Indice

1 – Introduzione

L'ingegneria del software (*software engineering*) negli ultimi anni è diventata molto importante ed è sempre più presente nella vita di tutti i giorni. Per esempio, l'economia di tutte le nazioni dipende dal "software", ovvero dalla progettazione di sistemi digitali. La fase di sviluppo, di rilascio di un software è l'ingegneria del software. Non a caso, la spesa per lo sviluppo di software influisce radicalmente sul PIL (Prodotto Interno Lordo) di un paese.

Ovviamente, solitamente il costo di **inizio** sviluppo di un software per un calcolatore è minore rispetto a quello dell'hardware. Paradossalmente, il costo di **mantenimento** del software è maggiore al costo di avviamento, o creazione, del progetto. L'ingegneria del software si concentra sul rapporto costo/efficacia del software sviluppato.

L'importanza di utilizzare alcuni principi dell'ingegneria del software influenza il risultato finale di un progetto. Infatti, l'utilizzo errato di tali metodi aumenta i costi di mantenimento a lungo termine, ostacola la facile individuazione di errori di sistema, aumenta il periodo di sviluppo di eventuali aggiornamenti e altro. In sintesi, il software diventerà molto costoso e poco affidabile di quanto possa essere se venissero seguiti i principi.

1.1 – Definizioni

Definizione di software

Un **software** è identificabile come il programma, ovvero il prodotto, creato per poter essere eseguito su un computer, insieme alla sua documentazione. Inoltre, esso può essere sviluppato per un cliente specifico o per un mercato generale.

Attributi per definire un software "buono"

Un **buon software** è tale quando il funzionamento e le prestazioni rispettano l'obbiettivo per cui è stato creato (verificabile soprattutto dall'utente finale), è mantenibile a lungo termine, affidabile e utilizzabile.

Definizione di ingegneria del software

L'**ingegneria del software** è una disciplina ingegneristica che racchiude tutti gli aspetti della produzione di un software.

Attività fondamentali dell'ingegneria del software

Nonostante gli approfondimenti che verranno effettuati in futuro, le attività fondamentali (top-down) sono: **specificazione** del software, **sviluppo** del software, **convalidazione** del software e **evoluzione** del software.

Differenza tra l'ingegneria del software e l'ingegneria del sistema (*system engineering*)

L'ingegneria del sistema si interessa di tutti gli aspetti dello sviluppo di sistemi basati sul computer, quindi sia hardware, sia software, sia ingegneria del processo (*process engineering*). L'ingegneria del software è una parte di questi processi più generali.

Obbiettivi dell'ingegneria del software

Gli obbiettivi dell'ingegneria del software sono: facilità di far fronte alla diversità di richieste del mercato; diminuire il tempo di sviluppo senza perdere qualità, quindi ottimizzazione del tempo; sviluppare software affidabili.

Costi dell'ingegneria del software

Generalmente intorno al 60% i costi servono per lo sviluppo e il restante 40% per le fasi di testing. Per progetti specifici, solitamente i costi di evoluzione superano i costi di sviluppo.

Per concludere, non esiste un unico metodo per rendere un progetto ottimo, ogni prodotto ha il suo sistema più adatto.

1.2 – Tipi di software

Un software, o prodotto, può essere generico o specifico.

Un prodotto **generico** ha la caratteristica principale di essere creato e modificato a piacimento dallo sviluppatore.

Al contrario, un prodotto **specifico** viene creato sempre da uno sviluppatore ma dovrà eseguire i compiti richiesti dal cliente. Quindi, anche le modifiche dovranno essere eseguite a seconda sempre delle richieste del cliente.

1.3 – Attributi di un "buon" software

Come detto precedentemente, un software può essere definito "buono" nel caso in cui soddisfi le seguenti caratteristiche:

- **Manutenibilità**. Il software è scritto in modo che possa evolvere a seconda delle esigenze del cliente. Questo attributo viene considerato fondamentale poiché il cambiamento del software è un requisito indispensabile per un ambiente aziendale in evoluzione.
- **Affidabilità e sicurezza**. Il software deve essere affidabile. Quest'ultimo requisito include una serie di caratteristiche, tra le quali anche la sicurezza. Ossia la gestione sicura dei dati, evitare che il software sia un virus dannoso per l'utente e anche una affidabilità fisica/economica. Per esempio, un software è affidabile se durante l'utilizzo **non** si verifica un malfunzionamento anomalo che possa provocare danni fisici ingenti all'utente o che **non** esplode provocando danni per riparare il dispositivo.
- **Efficienza**. Il software deve ottimizzare le risorse a disposizione. Ovvero, **non** deve fare un uso dispendioso delle risorse di sistema quali, memoria, processore, ecc.
- **Accettabilità**. Il software deve essere facilmente utilizzabile dal target di utenza a cui è destinato. Questo significa che esso dovrà essere comprensibile, utilizzabile e compatibile con i maggior sistemi operativi in circolazione.

1.4 – L'importanza dell'Ingegneria del Software

Sempre più individui e società fanno affidamento sull'avanzamento dei sistemi software. Per questo motivo si è alla ricerca sempre di software sicuri, economici e rapidi.

Solitamente, nel lungo periodo è economico usare metodi e tecniche di ingegneria del software piuttosto di scrivere semplicemente programmi come se fossero progetti privati. Nella maggior parte dei sistemi, il costo più alto si manifesta quando vengono effettuati dei cambiamenti al software dopo che quest'ultimo è già terminato (nel caso in cui non vengano utilizzate tecniche e metodi di ingegneria del software).

1.5 – Attività di processo software

- **Specifica software** (*software specification*), fase in cui i clienti e gli ingegneri definiscono il software che deve essere prodotto e le operazioni che deve eseguire;
- **Sviluppo software** (*software development*), fase in cui il software viene progettato e programmato;
- **Validazione software** (*software validation*), fase in cui il software viene controllato per verificare che rispetti ciò che il cliente ha richiesto;
- **Evoluzione software** (*software evolution*), fase in cui il software viene modificato per soddisfare i requisiti del mercato ed i cambiamenti del cliente.

1.6 – Questioni che riguardano il software

Ci sono alcune questioni che anche indirettamente colpiscono lo sviluppo software. Qui di seguito vengono esposti 4 questioni: eterogeneità, cambiamenti economici e sociali, sicurezza e affidabilità, scalabilità.

Per **eterogeneità** (*Heterogeneity*) si intende la capacità del software di far fronte alla grande vastità di sistemi distribuiti. Per esempio, alla possibilità di essere eseguito su molteplici sistemi operativi, oppure alla possibilità di essere eseguito su cloud, ecc.

Per **cambiamenti economici e sociali** (*business and social change*) si intende che a causa dei cambiamenti rapidi dell'economia, del mercato, delle situazioni sociali, l'ingegneria del software deve essere rapida a trovare un nuovo modo per implementare nuove tecnologie nei software già esistenti (quindi aggiornando quest'ultimi) o nell'adoperare nuove tecnologie nella creazione di nuovi software.

Per **sicurezza e affidabilità** (*security and trust*) si intende l'abilità del software a trasmettere la sicurezza e affidabilità all'utente finale. Ovviamente, queste due qualità devono essere dimostrate tramite prove concrete, risultati e quant'altro.

Per **scalabilità** (*scale*) si intende che il software deve essere sviluppato per una vasta scala di sistemi, dai piccoli sistemi integrati nelle unirà portabili ai più estesi sistemi cloud di Internet che servono una comunità più vasta di utenti.

1.7 – Tipi di applicazioni e concetti fondamentali

Non esistono metodi di ingegneria del software universali che sono applicabili a tutti i sistemi e a tutte le aziende. Tuttavia, il fattore più significativo per determinare quali metodi e tecniche di ingegneria del software siano più rilevanti è il tipo di applicazione che si vuole sviluppare:

- **Applicazioni autonome**. Sono applicazioni che possono essere eseguite su un PC o su un dispositivo mobile. Includono tutte le funzionalità necessarie e potrebbero non richiedere una connessione a una rete. Un esempio sono i programmi CAD.
- **Applicazioni interattive basate sulle transazioni**. Sono applicazioni eseguite su un computer remoto. Il loro accesso è possibile attraverso un calcolatore e solitamente ogni utente ha un suo account protetto da e-mail e password. Un esempio sono le applicazioni web per il commercio elettronico ma anche applicazioni aziendali.
- **Sistemi di controllo integrati**. Sono sistemi che controllano e gestiscono dispositivi hardware. Degli esempi sono il software di un cellulare e il software che controlla l'impianto antibloccaggio dei freni di un'automobile.
- **Sistemi di elaborazione batch**. Sono sistemi aziendali progettati per elaborare grandi blocchi di dati in sequenza.
- **Sistemi di intrattenimento**. Sono sistemi per uso personale progettati per il divertimento dell'utente.
- **Sistemi per la modellazione e la simulazione**. Sono sistemi sviluppati da scienziati e ingegneri per creare modelli di situazioni o processi fisici.
- **Sistemi per la raccolta e l'analisi dei dati**. I sistemi di raccolta dei dati sono sistemi che raccolgono i dati dai loro ambienti e li inviano ad altri sistemi per l'elaborazione.
- **Sistemi di sistemi**. Sono sistemi utilizzati nelle aziende e in altre grandi organizzazioni che sono composti da un certo numero di altri sistemi software.

Tuttavia, esistono alcuni concetti fondamentali di ingegneria del software che si applicano a tutti i tipi di sistemi software:

1. I **sistemi** devono essere sviluppati utilizzando un **processo di sviluppo chiaro e** accuratamente **pianificato**. È necessario quindi pianificare il processo di sviluppo e avere idee chiare su cosa sarà prodotto e quando sarà completato.
2. La **fidatezza e** le **prestazioni** sono **importanti**. Il software dovrà comportarsi come previsto, senza fallimenti, disponibile all'uso quando richiesto. Il suo funzionamento dovrà essere sicuro e protetto il più possibile contro attacchi esterni. Il sistema dovrà essere eseguito con efficienza, senza spreco di risorse.
3. È importante **capire e gestire** la **specifica e** i **requisiti** del **software** (cosa deve fare il software). Occorre sapere che cosa si aspettano i differenti clienti e utenti del sistema e occorre gestire le loro aspettative in modo che possa essere rilasciato un sistema utile entro i costi e i tempi previsti.
4. Occorre **utilizzare con efficienza le risorse esistenti**, quindi, se possibile, riusare il software che è già stato sviluppato anziché scrivere un nuovo software.

1.8 – Etica dell'ingegneria del software

L'ingegneria del software viene svolta in un ambito legale e sociale che limita la libertà degli ingegneri. Infatti, quest'ultimi devono comprendere che il loro lavoro comporta responsabilità più vaste della semplice applicazione di capacità tecniche, e devono agire in maniera eticamente e moralmente responsabile se vogliono essere rispettati come professionisti.

Esistono alcune aree in cui lo standard comportamentale non è limitato da leggi, ma più da tenui nozioni di responsabilità professionale:

- **Riservatezza**: si deve rispettare la riservatezza dei propri datori di lavoro e dei propri clienti, che sia stato firmato un accordo formale o meno;
- **Competenza**: non bisogna mentire sul proprio livello di competenza, né accettare lavori al di sopra delle proprie capacità;
- **Diritti di proprietà intellettuale**: si deve conoscere le leggi locali che governano l'uso delle proprietà intellettuali, come i brevetti e il diritto d'autore (*copyright*), in modo da proteggere quelle del proprio datore di lavoro e dei propri clienti;
- **Uso improprio del computer**: non bisogna sfruttare le proprie capacità tecniche per un cattivo uso dei computer altrui. L'uso improprio del computer varia dal banale (giocare con il computer aziendale) all'estremamente serio (disseminazione di virus).

Società professionali e istituzioni hanno un ruolo importante nell'impostare gli standard etici: organizzazioni come ACM (*Association for Computing Machinery*, associazione dei macchinari di calcolo), IEEE (*Institute of Electrical and Electronic Engineers*, istituto degli ingegneri elettrotecnici ed elettronici) e BCS (*British Computer Society*, società informatica inglese) **pubblicano un codice di condotta professionale o codice etico, sottoscritto dagli associati all'atto dell'iscrizione**.

Verranno presentati esempi negli esercizi (casi di studio) che potranno costruire punti di partenza per una discussione di gruppo sui problemi etici.

1.9 – Casi di studio

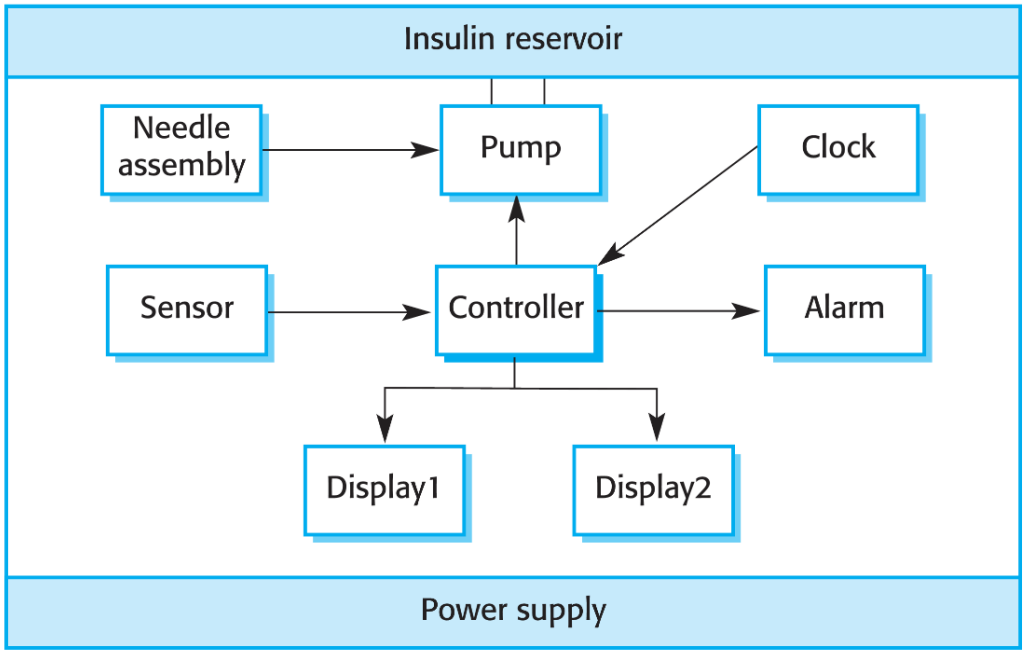
Per illustrare i concetti di ingegneria del software, verranno esposti quattro esempi di differenti tipi di sistema:

1. Il primo sarà un **sistema integrato**. Ovvero un sistema in cui il software controlla un dispositivo hardware integrato nel dispositivo. I problemi di un sistema del genere sono generalmente la dimensione fisica, la rapidità di risposta e la gestione della potenza. Il sistema scelto come esempio sarà una pompa di insulina utilizzata nel trattamento di pazienti diabetici.
2. Il secondo sarà un **sistema informativo**. L'obbiettivo principale è gestire e fornire l'accesso a un database di informazioni. I problemi relativi a questo sistema sono la protezione, l'usabilità, la riservatezza e l'integrità dei dati. L'esempio di sistema informativo scelto è un sistema di registrazione medica.
3. Il terzo sarà un **sistema di raccolta dati basata su sensori**. Gli obbiettivi principali sono la raccolta dei dati da un insieme di sensori e l'elaborazione di tali dati. I requisiti sono l'affidabilità, in qualsiasi situazione, e la manutenibilità. L'esempio scelto è una stazione meteorologica in un'area selvaggia.
4. Il quarto e ultimo sarà un **ambiente di supporto**. Questo sistema è formato da un insieme integrato di strumenti software che sono utilizzati per supportare qualche tipo di attività. L'esempio scelto è un ambiente di apprendimento digitale che viene utilizzato per supportare l'apprendimento degli studenti nelle scuole.

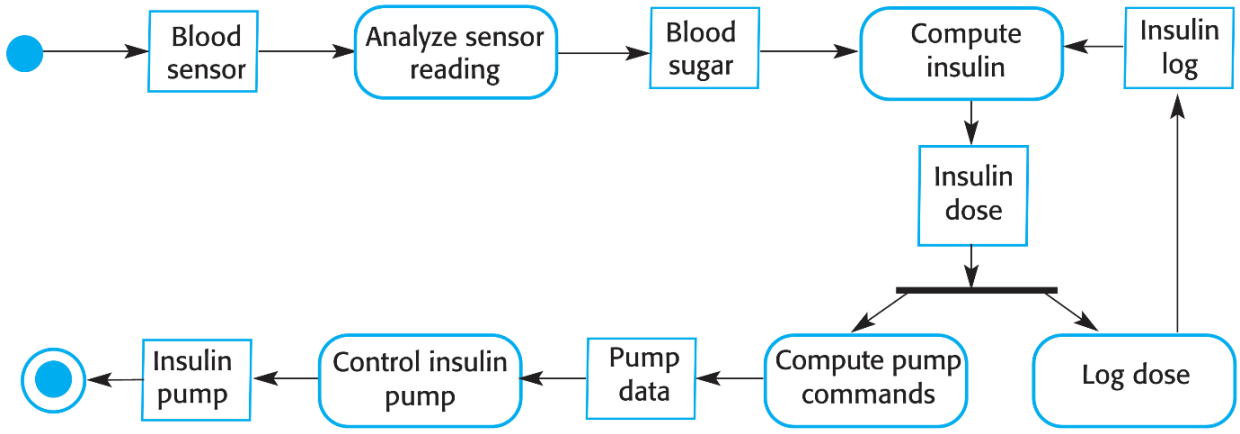
1.9.1 – Sistema di controllo di una pompa di insulina

Una pompa di insulina è un sistema medico che simula il funzionamento del pancreas (un organo interno). Il software è un **sistema integrato** che riceve informazioni da un sensore e controlla una pompa che rilascia una dose controllata di insulina a un paziente.

I componenti hardware e l'organizzazione sono mostrati nella seguente figura:



Il seguente modello UML (*Unified Modeling Language*, linguaggio di modellazione unificato) illustra come il livello iniziale degli zuccheri nel sangue si trasforma in una sequenza di comandi di controllo per la pompa:



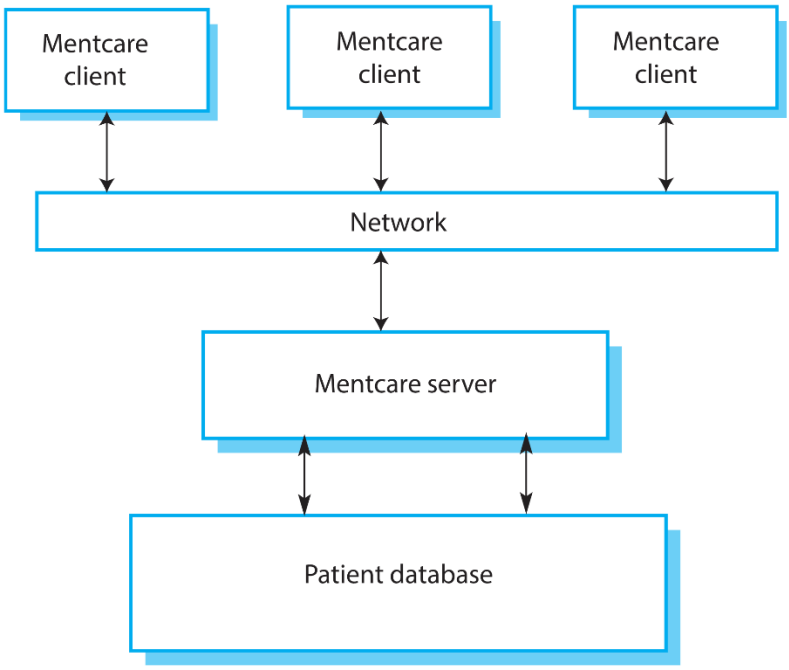
Questo è un sistema a **sicurezza critica**. Infatti, nel caso in cui la pompa si guasta o non funziona correttamente, il paziente può subire gravi danni o entrare in coma. Per cui, per provvedere a questo problema, il sistema deve soddisfare due requisiti fondamentali:

1. Deve essere disponibile a fornire l'insulina quando richiesta;
2. Deve comportarsi in modo affidabile e fornire il giusto quantitativo di insulina per correggere il livello corrente di zuccheri nel sangue.

Il sistema deve essere progettato e implementato in modo che questi due requisiti siano **sempre garantiti**.

1.9.2 – Sistema informativo di una clinica psichiatrica

Mentcare è un sistema informativo medico che registra informazioni su pazienti con problemi di salute mentale e le cure cui sono sottoposti. Molti di questi pazienti non richiedono cure ospedaliere dedicate, tuttavia devono frequentare regolarmente delle cliniche specialistiche.



Il sistema Mentcare è stato ideato per essere utilizzato nelle cliniche; utilizza un database centralizzato con le informazioni sui pazienti. Può essere eseguito anche da un computer portatile, in modo che possa essere utilizzato da siti che non hanno una connessione di rete protetta. Quando i sistemi accedono alla rete, possono visualizzare le informazioni sui pazienti registrati nel database e scaricare copie delle cartelle cliniche dei pazienti.

La sua organizzazione è rappresentata in figura.

Mentcare non è un sistema medico completo, per cui non contiene informazioni su altre condizioni mediche dei pazienti. Tuttavia, può interagire e scambiare dati con altri sistemi informativi medici.

Gli **obbiettivi** del sistema sono due:

1. **Generare** le **informazioni** che consentono ai responsabili dei servizi sanitari di valutare le condizioni cliniche rispetto agli obbiettivi locali o governativi;
2. **Fornire** al personale medico **informazioni tempestive** per definire le cure dei pazienti.

Le **caratteristiche chiave** del sistema invece sono:

1. **Gestione delle cure individuali**. Il personale medico può creare cartelle cliniche per i pazienti, modificare le informazioni registrate nel sistema, esaminare la storia clinica dei pazienti e così via.
2. **Monitoraggio dei pazienti**. Il sistema controlla regolarmente le cartelle cliniche dei pazienti che sono in cura ed emette degli avvisi se vengono rilevati determinati problemi. Uno degli elementi più importanti del sistema di monitoraggio è quello di tenere traccia dei pazienti che sono stati isolati e garantire che i controlli previsti dalla legge siano effettuati al momento giusto.
3. **Report amministrativo**. Il sistema genera mensilmente dei rapporti gestionali che mostrano il numero di pazienti trattati in ogni clinica, il numero di pazienti che hanno iniziato o finito una cura, il numero di pazienti isolati, farmaci prescritti e i loro costi, e così via.

Dal punto di vista dell'ingegneria del software, la **riservatezza** è un **requisito critico**. È essenziale che le informazioni sui pazienti siano confidenziali e non siano mai divulgate a nessuno, tranne al personale autorizzato ed i pazienti stessi.

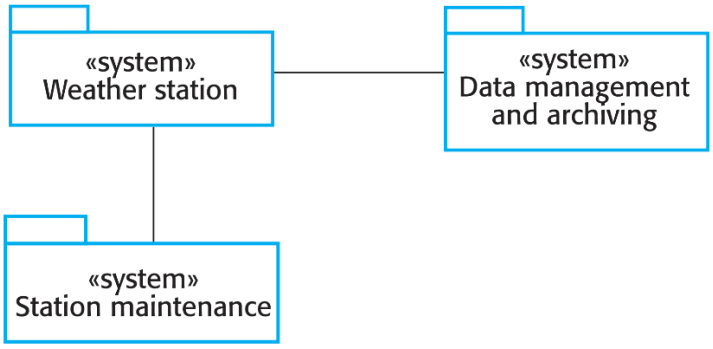
Mentcare è anche un **sistema** a **sicurezza critica**. Difatti, alcune malattie mentali potrebbero portare i pazienti al suicidio o ad arrecare danni ad altri. Ove possibile, il sistema dovrebbe segnalare al personale medico i pazienti potenzialmente suicidi o pericolosi.

Per concludere, il progetto del sistema deve tenere in considerazione i requisiti di riservatezza e sicurezza. Il sistema deve essere disponibile quando richiesto, altrimenti la sicurezza potrebbe essere compromessa e potrebbe essere impossibile prescrivere le cure appropriate ai pazienti. Purtroppo, si presenta un **potenziale conflitto**. La **riservatezza** è più facile da mantenere quando c'è una singola copia dei dati del sistema, ma in questo caso per **garantire la disponibilità** nel caso di guasto del server o di disconnessione da una rete, dovrebbero essere mantenute più copie dei dati.

1.9.3 – Stazione meteorologica in un'area selvaggia

Per agevolare il monitoraggio delle variazioni climatiche e migliorare l'accuratezza delle previsioni meteorologiche in aree remote, il governo di una nazione con vaste aree selvagge ha deciso di installare centinaia di stazioni meteorologiche in queste zone.

Le stazioni fanno parte di un sistema meteorologico più complesso, il quale raccoglie i dati delle stazioni e li mette a disposizione di altri sistemi dove vengono elaborati. I sistemi, riassunti nella seguente immagine, sono:



1. **Sistema di stazioni meteorologiche**. Questo sistema ha il compito di raccogliere i dati meteorologici, effettuare una prima elaborazione di questi dati e trasmetterli a un sistema di gestione.
2. **Sistema di gestione e archiviazione dei dati**. Questo sistema riceve i dati da tutte le stazioni meteorologiche, elabora e analizza i dati, e li archivia in un formato che può essere letto da altri sistemi, come quelli che effettuano le previsioni meteorologiche.
3. **Sistema di manutenzione delle stazioni**. Questo sistema può comunicare via satellite con tutte le stazioni meteorologiche per monitorare il corretto funzionamento e fornire un rapporto su eventuali problemi. Può aggiornare il software integrato nelle stazioni. Nel caso di problemi, questo sistema può essere utilizzato anche per controllare in modo remoto le stazioni.

Nell'immagini viene utilizzato il simbolo di package UML per indicare ciascun sistema è un insieme di componenti e i singoli sistemi sono identificati tramite lo stereotipo: UML $\ll system \gg$.

Le stazioni meteorologiche includono gli strumenti che misurano i parametri atmosferici e ciascuno è controllato da un sistema software che riceve periodicamente i valori dei parametri e gestisce i dati forniti dagli strumenti.

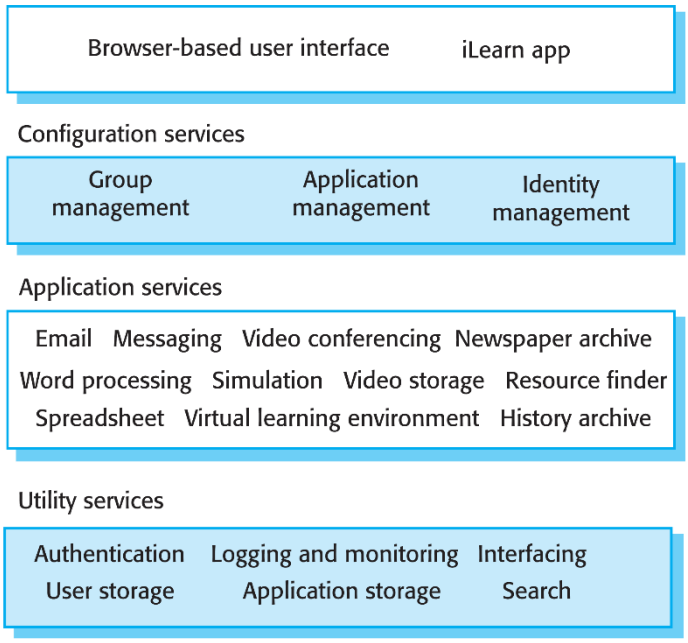
Quindi, il **software** di una stazione meteorologica ha il compito di raccogliere i dati e:

1. **Monitorare** gli strumenti, la carica delle batterie e l'hardware delle comunicazioni e riportare eventuali guasti del sistema di gestione.
2. **Gestire** l'alimentazione elettrica del sistema, assicurando che le batterie vengano caricate ogni volta ce le condizioni ambientali lo permettano, ma anche che i generatori vengano spenti quando le condizioni atmosferiche diventano potenzialmente pericolose, come nel caso di forti raffiche di vento.
3. **Consentire** la **riconfigurazione dinamica** quando alcune parti del software vengono sostituite con nuove versioni o quando intervengono gli strumenti di backup nel caso di guasto del sistema.

1.9.4 – Ambiente di apprendimento digitale

Un ambiente di apprendimento digitale è un sistema che si serve di una serie di strumenti generici e specifici per l'apprendimento, oltre a una serie di applicazione che sono adatte alle esigenze degli studenti che usano il sistema. Questo sistema di apprendimento fornisce i servizi generali, quali il servizio di autenticazione, i servizi di comunicazione sincrona e asincrona, e un servizio di immagazzinamento dei dati.

Il modello architetturale ad alto livello di un ambiente di apprendimento digitale (*iLearn*) che è stato progettato per scuole di studenti di età compresa fra 3 e 18 anni è l'immagine a fianco.



I tre **tipi di servizi** nel sistema sono:

1. **Servizi di utilità**. Forniscono le funzionalità di base indipendenti dalle applicazioni, le quali possono essere utilizzare da altri servizi del sistema. I servizi di utilità di solito sono sviluppati o adattati specificatamente per questo sistema.
2. **Servizi di applicazioni**. Forniscono applicazioni specifiche, come e-mail, condivisione di foto e videoconferenza, e consentono di accedere a materiali specifici, quali filmati scientifici o documenti storici.
3. **Servizi di configurazione**. Sono utilizzati per adattare l'ambiente di apprendimento a specifici servizi di applicazioni e per definire come i servizi devono essere condivisi tra insegnanti, studenti e loro genitori.

Il sistema deve supportare **due livelli di integrazione dei servizi**:

1. **Servizi integrati**. Sono servizi che offrono un'API (*Application Programming Interface*) e che sono accessibili da altri servizi tramite tale API. Un servizio di autenticazione è un esempio di servizio integrato. Anziché utilizzare i loro meccanismi di autenticazione, altri servizi possono chiamare un servizio di autenticazione per autenticare gli utenti. Se gli utenti sono già autenticati, allora il servizio di autenticazione può passare le informazioni di autenticazione direttamente a un altro servizio, tramite un'API, senza bisogno che gli utenti si autentichino di nuovo.
2. **Servizi indipendenti**. Sono servizi che sono accessibili semplicemente tramite un browser e che funzionano indipendentemente da altri servizi. Le informazioni possono essere condivise con altri tramite esplicite azioni dell'utente, come la richiesta di copia e incolla; potrebbe essere necessario ripetere l'autenticazione per ogni servizio indipendente.