



# Ciclo di vita del software

---

**Questa voce o sezione sull'argomento ingegneria del software non cita le fonti necessarie o quelle presenti sono insufficienti.**

Il **ciclo di vita del software**, in informatica, e in particolare nell'ingegneria del software, si riferisce al modo in cui una metodologia di sviluppo scompone l'attività di realizzazione di prodotti software in sottoattività fra loro coordinate, il cui risultato finale è la realizzazione del prodotto stesso e tutta la documentazione a esso associata: fasi tipiche includono lo studio o analisi, la progettazione, la realizzazione, il collaudo, la messa a punto, l'installazione, la manutenzione e l'estensione,<sup>[1]</sup> il tutto a opera di uno o più sviluppatori software.

Lo sviluppo software è una procedura assai complessa e vasta in tutte le sue fasi, tali da richiedere a volte molti sviluppatori software per il completamento in tempi ragionevoli o prefissati; talvolta un progetto software è portato avanti da una comunità di utenti distribuita in una rete attiva attraverso un gruppo di discussione, come accade ad esempio per diverse distribuzioni Linux; quando poi da un progetto software se ne sviluppa un altro a partire dallo stesso tipo di programma, ma portato avanti da un'altra community, si parla di fork.

## Storia

---

La comparsa in letteratura e nella pratica dello sviluppo del software del concetto di "ciclo di vita" si può far coincidere con la nascita dell'ingegneria del software, in quanto rappresenta un passaggio storico dallo sviluppo del software inteso come attività "artigianale" (ovvero affidata alla libera creatività dei singoli individui) a un approccio più industriale, in cui la creazione di programmi e sistemi software viene considerata come un processo complesso che richiede pianificazione, controllo, e documentazione appropriati (così come avviene tradizionalmente nei settori più maturi dell'ingegneria).<sup>[2]</sup>

Questa transizione si può ricondurre, in ultima analisi, all'aumentata complessità dei sistemi, all'avvento di un vero e proprio mercato del software, nonché a nuovi e più stringenti requisiti di qualità, legati per esempio all'uso di sistemi informatici in contesti critici (centrali energetiche, sistemi aerospaziali, armamenti e così via). Questo mutamento di prospettiva iniziò a verificarsi, storicamente, fra la fine degli anni sessanta e l'inizio del decennio successivo. Da allora, la ricerca su questi temi è stata estremamente prolifica in ambito sia industriale sia accademico. Spesso la complessità di un progetto software richiede la supervisione o gestione da parte di uno o più project manager da parte del back office aziendale.

## Descrizione

---

Quasi tutti i modelli di ciclo di vita del software prevedono una scomposizione del processo di sviluppo in insiemi di attività simili (quando non addirittura identici). Le distinzioni fra diversi cicli di vita si evidenziano su altri aspetti, quali:

- l'enfasi relativa che si attribuisce a ciascuna attività;
- l'individuazione degli attori specifici incaricati di ciascuna attività;
- l'ordine in cui le attività si svolgono.

In tutti i cicli di vita del software svolge inoltre un ruolo essenziale la documentazione dei prodotti delle varie sottoattività; la stesura della documentazione viene quindi regolamentata nello stesso modo delle attività menzionate. Le fasi di progettazione del software sono:

- Analisi
  - Esplicitazione dei requisiti
  - Analisi dei requisiti
  - Analisi del dominio
  - Analisi di fattibilità
  - Analisi dei costi
- Progettazione
  - Progetto architetturale
    - Localizzazione software
  - Progetto di dettaglio
- Programmazione
- Ispezione e Debugging
- Collaudo
- Deployment
- Manutenzione

## Analisi

L'analisi è l'indagine preliminare del contesto in cui il prodotto software deve inserirsi, sulle caratteristiche o requisiti che deve esibire ed eventualmente su costi e aspetti logistici della sua realizzazione; questa fase può essere scomposta in sottoattività quali analisi di fattibilità, analisi e modellazione del dominio applicativo, analisi dei requisiti e così via. In senso più ampio si può dire che l'analisi ha lo scopo di definire (il più precisamente possibile) il problema da risolvere. Questa fase è costituita anche da raccolta dei dati tramite colloqui tra cliente/committente e relativi sviluppatori. Al termine della fase verrà creato un documento che descrive le caratteristiche del sistema, tale documento viene definito *documento di specifiche funzionali*.

## Progettazione




Lo stesso argomento in dettaglio: **Design pattern**.

Nell'attività di progettazione si definiscono le linee essenziali della struttura del prodotto software in funzione dei requisiti evidenziati dall'analisi, tipicamente appannaggio di un *analista programmatore*. Anche la progettazione può essere scomposta in sottoattività, dal progetto architetturale al progetto dettagliato. Si può dire che la progettazione ha lo scopo di definire (a un certo livello di dettaglio) la soluzione del problema. In questa fase sarà sviluppato un documento che permetterà di avere una


definizione della struttura di massima (architettura di alto livello) e una definizione delle caratteristiche dei singoli componenti (moduli).

## Implementazione

 Lo stesso argomento in dettaglio: **Framework** e **Codice sorgente**.

L'implementazione, detta anche *sviluppo* o *codifica* del prodotto software, è la fase di realizzazione del software, che concretizza la soluzione software attraverso la programmazione, ovvero la stesura di programmi da parte di programmatore o sviluppatore.

## Organizzazione

 Lo stesso argomento in dettaglio: **Modulo (programmazione)**.

Nella maggior parte dei casi, i programmatore che curano l'implementazione distinguono almeno due attività afferenti all'implementazione: l'implementazione dei singoli moduli, che costituiscono il sistema, e l'integrazione di tali moduli a formare il sistema complessivo. Per esempio, un'applicazione desktop potrebbe essere ripartita in tre componenti fondamentali: l'interfaccia utente (GUI), la logica di elaborazione e la gestione dei dati. Tipicamente lo sviluppo del codice di un'applicazione avviene in locale sul PC dello sviluppatore che potrà operare una prima fase di test verificando la bontà o meno dell'output del programma svolto.

Il prodotto finale di questa fase è solitamente chiamato versione alfa del software, mentre la successiva versione beta è quella che include le correzioni e miglioramenti suggeriti dai riscontri dei collaudatori (o *tester*) nella successiva fase detta, appunto, di collaudo, fino alla distribuzione della versione ultima che soddisfa le specifiche funzionali richieste.

## Tecnologia

 Lo stesso argomento in dettaglio: **Integrated development environment** e **Controllo di versione**.

La fase di implementazione coinvolge spesso numerose tecnologie relative non solo al prodotto, ma anche al processo che lo realizza. Per quanto riguarda il prodotto, la stesura dei programmi necessita di almeno un linguaggio di programmazione, di alcune sue librerie, ma spesso coinvolge altri linguaggi, per esempio di scripting, e altre tecnologie come le basi di dati o la stessa Internet.

Il processo di implementazione, invece, non può prescindere da un editor per la scrittura del codice sorgente, e di un compilatore o un interprete per collaudare l'esecuzione del codice; non di rado sono utilizzati strumenti di building automatico o di debugging. La scrittura della documentazione sul prodotto, invece, può essere generata in maniera automatica a partire dai commenti scritti all'interno del codice sorgente attraverso tool opportuni, grazie a strumenti come Doxygen o Javadoc. Se l'implementazione è condotta da una squadra di sviluppatori, la loro coordinazione è tipicamente assistita da sistemi di controllo delle revisioni (p.es. Subversion o GIT), che identificano ogni versione intermedia del prodotto.


Talvolta alcune, o molte, delle tecnologie necessarie alla produzione sono disponibili in un ambiente di sviluppo integrato, un'applicazione che unifica gli strumenti necessari al programmatore, o in kit di sviluppo (SDK), una distribuzione di documentazione e implementazione di un linguaggio di programmazione o piattaforma.

## Collaudo

 Lo stesso argomento in dettaglio: **Collaudo del software**.

Il collaudo o testing, appannaggio di uno o più tester, consiste nella verifica e validazione di quanto (misurabilità) il prodotto software implementato soddisfi i requisiti individuati dall'analisi. La relativa infrastruttura di supporto utilizzata in tale fase è detta ambiente di testing; il collaudo, in altre parole, valuta la correttezza rispetto alle specifiche. Anche per il collaudo possono essere individuate le due sottoattività di collaudo dei singoli moduli e collaudo del sistema integrato; inoltre possono essere individuate ulteriori sottoattività per ogni aspetto del prodotto software che interessa collaudare: collaudo funzionale, collaudo di performance, collaudo di rottura, collaudo di regressione, collaudo di sicurezza, collaudo di accessibilità, collaudo di accettazione, ecc. . In caso di mancato rispetto delle specifiche il software, assieme al documento delle anomalie o bug, torna indietro agli sviluppatori con il compito di risolvere i problemi riscontrati attraverso anche il debugging del software. In genere le anomalie di funzionamento sono gestite tramite appositi software gestori di segnalazione anomalie, detti anche sistemi di ticketing o di bug tracking, che registrano i problemi segnalati al team di sviluppo e ne facilitano la relativa organizzazione, classificazione e gestione (p. es. Bugzilla, Mantis, Atlassian Jira, ecc.).

## Rilascio e messa in esercizio

 Lo stesso argomento in dettaglio: **Deployment**.

Una volta che il software ha superato con successo le verifiche della fase di collaudo, esso viene pubblicato in una versione definitiva e messo a disposizione, secondo le regole della specifica licenza d'uso prescelta, di chiunque o dei soli acquirenti. La fase di installazione e messa in esercizio del software prodotto sulle piattaforme operative di utilizzo è detta deployment.

Talvolta si provvede anche alla messa in esercizio del software, cioè all'installazione e configurazione del prodotto software nell'infrastruttura di esecuzione utilizzabile dagli utenti, detta l'ambiente operativo o di produzione o di esercizio. A seconda della complessità di tale infrastruttura la messa in opera può essere scomposta in varie sottoattività; esso, infatti, può variare dalla semplice copia di un file, alla "copia opportuna" di molti file organizzati in una complessa gerarchia di directory e componenti software, eventualmente distribuiti su hardware differenti.

## Manutenzione

La manutenzione comprende quelle sottoattività necessarie a modifiche del prodotto software successive alla distribuzione, al fine di correggere ulteriori errori attraverso patch (manutenzione correttiva), adattarlo a nuovi ambienti operativi (manutenzione adattativa o migrazione di ambiente), estenderne le funzionalità (manutenzione evolutiva) o convertirne l'implementazione in un'altra struttura (es. migrazione di framework o piattaforma). La manutenzione incide sui costi per una stima che si aggira intorno al 60% dei costi totali. Ogni modifica al software comporta necessariamente la necessità di nuovi collaudi, sia relativi alle nuove funzionalità eventualmente introdotte, sia mirati a verificare che le modifiche apportate non abbiano compromesso funzionalità preesistenti (collaudo di regressione).

## Documentazione

Parallelamente alle fasi di sviluppo di cui sopra è abitudine e buona norma da parte del team di progettazione, sviluppo e testing provvedere alla redazione della corrispondente documentazione del

software che sarà corredata al software in fase finale o durante le singole fasi stesse a supporto della fase successiva.

## Ispezione del software

 Lo stesso argomento in dettaglio: [Ispezione del software](#).

Ciascuna fase di sviluppo del software descritta può ricadere nella cosiddetta ispezione del software.


## Fasi di sviluppo

---

### Pre-alpha

La versione Pre-alpha si riferisce a tutte le attività svolte durante il progetto software prima dei test formali. Queste attività possono includere analisi dei requisiti, progettazione del software, sviluppo del software e test delle unità. Nel tipico sviluppo open source, esistono diversi tipi di versioni pre-alfa.


### Alpha

 Lo stesso argomento in dettaglio: [Versione alpha](#).

La versione alpha individua il software in fase di sviluppo, le cui funzionalità non sono ancora state completamente implementate e che presenta sicuramente una lunga serie di bug che dovranno essere risolti.

Solitamente durante lo sviluppo di un software vengono realizzate diverse versioni alpha, in alcuni casi (soprattutto nel caso di software open-source) pubbliche, per introdurre nuove caratteristiche del software delle quali è necessario controllare il funzionamento a fondo. Le versioni alpha sono quindi da considerare incomplete e mancanti di alcune funzionalità e, spesso, instabili e perciò non adatte all'utente finale.

### Beta

 Lo stesso argomento in dettaglio: [Versione beta](#).

La versione beta è il software non definitivo, ma già testato dagli esperti e messo a disposizione di un numero maggiore di utenti (solitamente gli utenti finali), confidando proprio nelle loro azioni imprevedibili che potrebbero portare alla luce nuovi bug o incompatibilità del software stesso.<sup>[3]</sup>

## Pubblicazione

---

Una volta pubblicato, il software è generalmente noto come "versione stabile". Il termine formale spesso dipende dal metodo: media fisici, versione online o un'applicazione web.

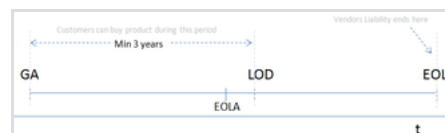
### Release to manufacturing (RTM)

Il termine *release to manufacturing* (**RTM**), (GA) viene utilizzato quando il prodotto viene distribuito al pubblico.

Viene tipicamente utilizzato in determinati contesti per la produzione di massa al dettaglio - al contrario di una produzione o progetto di software specializzato in una produzione o distribuzione commerciale o governativa - in cui il software viene venduto come parte di un pacchetto in una vendita di hardware e in genere il software e l'hardware correlato saranno in ultima analisi disponibili e venduti su larga scala/al pubblico nei punti vendita al dettaglio per indicare che il software ha raggiunto un livello di qualità definito ed è pronto per la distribuzione di massa al dettaglio. La RTM potrebbe anche significare in altri contesti che il software è stato consegnato a un cliente per l'installazione o la distribuzione ai computer o ai computer degli utenti finali correlati. Il termine non definisce il meccanismo o il volume di consegna; afferma solo che la qualità è sufficiente per la distribuzione di massa.

## Disponibilità generale (GA)

La **General availability** o **Disponibilità generale**, (GA) è la fase di marketing in cui tutte le attività di commercializzazione necessarie sono state completate e un prodotto software è disponibile per l'acquisto, a seconda, tuttavia, della lingua, della regione, della disponibilità elettronica rispetto ai media.<sup>[4]</sup> Le attività di commercializzazione potrebbero includere test di sicurezza e conformità, così come la localizzazione e la disponibilità mondiale. Il tempo tra la RTM e la GA può passare da una settimana a mesi, in alcuni casi, prima che possa essere dichiarata una release generalmente disponibile a causa del tempo necessario per completare tutte le attività di commercializzazione richieste dalla GA.



Ciclo di vita di un prodotto: disponibilità generale (GA), annuncio di fine vita (EOLA), data dell'ultimo ordine (LOD) e fine vita (EOL)

## Release to web (RTW)

**Release to the web** (RTW) è un mezzo di spedizione del software che utilizza Internet per la distribuzione. Il produttore non produce alcun supporto fisico in questo tipo di meccanismo di distribuzione. Le versioni web stanno diventando più comuni con l'aumentare dell'utilizzo di Internet.

## Processi di sviluppo software

La maggior parte delle metodologie di sviluppo del software consiste, almeno in linea di principio, in un linguaggio di modellazione e un processo.

1. Il linguaggio di modellazione è la notazione usata dalle metodologie per esprimere le caratteristiche di progetto;
2. il processo è l'elenco delle indicazioni riguardanti i passi da intraprendere per produrre il progetto stesso.

L'UML (*Unified Modeling Language*), ad esempio, è un linguaggio di modellazione, utilizzato dai processi per realizzare, organizzare, documentare i prodotti realizzati dalle fasi di cui il processo si compone. Coloro che, individualmente o in gruppo, lavorano allo sviluppo o alla modifica di un software,

adottano necessariamente un certo approccio nel modo di relazionarsi con i propri clienti/utenti, nell'organizzare il proprio lavoro, nella scelta delle tecniche da utilizzare, adottano un processo.

In modo consapevole o meno, ogni sviluppatore (o gruppo di sviluppatori) software ha un proprio processo di sviluppo, cioè un proprio modo di lavorare, basato sulla propria esperienza, sulla propria cultura, e sul contesto culturale e organizzativo in cui si trova a operare.

La storia dei successi (e soprattutto degli insuccessi) dei progetti di sviluppo software ha insegnato che ogni processo di sviluppo ha i propri pregi e i propri limiti e che un processo di sviluppo inadeguato alle concrete esigenze dello specifico progetto può condurre al fallimento del progetto stesso, o comunque all'insoddisfazione dei clienti/utenti e degli stessi sviluppatori.

Il ciclo di sviluppo del software, nella maggior parte dei casi, è iterativo, e ogni iterazione produce una sua release. Elenchiamo di seguito le fasi principali che possono far parte di ogni singola iterazione:

1. *Specifica dei requisiti*: ciò che viene richiesto dal committente;
2. *Studio di fattibilità (make or buy)*;
3. *Analisi e Design* dove per analisi si intende lo studio di cosa deve fare il sistema (punto di vista "logico") e per design lo studio di come il sistema deve essere implementato (punto di vista "tecnico"). Negli ultimi anni, la distinzione tradizionale tra analisi e design è entrata in crisi.
4. *Implementazione*;
5. *Integrazione e test*.

Se originariamente i modelli di ciclo di vita del software erano strumenti di natura esclusivamente concettuale, i moderni ambienti di sviluppo CASE (*Computer aided software engineering*) spesso automatizzano l'applicazione di un particolare ciclo di vita così come di altri aspetti dello sviluppo del software.

## Alcuni esempi di metodologia di sviluppo del software



Lo stesso argomento in dettaglio: *Metodologia di sviluppo del software*.

- Modello a cascata
- Modello evolutivo
- Modello trasformatzionale
- Prototipazione rapida
- Modello a spirale
- Modello a fontana (metamodello)
- Metodologia agile
- Modello a stella

### Modello a cascata



Lo stesso argomento in dettaglio: *Modello a cascata*.

Storicamente, il modello a cascata fu il primo modello di ciclo di vita del software. Prevede l'esecuzione



sequenziale delle fasi di *analisi, progetto, sviluppo, collaudo e manutenzione*.<sup>[5]</sup>

## Modelli iterativi


Nella categoria dei modelli iterativi rientrano tutti i processi di sviluppo che permettono di ritornare a una fase del procedimento già affrontata in precedenza. Per ogni fase "iterabile" si tende a partire con un abbozzo della soluzione che verrà successivamente raffinata al prossimo passaggio.

I modelli incrementali ed evolutivi sono una specificazione dei modelli iterativi.

## Modelli incrementali

 Lo stesso argomento in dettaglio: **Modello incrementale**.

## Modelli evolutivi

 Lo stesso argomento in dettaglio: **Modello evolutivo**.

Secondo la visione più comune nella moderna ingegneria del software, il modello a cascata non è più adatto a essere applicato, almeno per la maggior parte delle applicazioni. Fra gli elementi che hanno contribuito al tramonto di questo modello ci sono l'aumento di complessità delle applicazioni, l'introduzione delle GUI (che hanno portato all'attenzione dei progettisti questioni di usabilità precedentemente di scarso rilievo), e l'avvento di nuovi paradigmi di programmazione e metodologie di analisi e progetto, in particolare quelle legate al paradigma *object-oriented*.

In particolare, si sono affermati una vasta classe di cosiddetti modelli evolutivi, tutti basati sull'idea centrale di coinvolgere maggiormente il committente (o l'utente), sottoponendogli periodicamente versioni parziali o prototipi del prodotto finale. Una tecnica tradizionale in questo senso è, per esempio, quella della prototipazione rapida. In questa linea di pensiero una delle ultime generazioni è quella delle cosiddette metodologie agili.

## Note

---

- <sup>1</sup> <sup>^</sup>, <sup>([EN](#))</sup> N. D. Birrell e Martyn A. Ould, *A Practical Handbook for Software Development*, Cambridge University Press, 1988, ISBN [978-0-521-34792-1](#).
- <sup>2</sup> <sup>^</sup> *Evoluzione dello sviluppo software*, su [glennbouchard.com](#). URL consultato il 13 aprile 2022.
- <sup>3</sup> <sup>^</sup> *Cosa si intende per versione beta di un software?*, su [01net.it](#). URL consultato il 17 agosto 2022.
- <sup>4</sup> <sup>^</sup> Yvan Philippe Luxembourg, *Top 200 SAM Terms – A Glossary Of Software Asset Management Terms*, OMTCO, 20 maggio 2013. URL consultato il 21 maggio 2013 ([archiviato](#) il 10 agosto 2013).
- <sup>5</sup> <sup>^</sup> *Il ciclo di vita del software: il Modello a cascata*, su [brainybyte.it](#). URL consultato il 17 agosto 2022.

## Voci correlate

---



- [ISO 12207](#)
- [Controllo versione](#)
- [Modello di sviluppo del software](#)
- [Software](#)
- [Software house](#)

## Collegamenti esterni

---

- **([EN](#))** Denis Howe, *[software life-cycle](#)*, in *[Free On-line Dictionary of Computing](#)*. Disponibile con licenza [GFDL](#)



**Portale Informatica**



**Portale Ingegneria**

---

Estratto da "[https://it.wikipedia.org/w/index.php?title=Ciclo\\_di\\_vita\\_del\\_software&oldid=142190157](https://it.wikipedia.org/w/index.php?title=Ciclo_di_vita_del_software&oldid=142190157)"