stesso mediante contributi (pull request), domande, proposte di documentazione ecc. Per disporre di una community attiva è necessario un onere considerevole sia per la costituzione sia per la cura continua. Una community ben funzionante riduce al minimo i fork indesiderati del software. L’obiettivo è qui attivare il valore aggiunto della community mediante attività mirate. Anche in caso di pubblicazione minima andrebbe definito come trattare riscontri e domande sul software.

Compiti

- Chiarire chi siano i potenziali utenti o gli interessati.
- Determinare la forma adeguata per l’applicazione concreta mediante la Em002-4 Guida alle community OSS.
- Costituire la community.
- Attività di costituzione della community.

6. Allegato

6.1. Modifiche rispetto alla versione precedente

- Capitoli 4.3 e 5.3: pubblicazione completata con publicode.yml
- È stato aggiunto l’allegato F «Public Code»
- Diverse modifiche e miglioramenti redazionali minori

6.2. Riferimenti

Cfr. *Em002 Linee guida strategiche per il software open source nell’Amministrazione federale*.

6.3. Abbreviazioni

Cfr. Em002 Linee guida strategiche per il software open source nell’Amministrazione federale ed Em002-6 FAQ sull’OSS.

6.4. Documentazione di accompagnamento **Em002-2.2 Lista di controllo Analisi e preparazione OSS – Documentazione**

6.5. Documentazione del codice sorgente

La documentazione dovrebbe far parte del codice sorgente del progetto, affinché le modifiche alla documentazione siano archiviate in modo inalterabile e la documentazione sia versionata parallelamente al progetto. Ciò garantisce automaticamente che la documentazione, se correttamente

24. Lo standard di metadati https://yml.publiccode.tools/ descrive il software. Il formato definito rende la pubblicazione più facilmente reperibile e riutilizzabile. Ulteriori informazioni nell’allegato F.
25. Catalogo OSS: https://www.opensource.admin.ch

curata, non diverga dallo stato di sviluppo del progetto e che la documentazione relativa a versioni precedenti del progetto possa essere consultata in qualsiasi momento.

Inoltre ciò offre a terzi la possibilità di contribuire facilmente a modifiche della documentazione.

Come linguaggio di marcatura per la formattazione del testo andrebbe utilizzato Markdown, in quanto facile da scrivere, ampiamente diffuso e leggibile dalle macchine. I documenti Markdown sono inoltre visualizzati in un formato target ben leggibile da tutte le piattaforme correnti. I documenti Markdown possono essere resi in qualsiasi formato target indipendentemente dalla piattaforma utilizzata, mediante generatori di siti statici quali Jekyll, MkDocs, Gatsby o Sphinx, per esempio con le prescrizioni di identità visiva di un'amministrazione.

6.6. Destinatari e struttura della documentazione

La documentazione di progetto dovrebbe rivolgersi sia agli utenti finali sia agli specialisti tecnici e non concentrarsi quindi esclusivamente sulla tecnica. Per la scelta della lingua si veda il capitolo 3.6. Ciò favorisce la diffusione del software.

Un file README.md funge da documento introduttivo. I file con questo nome sono riconosciuti da tutte le piattaforme correnti e visualizzati come punto d'ingresso alla documentazione.[26]

README.md dovrebbe offrire una rapida panoramica di scopo, portata, stato, licenza e gruppi target del progetto. Concretamente README.md dovrebbe contenere i punti seguenti:

- Nome del software
- Breve descrizione dello scopo e della funzione del software. Portata e possibili limiti
- Istruzioni d'installazione
- Utilizzo del software
- Rinvio a un'eventuale istanza dimostrativa
- Indicazioni di supporto e contatto
- Come contribuire al software open source (Contributing)
- Licenza utilizzata (Licence)
- Stato del progetto / stato di sviluppo

6.7. Evidenziare l'open source e archiviare la licenza

Subito dopo la dichiarazione d'intenti andrebbe indicato in modo chiaro e inequivocabile che il progetto è software open source. La licenza utilizzata dal progetto andrebbe indicata con il corrispondente identificatore SPDX[27] e collegata al testo concreto della licenza contenuto nel file LICENSE. I modelli di licenza contengono tipicamente una parte variabile (p. es. nome del progetto, anno di copyright, nome dell'autore) da adattare nel file LICENSE.

La maggior parte delle piattaforme riconosce le licenze contenute in un file LICENSE e offre informazioni chiare sulla licenza, p. es. una breve sintesi delle condizioni di licenza.

Si raccomanda di aggiungere a ogni file sorgente creato un'intestazione minima di licenza e copyright. Le intestazioni esistenti nei file esistenti non andrebbero modificate. Se si contribuisce a un progetto con regole chiare sulle intestazioni, queste vanno rispettate.

---

26. Maggiori informazioni sulla creazione di un file readme: https://www.makeareadme.com/
27. Identificatore SPDX – https://spdx.org/about

Per maggiori dettagli si veda: Producing Open Source Software: State That the Project is Free.[28]

6.8. Stato di sviluppo attuale

Per mostrare rapidamente lo stato del progetto, lo stato di sviluppo attuale andrebbe documentato in README.md. Per i lettori è importante sapere se il progetto si trova in uno stato maturo o all'inizio dello sviluppo, se è attivamente curato e con quale frequenza vengono pubblicate nuove versioni.

Questa sezione può descrivere quale tipo di sostegno da parte di terzi sia attualmente più importante per il progetto. Potrebbe trattarsi per esempio di uno sviluppatore con conoscenze tecnologiche specifiche o di una persona incaricata di rielaborare la documentazione del progetto.

Per maggiori dettagli si veda Producing Open Source Software: Development Status.[29]

6.9. Istanza dimostrativa

Per le applicazioni si raccomanda di mettere a disposizione un'istanza dimostrativa, affinché l'applicazione possa essere provata senza oneri d'installazione. A seconda del tipo di applicazione, l'istanza dimostrativa può essere l'installazione produttiva o un ambiente di test. È importante che i lettori possano accedere all'istanza corrispondente nel modo più autonomo e diretto possibile.

In alternativa molti progetti open source mettono a disposizione strumenti per avviare facilmente l'applicazione. Ciò può avvenire per esempio mediante container, applicazioni portatili o script d'installazione.

6.10. Persona di contatto specializzata e titolarità del progetto

Il file README.md dovrebbe indicare la principale persona di contatto specializzata, affinché altre amministrazioni e organizzazioni interessate possano mettersi in contatto nel modo più diretto possibile.

Gli indirizzi e-mail personali non andrebbero utilizzati.

Il file README.md dovrebbe inoltre descrivere brevemente quale organizzazione sia titolare del progetto, in particolare se per esso è stata costituita un'associazione.

7. Documentazione d'installazione

Per le applicazioni, README.md dovrebbe rinviare a una documentazione d'installazione che descriva i requisiti hardware e software, quali componenti infrastrutturali (per esempio banche dati) siano utilizzati e come l'applicazione possa essere installata ed esercita.

7.1. Developer Guidelines

Il punto d'ingresso alle Developer Guidelines andrebbe archiviato in un file Markdown CONTRIBUTING.md, collegato in README.md. Le Developer Guidelines fanno parte della documentazione estesa destinata agli sviluppatori che desiderano contribuire al progetto e si concentrano in primo luogo sulla collaborazione e la comunicazione nel progetto piuttosto che sulla tecnica.

Le Developer Guidelines dovrebbero contenere i punti seguenti e rinviare se del caso ai documenti corrispondenti:

28. https://producingoss.com/en/getting-started.html#state-freedom
29. https://producingoss.com/en/getting-started.html#state-freedom

- Un rinvio al codice di condotta del progetto. Il codice di condotta stabilisce le norme sociali attese all'interno del progetto.
- Si raccomanda di scegliere come codice di condotta il Contributor Covenant Code of Conduct, pubblicato sotto licenza CC-BY-4.0,[30] oppure di basarvisi.[31]
- Un'indicazione secondo cui il progetto effettua revisioni del codice sotto forma di pull request GitHub, con rinvio alla documentazione GitHub sulle pull request.

Infine occorre rinviare alla Developer Documentation.

7.2. Developer Documentation

Mentre le Developer Guidelines descrivono la collaborazione e le norme sociali del progetto, la Developer Documentation si concentra sugli aspetti tecnici della partecipazione allo sviluppo del progetto. Essa dovrebbe descrivere almeno i punti seguenti:

- Quali dipendenze tecniche presenta il progetto e quali strumenti sono necessari per il suo sviluppo.
- Quali regole formali si applicano al codice sorgente del progetto, per esempio in materia di formattazione.
- Come è organizzato il codice sorgente del progetto e come le funzioni possono essere testate tecnicamente.
- Uno schizzo dell'architettura con una descrizione sommaria della stessa, preferibilmente con delimitazione del contesto e vista sommaria dei componenti.[32]
- Per le applicazioni: come l'applicazione può essere avviata a scopo di sviluppo o debug e quali componenti infrastrutturali sono necessari a tal fine.

8. Documentazione di accompagnamento della lista di controllo Em002-2.2 Analisi e preparazione OSS – Codice sorgente

8.1. Cronologia del codice sorgente

Il codice sorgente di un progetto è di norma archiviato in un sistema di gestione del codice sorgente (SCM) quale GitHub. Tali sistemi memorizzano non soltanto lo stato attuale del codice sorgente, ma anche le modifiche apportate nel tempo. Ciò comprende in particolare le cancellazioni nel codice sorgente, inclusi i file cancellati.

L'SCM contiene inoltre metainformazioni sulle modifiche del codice sorgente, quali il nome e l'indirizzo e-mail dell'autore o le firme PGP per commit e tag firmati.

8.2. Dati sensibili, protetti o confidenziali

Il codice sorgente e la sua cronologia possono contenere dati accessibili al pubblico, quali un elenco dei codici postali, dati generati casualmente o dati sintetici.

Il codice sorgente e la sua cronologia non devono contenere dati o informazioni (personali) sensibili o confidenziali ai sensi della legge federale sulla protezione dei dati (LPD) e della legge sulla sicurezza delle informazioni (LSIn). Ciò comprende in particolare:

30. https://creativecommons.org/licenses/by/4.0/
31. https://www.contributor-covenant.org/version/1/4/code-of-conduct.html
32. Quadro architetturale MMB (metodo di modellizzazione della Confederazione) o arc42. http://arc42.org/

- Nomi utente e password o altri segreti quali chiavi private, token di accesso o certificati.
- Nomi DNS interni, URL, nomi host, indirizzi IP, strutture di rete o nomi di unità.
- Nomi, indirizzi, immagini o dati personali analoghi senza il consenso della persona interessata
- Dati anonimizzati o pseudonimizzati (poiché è possibile annullare l'anonimizzazione o la pseudonimizzazione)

Anche se lo stato attuale del codice sorgente è privo di dati o informazioni sensibili, protetti o confidenziali, tali dati o informazioni possono comunque essere recuperati dalla cronologia del sistema SCM se in passato hanno mai fatto parte del codice sorgente o delle metainformazioni dell'SCM. La pulizia dello stato attuale del codice sorgente non rimuove completamente tali informazioni.[33]

In caso di dubbio si raccomanda pertanto di rimuovere completamente la cronologia dopo la pulizia del codice sorgente, prima di pubblicare quest'ultimo.

I nomi e gli indirizzi e-mail degli autori del codice sorgente sono di norma disponibili direttamente nel codice sorgente o nei metadati dell'SCM, per esempio nei commit GitHub o nei tag GitHub annotati. Dal profilo giuridico ciò non è problematico, poiché spetta all'impresa di sviluppo disciplinare la pubblicazione di tali informazioni. Si raccomanda tuttavia di sensibilizzare l'impresa di sviluppo o i collaboratori interni della Confederazione sul fatto che i nomi e gli indirizzi e-mail degli autori del codice sorgente possono essere divulgati al momento della pubblicazione del software.

8.3. Rimozione delle indicazioni sugli autori

La cronologia del software può essere modificata per nascondere le indicazioni sugli autori. Alcuni strumenti[34] possono automatizzare ampiamente questa fase.

La rimozione delle indicazioni sugli autori elimina l'attribuzione, la tracciabilità e le informazioni di contatto. L'onere che ne deriva non va sottovalutato, a seconda del volume di codice sorgente. Per questi motivi la rimozione degli autori non è raccomandata. I nomi possono essere pubblicati se a) è stato ottenuto il consenso delle persone interessate e b) non vi si oppongono motivi rilevanti per la sicurezza.

8.4. Considerazione delle licenze delle librerie

Praticamente ogni progetto utilizza librerie di terzi (librerie software, dipendenze) per accedere a funzioni di uso frequente senza doverle programmare. Tali librerie dipendono di norma da altre librerie, con conseguenti dipendenze a più livelli (dipendenze transitive). A seconda del linguaggio di programmazione, anche piccoli progetti possono presentare centinaia di dipendenze.

I risultati di tale accertamento vanno considerati nella scelta della licenza conformemente alla Em002-3 Guida alle licenze OSS e nella lista di controllo Em002-2.3 Rilascio e pubblicazione OSS.

Per ciascuna di tali dipendenze occorre assicurarsi che la sua licenza sia compatibile con quella del progetto. Sono particolarmente problematiche le dipendenze che non sono soggette a una licenza open source, che non hanno dichiarato alcuna licenza o che utilizzano una licenza virale non corrispondente a quella del progetto. Alcune librerie sono licenziate sotto più di una licenza e consentono all'utente di sceglierne una.

- In tutti i principali linguaggi di programmazione le dipendenze del progetto (comprese quelle transitive) e le relative licenze possono essere elencate automaticamente. Tra questi figurano:

33. Una possibilità per ripulire un repository è offerta dallo strumento BFG Repo-Cleaner
34. https://www.adamdehaven.com/blog/update-commit-history-author-information-for-git-repository/

- Il plugin Project Info Reports per il gestore di pacchetti Apache Maven per progetti Java, che mediante il rapporto sulle dipendenze genera un output HTML di tutte le librerie utilizzate, licenza dichiarata compresa.[35]
- Il NPM Licence Checker per il gestore di pacchetti NPM per progetti JavaScript, che può restituire tutte le librerie utilizzate, licenza dichiarata compresa, in vari formati, p. es. CSV.[36]
- Il Licence Finder di Pivotal, che supporta diversi gestori di pacchetti e linguaggi di programmazione, tra cui Ruby Gems, Python Eggs e Godeps.[37]

9. Utilizzo di librerie con licenze proprietarie

Le librerie soggette a una licenza proprietaria sono in linea di principio problematiche nei progetti open source e di regola non possono essere utilizzate, in particolare se il codice sorgente è stato pubblicato sotto una licenza virale quale l'AGPL.

Vi preghiamo di rivolgervi al fornitore e al servizio giuridico dell'ufficio interessato per verificare se e a quali condizioni la libreria proprietaria possa essere utilizzata nel vostro progetto.

9.1. Utilizzo di gestori di pacchetti

Praticamente tutti i linguaggi di programmazione diffusi mettono a disposizione gestori di pacchetti per amministrare le proprie dipendenze, quali Apache Maven per i progetti Java, NPM per i progetti JavaScript o NuGet per i progetti .NET.

Utilizzando un gestore di pacchetti, le dipendenze impiegate dal progetto non fanno più direttamente parte del suo codice sorgente, per esempio mediante la copia del codice sorgente della dipendenza nel progetto o la copia delle dipendenze in forma binaria. Il progetto ottiene invece tutte le sue dipendenze tramite le dichiarazioni del gestore di pacchetti.

Ciò consente di elencare e referenziare correttamente tutte le dipendenze. Impedisce inoltre la modifica involontaria del codice delle dipendenze. Se sono necessarie modifiche, esse vanno effettuate direttamente alla fonte della dipendenza.

9.2. Dichiarazione della licenza del progetto nel descrittore del gestore di pacchetti

Il progetto dovrebbe dichiarare la propria licenza nel descrittore appropriato del gestore di pacchetti, utilizzando l'identificatore SPDX,[38] affinché la licenza del progetto possa essere restituita dal gestore di pacchetti. Ciò è particolarmente importante per le librerie, affinché i loro utenti possano interrogarne automaticamente la licenza tramite il gestore di pacchetti.

9.3. Attribuzione e indicazioni di copyright

Molte librerie sono soggette a una licenza che richiede al loro utente un'indicazione di copyright originale o un'attribuzione.

Tra queste figurano le licenze seguenti (estratto):

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

35. Info sul plugin https://maven.apache.org/plugins/maven-project-info-reports-plugin/plugin-info.html
36. NPM Licence Checker https://github.com/davglass/license-checker
37. Licence Finder di Pivotal https://github.com/pivotal/LicenseFinder
38. Identificatore SPDX – https://spdx.org/

Per le altre si veda l'elenco delle licenze riconosciute dall'OSI.[39]

Nella pratica ciò riguarda quasi tutte le librerie utilizzate. Poiché il numero di librerie impiegate nei piccoli progetti è già molto elevato, un'indicazione di copyright giuridicamente corretta o l'attribuzione necessaria è possibile soltanto con un onere considerevole.

Si raccomanda pertanto di procedere come segue:

- Allestire un elenco delle librerie utilizzate, mediante i meccanismi del gestore di pacchetti o strumenti quali il NPM Licence Checker o il Licence Finder di Pivotal.
- Preparare manualmente l'elenco affinché contenga le informazioni seguenti: nome ufficiale del progetto della libreria, sito web del progetto della libreria, licenza utilizzata per la libreria sotto forma di identificatore SPDX con collegamento al testo della licenza.
- Collocare l'elenco nel codice sorgente del progetto, p. es. in un file THIRD-PARTY-LICENSES.md.
- Per le applicazioni: inserire un collegamento a THIRD-PARTY-LICENSES.md in una finestra di dialogo «Informazioni su» o «Informazioni su questa applicazione».
- La finestra di dialogo «Informazioni su» è ampiamente utilizzata a tale scopo, per esempio in Google Chrome (nella finestra «Informazioni su Chrome»).
- Per le librerie: rinviare in README.md a THIRD-PARTY-LICENSES.md.
- Nell'ambito del processo di code review, assicurarsi che le modifiche alle librerie siano riportate in THIRD-PARTY-LICENSES.md.

10. Esempio di blocco di codice THIRD-PARTY-LICENSES.md

Questo progetto utilizza software open source:

- Spring Boot, projects.spring.io/spring-boot sotto licenza Apache-2.0
- caniuse-db, GitHub sotto licenza CC-BY-4.0

10.1. Documentazione di accompagnamento della lista di controllo Em002-2.3 Rilascio e pubblicazione OSS

10.1.1. Pubblicazione con publiccode.yml

Publiccode.yml è uno standard di metadati per repository di software contenenti software sviluppato o acquistato da amministrazioni pubbliche. L'obiettivo è rendere tale software più facilmente reperibile e riutilizzabile. Lo standard di metadati è affermato a livello internazionale ed è descritto pubblicamente all'indirizzo https://yml.publiccode.tools/. Un file publiccode.yml contiene tipicamente informazioni quali titolo e descrizione (possibile in più lingue), stato di sviluppo, indicazioni di contatto, informazioni sulla licenza ecc. Per la Svizzera sono previste estensioni specifiche per il Paese.

Per ogni software va creato un file yml nella directory principale del repository. Se la soluzione si compone di più repository (ausiliari), è sufficiente un unico file, poiché esso descrive il caso d'applicazione della soluzione.

Sulla base dei dati strutturati contenuti nei file publiccode.yml viene generato, per l'intera Amministrazione federale, un catalogo OSS (opensource.admin.ch). Il catalogo offre una panoramica

39. Le licenze open source sono licenze conformi all'Open Source Definition — in breve, consentono di utilizzare, modificare e condividere liberamente il software. https://opensource.org/licenses

del software pubblicato e agevola il reperimento e il riutilizzo di soluzioni esistenti.

Se viene creato un nuovo repository o una nuova organizzazione, ciò va segnalato per l'inserimento nel catalogo OSS all'indirizzo e-mail opensource@bk.admin.ch.

Oltre al catalogo OSS è disponibile anche un editor grafico e validatore per i file publiccode.yml.

Catalogo OSS ed editor: https://www.opensource.admin.ch/

10.2. Ulteriori fonti e licenza

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

- Google Open Source Docs di Google LLC, sotto licenza CC-BY-4.0[40]
- Producing Open Source Software, How to Run a Successful Free Software Project di http://www.red-bean.com/kfogel C-BY-SA-4.0[41]
- GitHub Healthy Contributions[42]
- Standard for Public Code[43]

40. Google Open Source Docs – https://opensource.google.com/docs/
41. Producing Open Source Software, How to run a Successful Free Software Project - https://producingoss.com/
42. GitHub Healthy Contributions
43. Standard for Public Code – https://standard.publiccode.net/