POLITECNICO DI BARI

DIPARTIMENTO DI INGEGNERIA ELETTRICA E DELL'INFORMAZIONE

CORSO DI LAUREA TRIENNALE IN

INGEGNERIA INFORMATICA E DELL'AUTOMAZIONE

TESI DI LAUREA IN INGEGNERIA DEL SOFTWARE

SYSTEMATIC MAPPING STUDY SU SOFTWARE ECOSYSTEM

Relatore:

Prof.ssa Marina MONGIELLO

Correlatore:

Dott. Francesco NOCERA

Laureando:

Domenico SACCO

Anno Accademico 2015/2016

INDICE

PREFAZIONE

Lo scopo di questo lavoro di tesi è quello di fornire una panoramica sugli ecosistemi software, i quali hanno subito in tempi recenti una rapida diffusione. Viene di seguito usato come metodo di ricerca e classificazione per tali studi il Systematic Mapping Study, il quale permette di avere un quadro ben definito di un argomento così esteso.

Oggetto di questo lavoro infatti non è solo la trattazione degli ecosistemi software e delle loro applicazioni, ma anche la valutazione dello stato di avanzamento della ricerca in questo settore e dello sviluppo che questo ha avuto nel corso degli ultimi anni.

La prima parte mira a mostrare le principali caratteristiche del Systematic Mapping Study, descrivendone in dettaglio il funzionamento nelle sue singole fasi e spiegando i motivi per i quali è stato impiegato, illustrando pro e contro di tale tipo di analisi.

Successivamente vengono analizzati gli ecosistemi software, soffermandosi su esempi pratici ed evidenziando in particolare il ritorno economico derivante dal loro impiego.

Nella terza parte viene descritta l'applicazione del Systematic Mapping Study sui SECO (Software Ecosystems), in particolare le Research Questions proposte e la ricerca degli articoli sulle più importanti biblioteche digitali.

Nella quarta fase vengono estratti i dati sui SECO ed esposta l'analisi fatta sui risultati. Successivamente si descrivono le caratteristiche fondamentali degli ecosistemi software, i principali attori coinvolti, l'architettura ed i problemi principali che si incontrano nell'impiego dei SECO.

Il lavoro si conclude con delle considerazioni finali, le quali riguardano la situazione attuale del mercato, delle applicazioni dei SECO ed il loro stato di avanzamento nella ricerca.

CAPITOLO 1:

LO STUDIO SISTEMATICO

A seguito dell'avanzamento in un determinato ambito della ricerca scientifica, vi è anche un aumento del materiale reso disponibile dalle pubblicazioni, il quale presenta la necessità di essere adeguatamente catalogato.

Negli ambiti scientifici più vecchi come la medicina, la classificazione del materiale segue delle metodologie già specificate e collaudate.

Nel caso dell'Ingegneria del Software invece, dato il notevole tasso di sviluppo di questa disciplina, non risulta banale una metodologia in grado di catalogare adeguatamente la grande mole degli studi pubblicati ogni anno.

Nel 2004, una tendenza generale verso il paradigma evidence-based ha portato ad impiegarlo anche nell'Ingegneria del Software, ne è scaturita una maggiore attenzione relativa ai metodi di ricerca di natura empirica e sistematica [1].

Numerose sono le motivazioni per le quali è richiesto uno studio di natura sistematica, fra le quali le più importanti sono:

- Ottenere una visione d'insieme di carattere riassuntivo;
- Analizzare sviluppi futuri in un certo campo;
- Studiare prove empiriche per determinare avvaloramenti o contraddizioni con ipotesi teoriche;
- Partire dalle evidenze empiriche di alcuni studi per creare una nuova ipotesi.

Una tra le principali metodologie impiegate per la classificazione degli studi scientifici è la Systematic Literature Review, la quale prevede un'approfondita revisione dei documenti appartenenti ad un determinato ambito, alla quale segue la descrizione generale dei metodi impiegati negli studi e del risultato a cui questi portano. In alternativa si possono trarre conclusioni dai confronti eseguiti fra i documenti studiati.

Un processo di revisione sistematico deve documentare accuratamente le strategie di ricerca sulle quali si basa ed i criteri di qualità con cui valuta gli studi che analizza, tali spiegazioni sono necessarie per certificare l'attendibilità della revisione svolta e convincere i lettori circa la validità del lavoro di revisione svolto.

Se da un lato tale tipo di revisione risulta vantaggiosa perché, in quanto approfondita, riduce la possibilità di errori e distorsioni nell'analisi, risulta comunque gravosa in termini di sforzo fatto per via del livello di approfondimento così elevato.

Un'alternativa alla Systematic Literature Review è appunto il Systematic Mapping Study, il quale permette di avere una visione più generale, ma al contempo richiede un minore sforzo per ottenere questo tipo di analisi.

1.1 Caratteristiche fondamentali del SMS

Come già specificato in precedenza, il Systematic Mapping Study è pensato per fornire una panoramica generale di un determinato argomento, infatti a differenza della Systematic Literature Review, è in grado di partire con più di una domanda a cui rispondere, in quanto lo scopo generale è appunto quello di mappare l'area di ricerca [2].

I due tipi di analisi infatti differiscono anche per la finalità per cui sono predisposte: il Systematic Mapping Study viene principalmente impiegato per analizzare lo stato di

avanzamento della ricerca, individuando i punti in cui risulta più o meno approfondita, la Systematic Literature Review serve invece per una rigorosa classificazione e identificazione delle pubblicazioni [3][4].

Data la natura meno approfondita del SMS, ne consegue che i criteri di ricerca impiegati saranno meno concentrati e che l'analisi potrà comprendere anche un numero molto elevato di studi.

Gli aspetti vantaggiosi del SMS più importanti risiedono nella possibilità di combinare i risultati ottenuti e dal grande numero di pubblicazioni che vengono analizzate, le quali forniscono un notevole contenuto informativo circa fenomeni ed effetti di natura empirica, trattandoli con diverse impostazioni da vari punti di vista [3].

Fra gli svantaggi tuttavia si annoverano lo sforzo computazionale maggiore derivante dai confronti di un così elevato numero di studi e la possibilità di distorsioni nelle meta-analisi, occorrenza tuttavia rara in caso di impiego di una metodologia ben definita e collaudata [5].

1.2 Fasi del SMS

Le tre macrofasi in cui si divide il Systematic Mapping Study sono:

- **Direttive di ricerca:** Si tratta per prima cosa di definire il protocollo di ricerca e successivamente di determinare le domande a cui questa deve rispondere.
- **Raccolta dati:** Determina le fonti da cui prelevare i dati e l'eventuale stringa da inserire nei database per effettuare la ricerca.
- **Risultati:** Si estraggono i dati dagli studi più importanti e questi vengono analizzati e classificati, portando alla creazione della Systematic Map.

Lo schema può essere visto come segue, in alto sono descritte le azioni ed in basso il loro risultato:

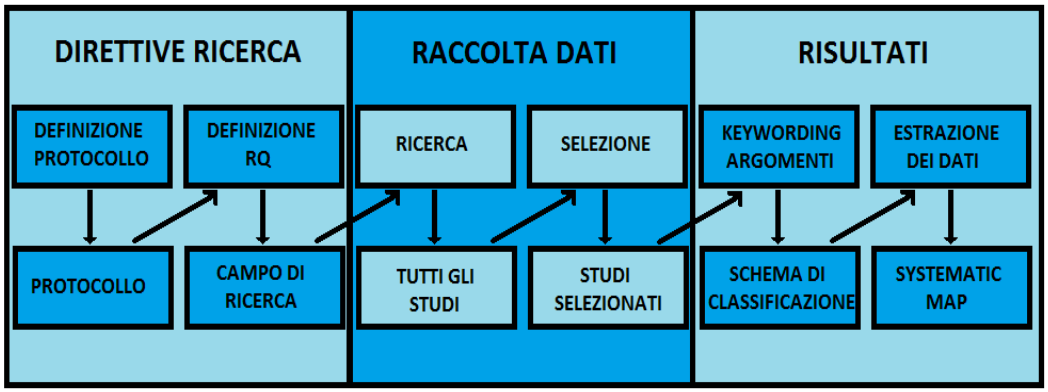

Figura 1.1: Descrizione delle fasi del SMS

1.3 Fase I: Direttive di ricerca

1.3.1 Definizione del protocollo:

Viene definita in maniera chiara la successione ed il contenuto delle fasi in cui la ricerca si articola. Il protocollo può essere soggetto a ridefinizioni successive in base al'apporto di ulteriori informazioni, utilizzando un approccio di natura incrementale.

1.3.2 Definizione delle Research Questions

Sono il fulcro del Systematic Mapping Study, in quanto costituiscono il primo contributo per selezionare gli studi d'interesse nella seconda fase.

Sono strutturate in base a tre parametri:

- **Population:** complesso degli studi scientifici riguardanti l'ambito in questione
- **Intervention:** studi empirici in materia

- **Outcome:** tipologia e quantità di risultati empirici relativi agli studi

Dopo aver specificato le Research Questions, viene definito l'ambito della ricerca, che conterrà gli studi da valutare.

1.4 Fase II: Raccolta dati

1.4.1 Strategia di ricerca

E' necessario identificare la base su cui ricercare i dati ed è cruciale definire correttamente la stringa di ricerca, in modo che elimini gli studi non significativi e mantenga soltanto quelli che risultino essere effettivamente necessari. Per raggiungere tale scopo le stringhe possono diventare molto complesse ed impiegare connettori logici come AND ed OR.

In alcuni casi è possibile accompagnare la ricerca automatica con la navigazione manuale, al fine di garantire una maggiore completezza nella mappatura dell'ambito di ricerca.

I motori di ricerca generalmente più utilizzati sono librerie digitali preposte allo scopo, come IEEE Xplore o ScienceDirect.

La ricerca deve essere supportata non solo dalla stringa, ma anche da opportuni parametri restrittivi atti ad eliminare i documenti non strettamente correlati con la disciplina a cui appartiene l'argomento, quelli troppo vecchi e soprattutto bisogna eliminare le ridondanze causate da documenti che contengono le stesse informazioni.

1.4.2 Selezione degli studi

Questa fase si basa sull'utilizzo di determinati criteri di inclusione ed esclusione, oltre che all'impiego di opportuni filtri che riducono progressivamente il numero di risultati, portando ad una progressiva focalizzazione sull'area di interesse.

Un esempio dell'impiego dei criteri di inclusione ed esclusione è il seguente [6]:

Inclusione: libri, documenti, resoconti tecnici e "letteratura grigia" (cioè documenti che non vengono diffusi attraverso canali commerciali normali) che descrivono studi empirici circa la progettazione di software orientato agli oggetti. Dei documenti appartenenti allo stesso studio viene selezionato soltanto quello più recente e se un singolo documento tratta più studi, questi vengono trattati individualmente.

Esclusione: Gli studi che non riportano conclusioni empiriche o riportano documentazione disponibile soltanto in forma di estratti o presentazioni PowerPoint.

Viene consigliato inoltre l'impiego di più persone per migliorare l'attendibilità relativa alla decisione di inclusione\esclusione di un determinato studio. Riveste particolare rilievo l'individuazione dei criteri di qualità, che servono per evitare di includere un numero di studi maggiore del necessario, ad esempio includendo studi primari che si basano su aspetti di altri studi primari che sono già stati inclusi in precedenza[2]. I criteri di qualità sono impiegati solitamente anche nelle Systematic Literature Review [9][10][11].

Un esempio di criterio di qualità è il criterio di uguaglianza: Lo scopo di tale criterio è di valutare il peso che hanno i singoli studi, cioè quanto questi sono importanti. Tale soluzione permette un miglioramento della comprensione dell'argomento trattato e influenza positivamente l'affidabilità dell'analisi[2].

Solitamente i criteri di qualità vengono espressi mediante una serie di sezioni, ciascuna delle quali è costituita da un certo numero di domande. Viene attribuito un punteggio

alle singole sezioni per ogni studio, determinando in quale misura risponde alle domande contenute nella sezione.

Da questo punto in poi è possibile osservare l'andamento del trend degli studi nel tempo, valutando in quale misura il numero di pubblicazioni relative al campo di interesse sia variato nel corso degli anni.

1.4.3 Keywording degli argomenti

Il keywording è una procedura atta a ridurre il tempo impiegato per la classificazione degli studi selezionati. La procedura è costituita da due fasi: la prima è la lettura del documento e l'identificazione delle parole chiave che servono per individuare il contesto in cui questo si pone, la seconda è il confronto fra le parole chiave ottenute, al fine di avere una comprensione generale relativa alla natura ed al contributo della ricerca. Lo scopo del keywording è quello di aiutare ad identificare un set di categorie con il quale classificare i documenti.

Spesso, per identificare le keywords, risulta necessaria la sola lettura dell'abstract, ma nel caso in cui questo sia troppo poco specifico per derivarle, allora si leggono l'introduzione o la conclusione. Il seguente schema descrive la procedura svolta nel Systematic Mapping:

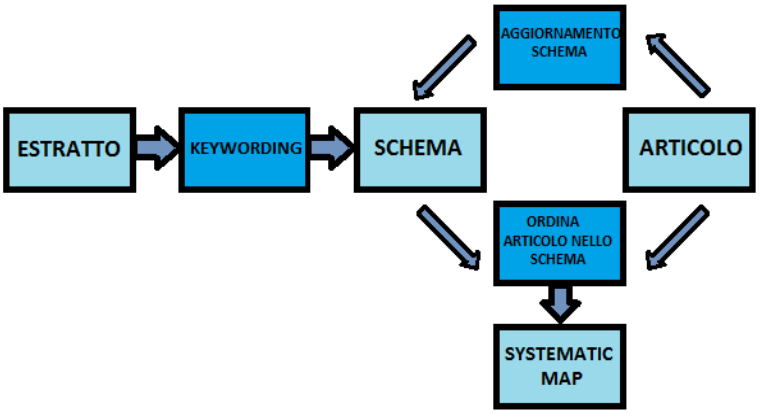

Figura 1.2: Fasi del keywording

1.4.4 Estrazione dei dati

Parte dalla definizione dei campi da estrarre, come il titolo, gli autori, le fonti o l'anno di pubblicazione. Vengono aggiunti allo schema i documenti presenti. E' possibile classificare a questo punto il numero di studi che appartiene ad una determinata classe o che risponde ad una certa RQ (anche a più di una). Lo schema di classificazione può variare durante il processo di elaborazione dei documenti, con l'aggiunta di un'ulteriore categoria oppure fondendo quelle già esistenti.

Uno degli schemi di classificazione più usato è quello di Wieringa et al. [7].

Tabella I: Tipi di ricerca

CATEGORIA	DESCRIZIONE
Validation Research	Le tecniche impiegate sono nuove e non sono mai state applicate nella pratica. Queste spaziano da esperimenti a lavoro di laboratorio.
Evalutation Research	Le tecniche vengono implementate e successivamente valutate. Sono descritti i vantaggi e gli svantaggi relativi all'implementazione pratica.
Solution Proposal	Viene proposta la soluzione per un problema, può essere innovativa oppure un approfondimento di una tecnica esistente. Viene descritto il vantaggio dell'impiego della soluzione con argomentazioni o esempi pratici.

Philosophical Papers	Presentazione di un nuovo approccio nell'osservazione di fenomeni esistenti, strutturando il campo con una particolare tassonomia o con schemi concettuali.
Opinion Papers	Si espone un'opinione personale sul vantaggio o svantaggio dell'impiego di una tecnica, oppure osservazioni su come adottarla. Non si affida a lavori correlati o metodi di ricerca.
Experience Papers	Espone cosa e come fare in un certo caso pratico, deve attenersi all'esperienza personale dell'autore.

Lo schema di classificazione deve essere concepito in modo che non sia necessaria una lettura approfondita per collocare uno studio in una data categoria, così risulta possibile analizzare più documenti ma con uno sforzo di elaborazione limitato.

1.5 Fase III: Risultati

I risultati ottenuti vengono catalogati in base ad alcune facets (sfaccettature), quelle solitamente usate sono l'argomento su cui si basa lo studio ed il tipo di ricerca svolto da questo. La catalogazione secondo le sfaccettature avviene dopo aver applicato i filtri scelti. I risultati vengono predisposti in un opportuno grafico, generalmente questo può essere una tabella di frequenza oppure un diagramma a bolle.

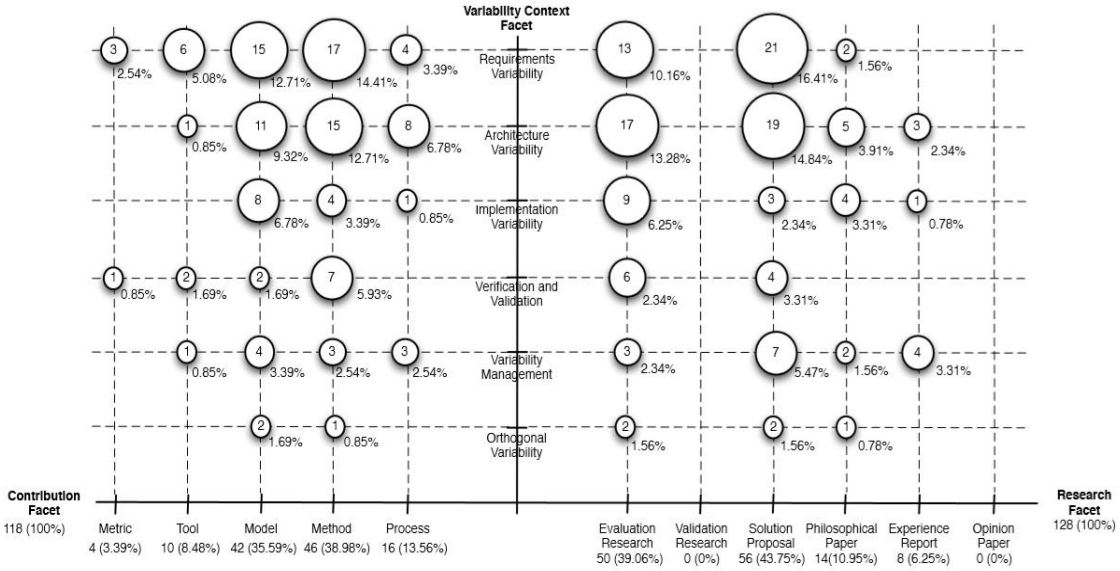

Figura 1.3: Esempio di diagramma a bolle

1.6 Linee guida per il Systematic Mapping Study

Seguono ulteriori direttive per effettuare un Systematic Mapping Study efficace[8]:

Utilizzo complementare dei metodi: I metodi di revisione sistematica SMS ed SLR possono entrare in parte in contraddizione a causa dei loro scopi differenti. Quello che bisogna fare è usare per primo il SMS, in modo da mappare l'area oggetto di studio, per poi eseguire una SLR sull'argomento su cui si sceglie di concentrarsi. Anche lo svolgimento autonomo di un SMS ha un suo valore: permette di capire in quali argomenti gli studi sono più o meno concentrati nell'area di interesse.

Classificazione dei documenti in base alle prove ed alla novità: Anche senza entrare in merito al metodo di ricerca impiegato negli studi, è possibile valutarli in base ad uno schema di classificazione di alto livello, il quale può basarsi in futuro anche sul livello delle prove presentate e sulla novità della ricerca. Uno schema di classificazione dovrebbe includere anche categorie non empiriche.

Profondità adattativa nella lettura degli studi: Si consiglia una maggiore estensione nell'impiego degli estratti negli studi relativi all'Ingegneria del Software in futuro. Tuttavia, nonostante gli estratti si rivelino molto utili per una rapida classificazione dei documenti, talvolta possono rivelarsi fuorvianti. Nel caso in cui l'estratto non sia sufficiente per pervenire ad un'adeguata categorizzazione, è sconsigliato predefinire in linea generale quali parti del documento leggere, ma piuttosto valutare per ogni singola istanza il livello di approfondimento di lettura da adottare.

Visualizzare i dati ottenuti: I dati estratti dallo studio possono venire esposti conteggiando la frequenza delle pubblicazioni, in modo da avere un'idea più generale in merito alla copertura dei settori. I dati possono venire presentati secondo vari tipi di grafici, in particolare quello a bolle, il quale si presta bene nell'aiutare a definire in quali ambiti del settore la ricerca è più importante.

CAPITOLO 2:

SOFTWARE ECOSYSTEMS

2.1 Definizione

Il concetto di ecosistema è parte integrante dell'ecologia ed è definito nella maniera seguente [12]:

Unità funzionale fondamentale in ecologia: è l'insieme degli organismi viventi e delle sostanze non viventi con le quali i primi stabiliscono uno scambio di materiali e di energia, in un'area delimitata.

Tale definizione tuttavia sembra avere poco a che fare con il Software. La definizione di ecosistema da cui si parte solitamente per formulare il concetto di ecosistema software è infatti quella di ecosistema umano [13]:

Un ecosistema umano è costituito da attori, dalle connessioni fra di loro, dalle attività che svolgono e dagli scambi che avvengono tramite le connessioni presenti, riguardanti fattori di natura fisica e non.

A sua volta questo concetto viene suddiviso in ecosistema umano commerciale o sociale:

Un ecosistema di natura commerciale gli attori sono entità finanziare, venditori e clienti, tale ecosistema include, fra le sue transazioni, anche quelle di natura finanziaria, ma anche scambi di conoscenza relativi alla piattaforma o ai suoi applicativi. Gli attori sono costituiti da beni e servizi.

Un ecosistema sociale ha invece come attori degli utenti, come connessioni quelle di natura sociale e come oggetto delle transazioni, lo scambio di conoscenza.

Questa definizione già lascia intuire la natura relazionale che costituisce l'ecosistema software, a sua volta definito come [14]:

Un ecosistema software è costituito dall'interazione fra un gruppo di attori, la quale avviene su di una certa piattaforma e porta all'implementazione di soluzioni e servizi.

E' possibile inquadrare l'ecosistema software nella tipologia di ecosistema commerciale, dove beni e servizi sono le soluzioni applicative. Lo scopo fondamentale per cui sono nati i Software Ecosystem è infatti proprio quello di accrescere il valore iniziale del prodotto. Solitamente il processo che porta alla generazione di un Ecosistema Software è costituito da un iniziale successo del prodotto, il quale si rivela particolarmente versatile. Data la quantità di soluzioni che il software diventa in grado di offrire, l'azienda sceglie di aprire agli sviluppatori indipendenti in quanto non riuscirebbe autonomamente a sostenere i costi di R&D relativi ad ogni singola soluzione. Sacrificando una parte degli introiti e l'esclusività della conoscenza di determinate caratteristiche del suo prodotto, l'azienda potrà mettere sul mercato un prodotto le cui potenzialità vengono pienamente sfruttate.

2.2 Tassonomia dei SECO

Nonostante gli ecosistemi software siano stati impiegati estensivamente soltanto negli ultimi anni, con l'espansione del Web 2.0, la definizione delle varie tipologie da cui è costituito risale agli anni novanta, in particolare la tassonomia si basa su due caratteristiche: ciò che costituisce il fulcro dell'ecosistema e la piattaforma sulla quale questo pone le sue basi.

Di seguito viene specificata la tassonomia, con relativo esempio per ciascuna combinazione.

Tabella II: Tassonomia dei SECO

Categoria/Piattaforma	**Desktop**	**Web**	**Mobile**
Sistema operativo	Linux	GoogleAppEngine	Android
Applicazione	Microsoft office	eBay	nessuna
Programmazione da parte dell'utente	Microsoft Excel	MIT App Inventor	nessuna

Come si nota, alcune combinazioni nell'ambito mobile non hanno ancora avuto diffusione.

Il livello di integrazione secondo il quale è possibile costruire un ecosistema software può essere più o meno profondo. Il livello più profondo di integrazione è costituito da un sistema operativo che può venire modificato in tutte le sue componenti, quello meno profondo da routine e codici che un programma, già costruito da terze parti, può eseguire. Sono esposti successivamente i vari livelli su cui può essere centrato un SECO.

2.3 SECO centrato sulla programmazione utente

Il fulcro su cui si basa è la capacità di fornire un linguaggio di programmazione specifico per il dominio in cui opera, il quale deve essere espressivamente potente ma al contempo facile da apprendere ed utilizzare. Se da un lato tale tipo di concentrazione su di un ambito specifico rende più controllabile questo tipo di ecosistema, in quanto ciò che l'utente può fare ricade esclusivamente nei limiti del programma, dall'altro proprio questo tipo di controllabilità limita il potenziale di sviluppo e la fascia di fruitori del software. Non è comunque raro che questo tipo di ecosistemi riesca comunque ad

espandersi ed a generare soluzioni per impieghi inizialmente non concepiti dagli sviluppatori.

E' di vitale importanza per la longevità di questo ecosistema, la capacità di condividere adeguatamente le soluzioni che vengono create dagli utilizzatori, mediante piattaforme di comunicazione adeguate.

Un esempio è Lego Mindstorms, una serie di prodotti della LEGO che combina i classici mattoncini con motorini, sensori ed altri dispositivi elettrici, fornendo all'utilizzatore la capacità di costruire dei sistemi automatizzati. Questo ecosistema è costituito da una nutrita comunità di sviluppatori indipendenti ed ha portato alla creazione di dispositivi sempre più complessi, come dei bracci robotici automatizzati.

2.4 SECO centrato su una applicazione

In questo caso il fenomeno relativo alla crescita a due stadi, cioè del successo dell'applicazione prima e della sua diffusione in un secondo momento, risulta ancora più accentuato. In questo caso però l'azienda deve riuscire a reinventarsi un'adeguata politica di gestione del prodotto e deve essere in grado di fornire degli strumenti (come tool ed editor adeguati) per assistere lo sviluppatore nel processo di modifica. Un rischio non indifferente consiste infatti nella perdita sul controllo del prodotto, nel caso in cui non sia il produttore ad aver fornito gli strumenti per l'edititing , ma una comunità di sviluppatori indipendenti o un'azienda concorrente. Un fattore chiave per il successo è in questo caso l'effettivo interesse da parte degli sviluppatori ad ampliare le funzionalità del programma. La compagnia produttrice dell'applicazione deve anche essere in grado di interfacciare i clienti con le estensioni dell'applicazione prodotte da terze parti, fornendo opportuni canali di pubblicizzazione e comunicazione.

2.5 SECO centrato sul sistema operativo

In questo caso è proprio il sistema operativo ad essere oggetto di modifiche da terze parti. Ne consegue che lo sviluppo di applicazioni che poggiano su quel particolare SO avviene in piena autonomia.

Dato che il sistema operativo necessita di una piattaforma hardware sulla quale operare, il successo e la longevità dell'ecosistema dipendono rispettivamente da quante unità compatibili con il sistema operativo in questione vengono vendute e dalla capacità di mantenere vivo l'interesse circa lo sviluppo e l'estensione del sistema operativo. E' pertanto necessaria una forte interazione con il produttore della piattaforma hardware. Inoltre, con quante più piattaforme fisiche il sistema operativo in questione sarà compatibile, tante più possibilità avrà di venire maggiormente diffuso.

Un esempio di successo in questo caso è celeberrimo sistema operativo Linux, il quale è compatibile con un lunghissimo elenco di dispositivi.

L'adozione di un sistema operativo rispetto ad un altro dipende dal successo e dalla diffusione delle sue applicazioni. Un sistema operativo deve in questo caso essere versatile ed in grado di permettere un agevole sviluppo di programmi applicativi da parte degli sviluppatori indipendenti, i quali a loro volta permetteranno una maggiore diffusione del SO in quanto necessario per quelle applicazioni. Per suscitare un maggiore interesse nella comunità degli sviluppatori, vengono spesso usati ambienti di sviluppo popolari, i quali vengono creati solitamente da consorzi di società, come Eclipse, la quale include molte delle più grandi società del settore, come Ericsson, HP, IBM ed Intel.

Un altro problema di non poco interesse è la scarsa dinamicità delle quote di mercato nell'ambito dei sistemi operativi: un nuovo OS farebbe una grandissima difficoltà ad

inserirsi fra altri colossi come MAC OS e Windows, i quali non solo hanno il predominio del mercato, ma anche le risorse per poter continuamente aggiornare ed estendere i propri sistemi operativi e mantenere di conseguenza la loro posizione attuale. La distribuzione delle quote fra Gennaio ed Aprile 2013 e quelle fra Gennaio ed Aprile del 2016 risultano molto simili. Nel 2013 MAC OS deteneva il 6,42% delle quote e nel 2016 il 6,05%, stesso discorso per Windows: nel 2013 il 91.53% e nel 2016 l'85,7% [15].

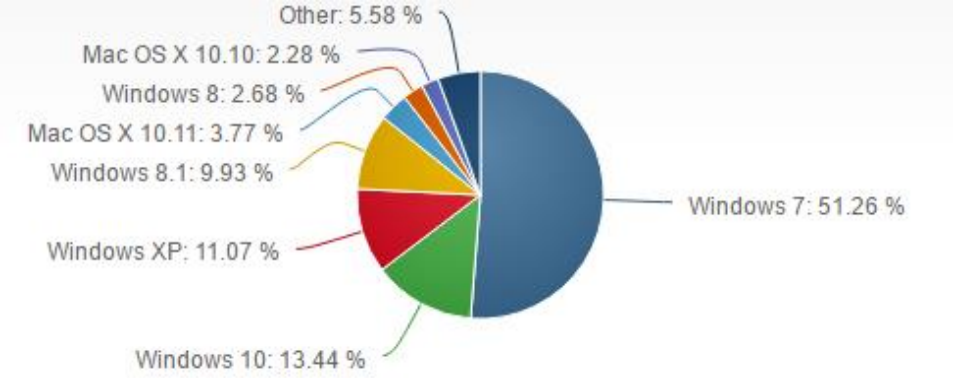

Figura 2.1: Quote di mercato nel 2016

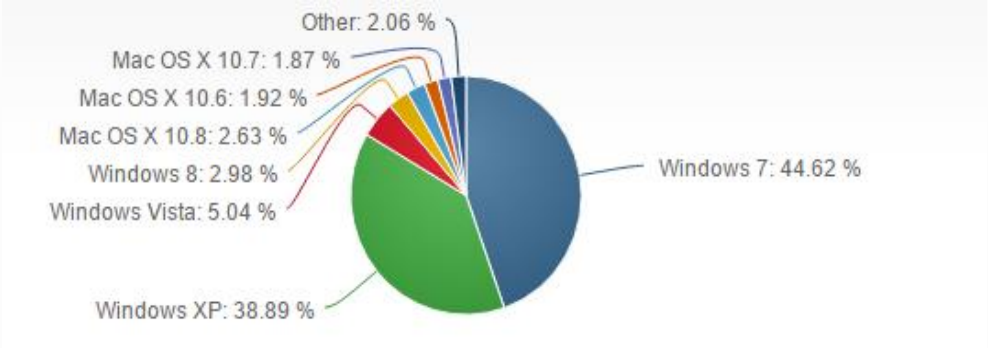

Figura 2.2: Quote di mercato nel 2013

2.6 Differenza fra Software Ecosystem e Software Open Source

Nel caso del software Open Source, quello che accade è che viene ceduta la proprietà intellettuale del codice, ciò significa che tutti oppure alcuni aspetti del software diventano accessibili a tutti (non è detto tuttavia che ciò avvenga gratuitamente).

Negli Ecosistemi software quello che viene fatto è aprire la piattaforma agli sviluppatori e dividere gli introiti con questi, tuttavia non viene condivisa alcuna proprietà intellettuale ed è anche possibile maturare diversi livelli di accesso in base al grado di confidenza con il produttore, infatti quello che spesso accade è che, al fine di mantenere il controllo su di una parte della produzione software, il produttore concede un livello di accesso maggiore ai propri dipendenti rispetto che agli sviluppatori di terze parti.

2.7 Un esempio di SECO: L'ecosistema Android

Il sistema operativo Android è concepito per funzionare su dispositivi mobile, si basa su Kernel Linux ma a differenza di questo ha alleggerito, proprio per essere performante su dispositivi portatili, gran parte del suo contenuto.

La diffusione di tale sistema operativo è un dato di fatto: Android detiene l'82% delle quote di mercato nei dispositivi mobile [16]. Parte del successo di questa piattaforma e del vertiginoso ritorno economico sono dovuti proprio alla lungimiranza mostrata nella creazione e nella cura di un ecosistema software che inquadri il sistema operativo.

La struttura dell'ecosistema può essere riassunta nel modo seguente:

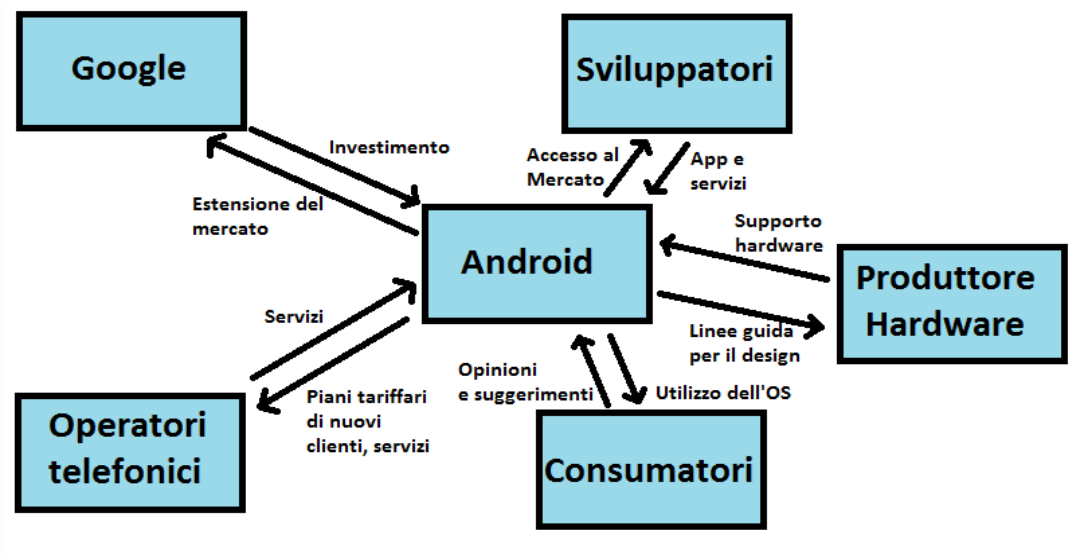

Figura 2.3: Ecosistema Android

Ogni elemento dell'ecosistema trae beneficio dall'interazione con Android, infatti:

- Google ha la possibilità di raggiungere, grazie ai dispositivi mobile, una fetta di mercato significativamente più grande.

- Gli operatori telefonici possono ottenere nuovi clienti ed usare il sistema operativo come supporto per agevolare il proprio lavoro relativo alla comunicazione fra dispositivi.
- Gli sviluppatori possono usare la piattaforma e gli annessi ambienti di sviluppo (come Android Studio) per accedere ad un vastissimo mercato.

- Il produttore hardware trae beneficio dal maggior volume delle vendite che viene a crearsi a seguito dell'aumento degli elementi coinvolti nell'ecosistema. Inoltre può avere un'idea specifica della domanda e decidere in maniera più efficiente come soddisfarla.

- L’utente finale ha bisogno del sistema operativo per sfruttare i servizi fornitigli dalle applicazioni e dall’operatore telefonico.

Oltre ai benefici tratti, i singoli attori apportano un contributo significativo, in particolare:

- Google investe nella piattaforma Android, portandovi capitali e personale qualificato.

- Gli operatori telefonici permettono, mediante i loro servizi, di mettere in comunicazione e rendere pienamente funzionali i dispositivi Android.

- Gli sviluppatori forniscono un gran numero di app le quali permettono svariati utilizzi del dispositivo, accrescendo la domanda.

- Il produttore rende possibile l’installazione di Android su di una piattaforma fisica.

- L’utente finale invia feedback (anche indirettamente, attraverso indagini di mercato) e consente di migliorare ed aggiornare continuamente le funzionalità del sistema operativo.

Uno degli aspetti fondamentali è che, in questo ecosistema, non solo i vari attori interagiscono, ma evolvono e si aggiornano come conseguenza delle loro interazioni, creando quindi anche una struttura in grado di resistere ad eventuali variazioni della domanda o alla nascita di domanda per ulteriori servizi.

2.8 Ritorno economico dei SECO

Oltre a creare, come già asserito, una struttura dinamica e longeva all'interno del mercato, i SECO aumentano il valore del prodotto.

In particolare ciò accade perché ogni attore dell'ecosistema cattura il valore del prodotto e lo incrementa apportandovi il suo contributo.

In particolare il ciclo su cui si basa tale aumento del valore in ecosistemi come Android ed iOS è il seguente: Un numero maggiore di app sullo store comporta un numero maggiore di utenti (perché il prodotto aumenta di valore, in quanto possiede ora maggiori potenzialità di utilizzo) ciò porta ad un aumento ulteriore della domanda ed all'impiego di nuovi sviluppatori. Essendo l'ultimo elemento citato sotto il controllo delle aziende, queste hanno infatti deciso di investire ingenti capitali per gli sviluppatori, anche esterni.

2.9 Implicazioni dei SECO nell'Ingegneria del Software

Nonostante la creazione di un ecosistema software si riveli, nella maggior parte dei casi, una scelta vincente, sussistono sicuramente dei problemi nel concepire un modello di processo adatto ad operare negli ecosistemi software, in quanto in questo caso l'azienda o le aziende che interagiscono con l'ecosistema, perdono il controllo su parte del ciclo di vita del software.

Le problematiche sono in realtà due: come riuscire ad effettuare la transizione dell'azienda da un modello produttivo autonomo all'interazione con terze parti nel processo della produzione del software e solo successivamente, come adottare un modello di processo orientato alla cooperazione.

La soluzione al primo problema risiede nel cambiamento dell'approccio che l'azienda deve avere riguardo al prodotto: da centralizzato deve divenire decentralizzato.

Per implementare tale strategia è necessario un approccio compositivo [13], ovvero considerare come distinte le singole parti del software.

Nonostante la decentralizzazione tuttavia, è ancora necessario lo scambio di informazioni fra le parti che si occupano dei singoli aspetti del software, le quali non seguiranno più una stessa guida designata dal processo software (come avviene nella singola azienda che produce l'intero prodotto), ma comunicheranno fra di loro attraverso l'architettura.

Un'altra modifica nell'approccio consiste nel passaggio da una strategia di tipo top-down, ad una di tipo bottom-up, dove ogni gruppo di lavoro indipendente si occupa delle sue componenti. Tuttavia risulta necessaria un'intensa comunicazione per accordarsi sull'interfaccia fra le componenti.

E' necessario inoltre concentrarsi sulla retrocompatibilità del software, altrimenti ciò porterebbe a penalizzare il mercato, a detrimento di tutti gli attori presenti.

Se da un lato l'approccio diventa più complesso, semplifica però di molto la gestione delle versioni, ma è comunque necessaria una comunicazione fra gli attori riguardante le innovazioni di ogni release.

Sempre analizzando Android, lo store Google Play presenta la seguente situazione per le versioni utilizzate [18].

Version	Codename	API	Distribution
2.2	Froyo	8	0.1%
2.3.3 - 2.3.7	Gingerbread	10	2.7%
4.0.3 - 4.0.4	Ice Cream Sandwich	15	2.5%
4.1.x	Jelly Bean	16	8.8%
4.2.x		17	11.7%
4.3		18	3.4%
4.4	KitKat	19	35.5%
5.0	Lollipop	21	17.0%
5.1		22	17.1%
6.0	Marshmallow	23	1.2%

Data collected during a 7-day period ending on February 1, 2016.
Any versions with less than 0.1% distribution are not shown.

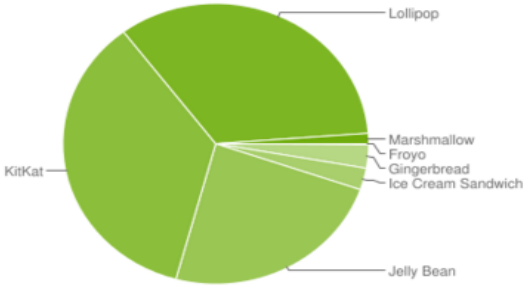

Figura 2.4: Quote di mercato dello store Google Play per versione Android

La versione Marshmallow 6.0 comprende una quota minima del mercato, l'1,2%. In assenza di retrocompatibilità l'attuale volume di vendite non sarebbe lontanamente raggiungibile. Tuttavia è necessario concentrarsi, per fornire il maggior numero di nuove funzionalità, sulle versioni più recenti, tralasciando quelle più vecchie, come quelle prima delle 4.1.x, che occupano a loro volta solo una piccola parte del mercato, ovvero il 5.3%.

L'ecosistema software porta con se spesso un apparente paradosso: nonostante l'agevolezza per lo sviluppo del sistema aumenti, grazie alla delega a terzi di parte del lavoro, si può verificare che le nuove versioni vengano pubblicate con difficoltà crescente rispetto a quelle più vecchie. Se non viene gestito con attenzione il processo di aggiornamento delle componenti software, la frequenza delle release rischia di

divenire sempre minore nel corso del tempo. Questo fenomeno è causato dalla crescente complessità delle relazioni intercorrenti fra i vari elementi software dell'ecosistema, in quanto sviluppano delle dipendenze reciproche che non possono più essere controllate dal singolo produttore software.

Una soluzione a questo problema consiste nel lasciare che i produttori delle varie componenti del software che costituiscono l'ecosistema, sviluppino indipendentemente nuove versioni, l'unico vincolo corrisponde al già citato rispetto della retrocompatibilità.

Un'ulteriore questione da risolvere dopo la transizione da produttore autonomo ad attore di un ecosistema è la gestione della proprietà del prodotto: l'azienda deve adattarsi alla perdita del possesso integrale del prodotto e trovare accordi appetibili rispetto a quelli che la concorrenza è in grado di offrire, al fine di catturare l'attenzione degli sviluppatori indipendenti.

2.10 Approccio Compositivo

Come già affermato precedentemente, è di vitale importanza ripensare il tradizionale approccio del processo software per una corretta pianificazione in un contesto SECO, il quale non implica che l'azienda produttrice possa controllare tutti gli aspetti della produzione.

Una soluzione presentata è l'approccio compositivo [19], il quale punta all'abbandono di una strategia di sviluppo guidata dal processo e si concentra invece sull'adozione di vincoli e regole definiti dall'architettura.

In tale approccio è importante anche il raggiungimento di un alto livello di disaccoppiamento fra le componenti del sistema, in modo da lasciare libere il più

possibile le terze parti di operare in maniera indipendente, ma rispettando sempre quello che l'architettura richiede.

Le caratteristiche fondamentali dell'approccio compositivo sono le seguenti:

- I **clienti assemblano i loro prodotti** selezionando dalle funzionalità rese loro disponibili.

- Le componenti soddisfano i principi di **sviluppo indipendente**, ma può essere necessario un **accordo sull'interfaccia** adottata fra le componenti.

- I **team di lavoro sono piccoli** (si arriva al singolo sviluppatore indipendente) e gestiti in
  modo autonomo.

- I **gruppi di lavoro** possono essere **esterni o interni** all'organizzazione.

- L'architettura del sistema è pensata per **agevolare la composizione** dei suoi elementi.

- I **rilasci possono avvenire autonomamente** e con cadenza arbitraria.

CAPITOLO 3:

APPLICAZIONE DEL SMS

3.1 Research Questions

Al fine di poter esporre in maniera approfondita gli ecosistemi software, sono state fissate delle domande a cui lo studio provvederà a rispondere:

- **RQ1: quali sono le principali caratteristiche dei SECO?**
- **RQ2: quali sono le entità e gli attori principali operanti all'interno di un ecosistema software?**
- **RQ3: quali sono gli aspetti architetturali relativi ai SECO?**
- **RQ4: seguendo la letteratura classica, quali sono i problemi principali e le relative soluzioni riguardanti i SECO?**
- **RQ5: Quali sono le principali aree di interesse ed applicative dei SECO?**

Scopo dello studio è anche quello di determinare le caratteristiche generali delle pubblicazioni, quali variazione del numero nel corso del tempo e determinazione degli argomenti e delle tipologie più ricorrenti negli studi.

Allo scopo di estrarre gli articoli, sono state consultate tre librerie digitali:

- **IEEE Xplore**: la libreria di carattere didattico della IEEE che contiene articoli provenienti da diversi ambiti dell'ingegneria.

- **ACM Digital library**: base di dati della Association for Computing Machinery, contiene tutti gli articoli e riviste pubblicati dall'associazione.

- **ScienceDirect**: base di dati contenente 26000 libri e 2500 riviste riguardanti varie discipline scientifiche.

3.2 Modalità di ricerca

Le librerie digitali sono state interrogate adoperando la seguente stringa di ricerca:

("SECO") OR ("software ecosystems") OR ("software ecosystem")

OR ("software eco systems") OR ("software eco system")

I risultati sono stati esportati in formato .bib (bibTeX) per poi poter essere elaborati ulteriormente.

Le attività relative al mapping sistematico sono state svolte fra Giugno ed Agosto 2016.

3.3 Ricerca su IEEE Xplore

Il totale degli articoli trovati con la stringa di ricerca è di 309 su 3981551 articoli disponibili, così distribuiti nel corso degli anni:

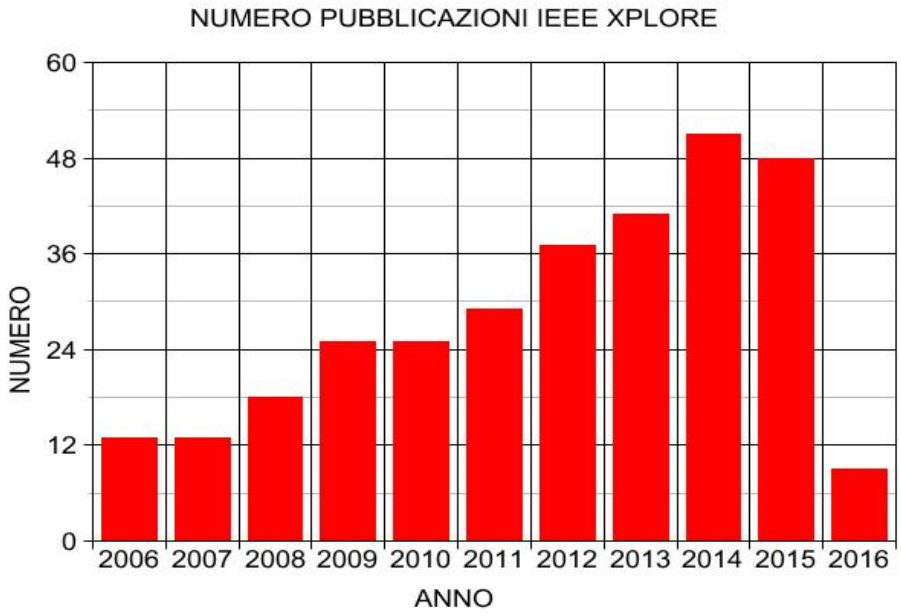

Figura 3.1 Distribuzione annuale delle pubblicazioni IEEE XPLORE

In questo caso è possibile notare, con le dovute approssimazioni, un andamento di natura esponenziale nel numero di pubblicazioni riguardanti gli ecosistemi software nel corso degli anni.

Questo dato testimonia un interesse crescente nei confronti di questo argomento, in parte sicuramente dovuto alla capillare diffusione degli ecosistemi software ed al grande ritorno economico che questi possono generare, come già visto in precedenza.

3.4 Ricerca su ACM Digital Library

Su 2564916 articoli presenti nella libreria, la ricerca ne ha selezionati 622, su base annuale la situazione riscontrata è stata la seguente:

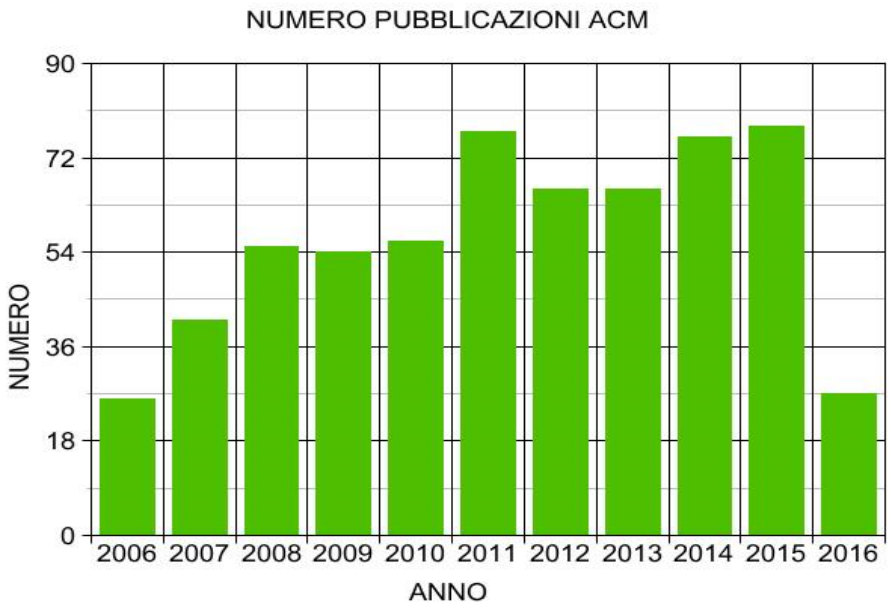

Figura 3.2 Distribuzione annuale delle pubblicazioni ACM

3.5 Ricerca su ScienceDirect

Gli articoli trovati che trattano degli ecosistemi software sono 339, in particolare:

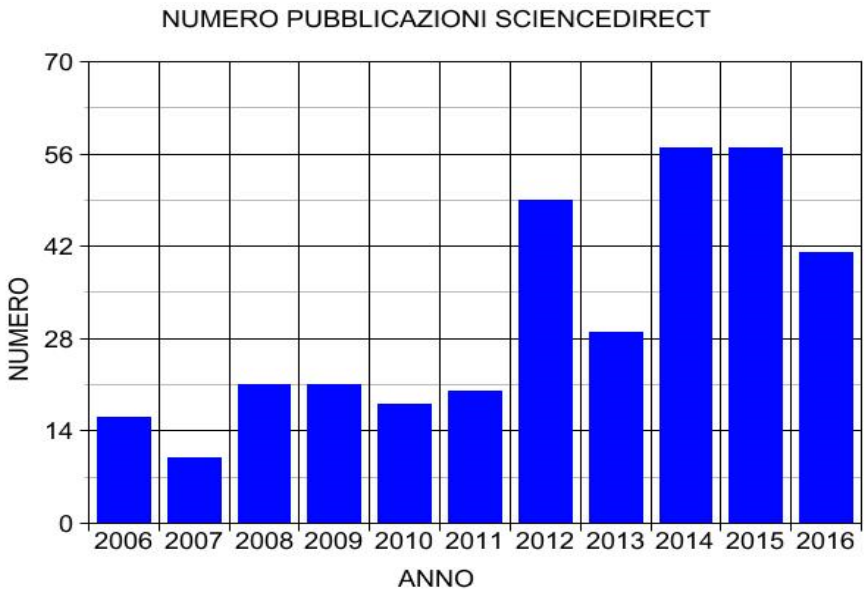

Figura 3.3 Distribuzione annuale delle pubblicazioni ScienceDirect

3.6 Elaborazione dei dati

I documenti .bib esportati dalle basi di dati contengono i seguenti campi:

- Titolo
- Autore/i
- Informazioni sulla fonte (titolo del libro, tema della conferenza, titolo della rivista…)
- Anno di pubblicazione
- Parole chiave
- Estratto

Dai file .bib esportati dalle librerie, per mezzo di un semplice programma, è stato possibile determinare quali sono le maggiori riviste che contengono le pubblicazioni sull'argomento:

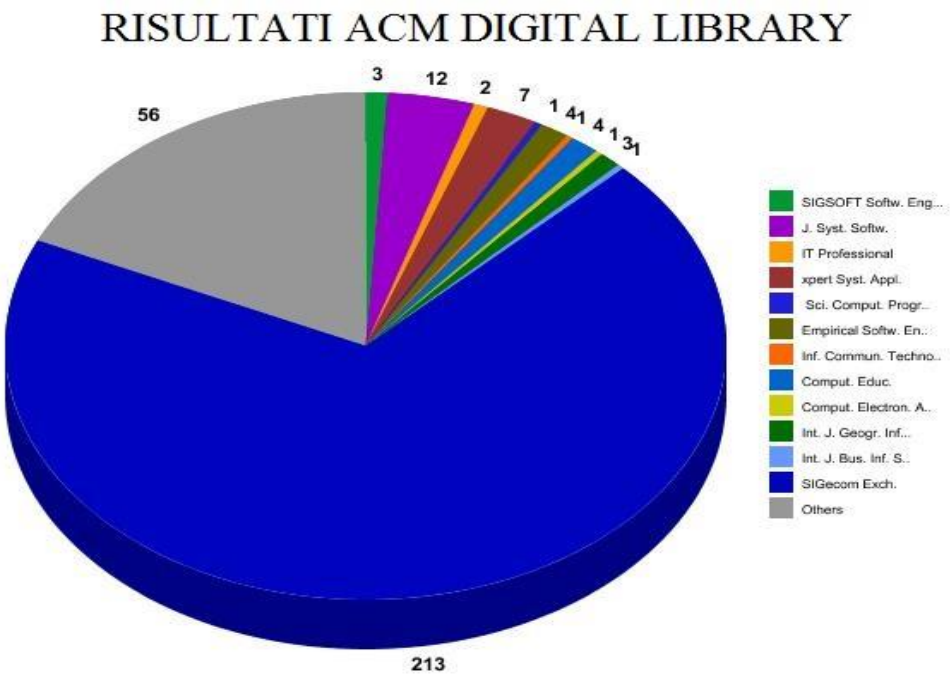

Figura 3.4

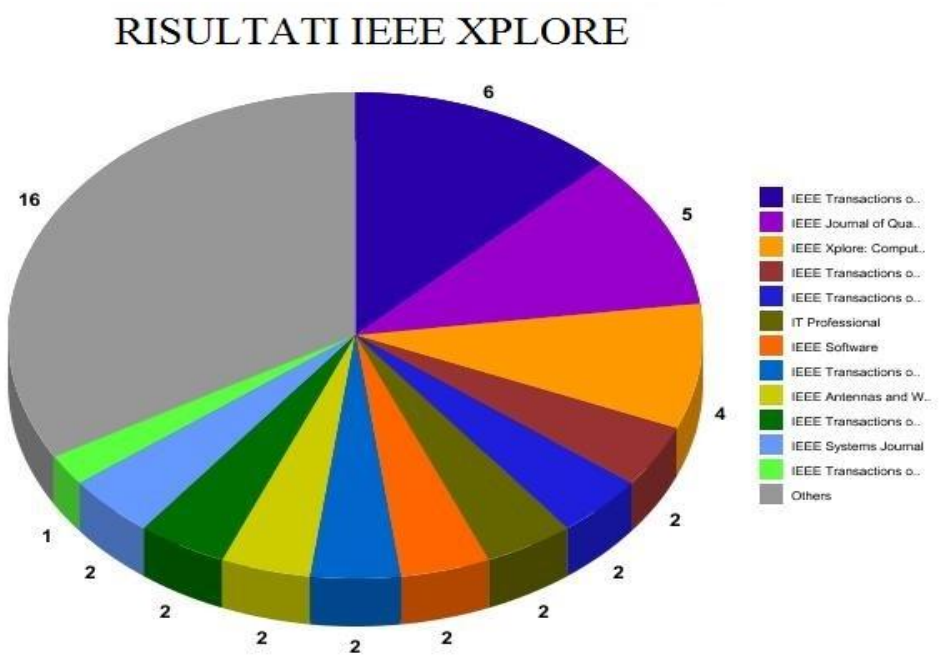

Figura 3.5

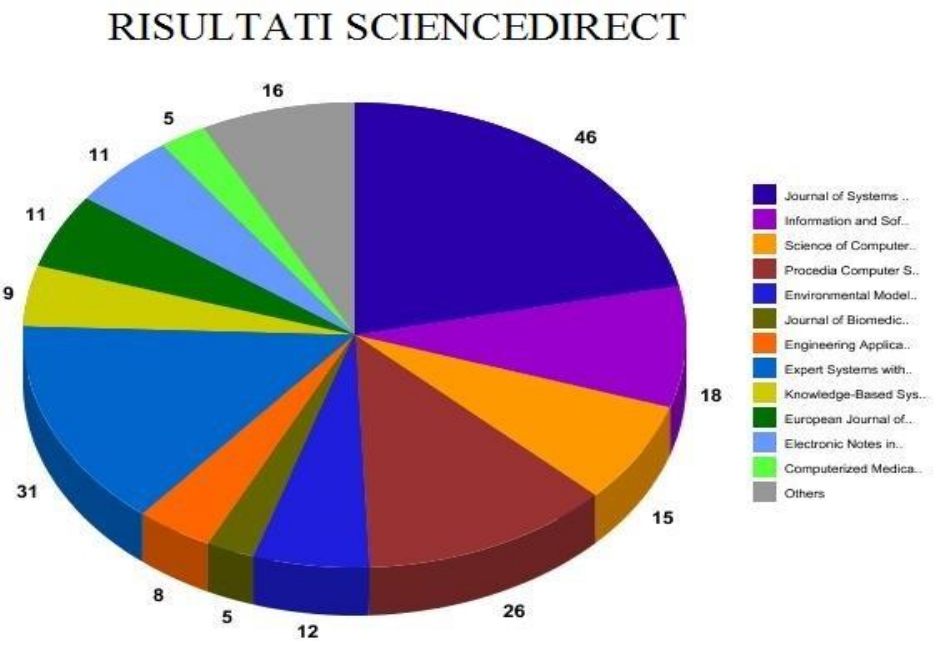

Figura 3.6

3.7 Parole chiave

Le keywords sono state estratte per mezzo di un programma applicativo in grado di individuarle e raggrupparle tenendo conto della distanza di Hamming. Tale operazione è risultata necessaria al fine di evitare la classificazione come keywords distinte di parole come “ecosystem” ed “ecosystems”, successivamente si è proceduto ad individuare le top 5 e 15 keywords.

Al fine di calcolare la distanza di Hamming intercorrente fra due stringhe è stato utilizzato il seguente metodo statico:

```
static int CompareStrings(String A, String B) {
     int minlength = 0, diff = 0;
     if (A.length() < B.length()) {
          minlength = A.length();
          diff = B.length() - minlength;
     } else {
```

```
                minlength = B.length();
                diff = A.length() - minlength;
            }
            for (int i = 0; i < minlength; i++) {
                if (!A.substring(i, i +
1).equalsIgnoreCase(B.substring(i, i + 1))) {
                    diff++;
                }
            }
            return diff;
        }
```

E' stata scelta una distanza di Hamming massima di 1 per il raggruppamento sotto la stessa parola chiave. Si è però notato che con una distanza pari a 2 i risultati erano estremamente simili, subendo però distorsioni per i termini più brevi, come negli acronimi.

L'analisi delle parole chiave si rivelerà utile per determinare quali sono gli argomenti correlati agli Ecosistemi Software, per poterli approfondire singolarmente.

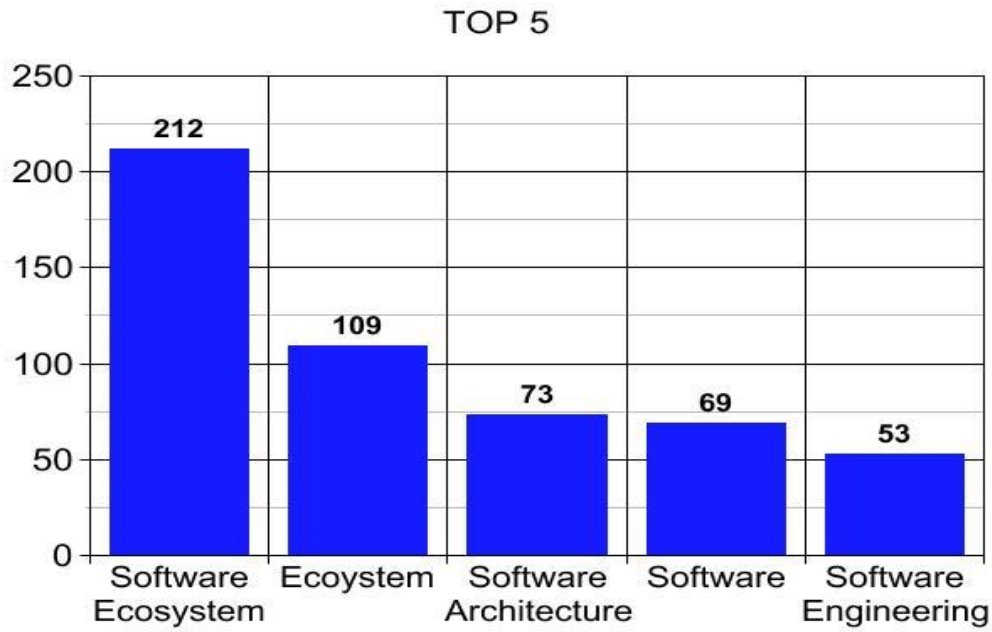

Figura 3.7

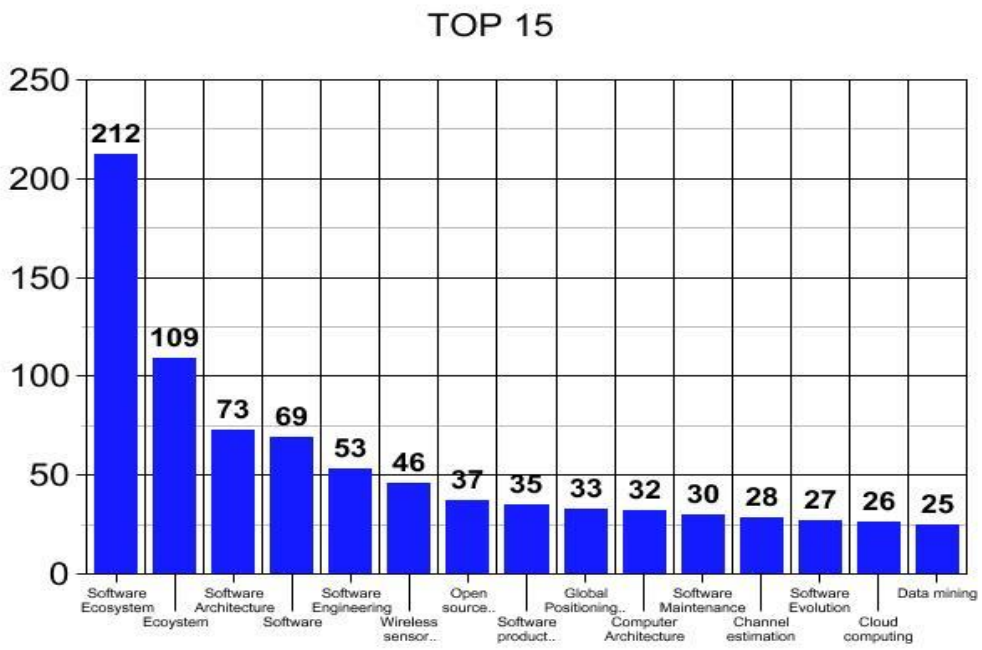

Figura 3.8

In seguito, le parole chiave sono state impiegate al fine di individuare quali sono gli argomenti più ricorrenti in base alla libreria ed all'anno di pubblicazione:

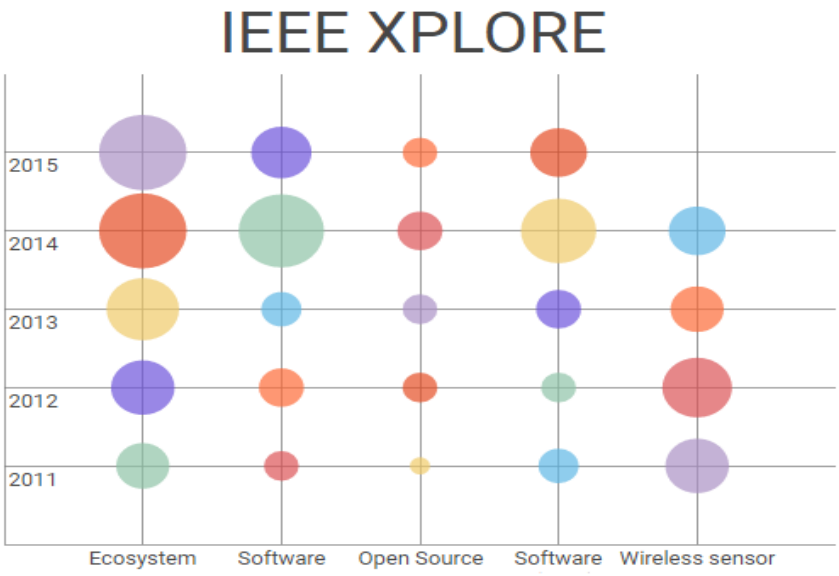

Figura 3.9

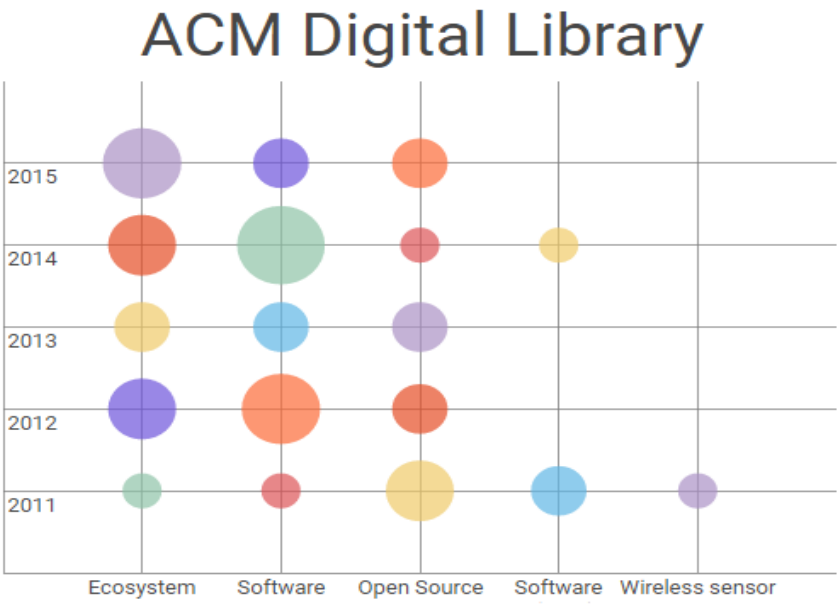

Figura 3.10

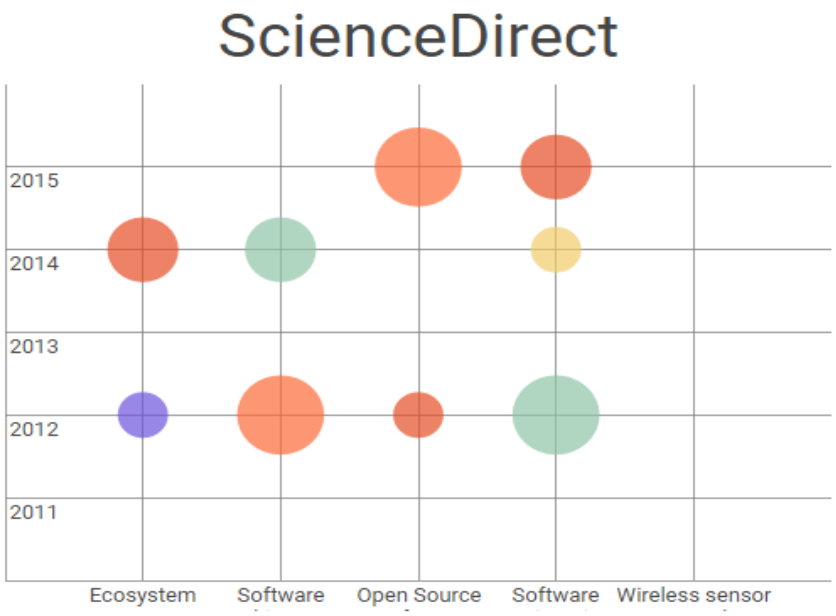

Figura 3.11

CAPITOLO 4:

ANALISI ED APPLICAZIONI DEI SECO

Dopo le interrogazioni effettuate sulle basi di dati, si è proceduto a scremare gli articoli che non parlavano in maniera specifica ed approfondita degli ecosistemi software.

Degli articoli trovati, appartenenti al quinquennio 2011-2015, ne sono stati selezionati 20, i quali sono stati letti in maniera approfondita al fine di poter determinare quali sono le caratteristiche ed i benefici degli ecosistemi software.

4.3 Elenco dei documenti oggetto di studio

Tabella III: Documenti oggetto di studio

Num	Documento
1	P. D. Saini and A. A. Aqrawi , Critical Success Factors of exploration and production software ecosystems, 2015.
2	L. Yu, The Market-Driven Software Ecosystem, 2013.
3	D. E. P. Meiru Che, Architectural Design Decisions in Open Software Development: A Transition to Software Ecosystems, Texas, 2014.
4	M. B.-F. P. C. S. M. M. R. Simon Urli, Managing a Software Ecosystem Using a Multiple Software Product Line: a Case Study on Digital Signage Systems, 2014.
5	Romain Robbes and Mircea Lungu. A study of ripple effects in software ecosystems . In Proceedings of the 33rd International Conference on Software Engineering (ICSE 2011), 2011.
6	D. Lettner, F. Angerer, H. Prähofer e . P. Grünbacher, A Case Study on

	Software Ecosystem Characteristics in Industrial Automation Software, Nanjing, 2014.
7	Hess, Steffen and Naab, Matthias and Trapp, Marcus and Magin, Dominik and Braun, Susanne, The importance of Mobile Software Ecosystems in Smart Rural Areas, 2015.
8	Mahadev Satyanarayanan, Rolf Schuster, Maria Ebling, Gerhard Fettweis, Hannu Flinck, Kaustubh Joshi, and Krishan Sabnani, An Open Ecosystem for Mobile-Cloud Convergence, 2015.
9	E. Knauss, D. Damian, A. Knauss e A. Borici, Openness and Requirements: Opportunities and Tradeoffs in Software Ecosystems, Karlskrona, 2014.
10	R. P. d. Santos e C. M. L. Werner, ReuseECOS: An Approach to Support Global Software Development through Software Ecosystems.
11	T. Ruokolainen e L. Kutvonen, Anarchitecture framework for facilitating sustainability of open service ecosystems, 2012.
12	Ejub Kajan, Ljubomir Laziü, and Zakaria Maamar , Software Engineering Framework for Digital Service-oriented EcoSystem, 2011
13	K.-B. Schultis, C. Elsner e D. Lohmann, Moving Towards Industrial Software Ecosystems: Are Our Software Architectures Fit for the Future?, San Francisco, 2013.
14	P. Pelliccione, Open Architectures and Software Evolution: the case of Software Ecosystems, 2014.
15	E. Handoyo, S. Jansen e S. Brinkkemper, Software Ecosystem Modeling: The Value Chains, Neumünster Abbey, 2013.
16	Heuijin Lee, Sungwon Kang, Myungchul Kim, An Efficient Application-Device Matching Method for the Mobile Software Ecosystem, 2014.
17	R. Santos, C. Werner, O. Barbosa e C. Alves, Software Ecosystem: Trends and Impacts on Software Engineering, 2012.

18	Katja Andresen, Carsten Brockmann, Christina Dräger, A Classification of Ecosystems of Enterprise System Providers – An Empirical Analysis, 2013.
19	Matthias Tichy, Jan Bosch, Michael Goedicke, and Brian Fitzgerald, 2nd International workshop on rapid continuous Software Engineering, Florence, Italy, 2015.
20	J. Bosch, Software Ecosystems: Implications for Strategy, Business Model and Architecture, 2011.

4.1 Caratteristiche principali degli ecosistemi software

Tabella IV: Caratteristiche SECO

Caratteristica	Articoli
Collaborazione tra enti dell'ecosistema	1, 2, 8, 12, 15, 18
Analisi e progettazione attraverso diversi strumenti	3, 4, 6, 17, 18
Strategie di lavoro	1, 4, 8, 9, 12, 15, 20
Stabilità delle interfacce	7
Sviluppo di software aperto	3, 9, 16, 17
Integrazione di nuove funzionalità	1, 8, 16, 17, 20
Sicurezza ed affidabilità	7, 8, 17, 18
Scalabilità del personale e delle componenti	11, 12, 14, 19
Partecipazione degli attori	8, 13, 18
Mercati e canali di distribuzione	1, 2, 5, 17
Numero di utenti e relative conseguenze	7, 14, 17, 20

Relazione tra popolarità e vantaggi economici, strategici e tecnici	2, 4, 5, 8
Legame intercorrente tra software e prospettiva di business	2, 8, 10, 13, 20
Riuso del software come approccio principale	13, 14
Gestione ed evoluzione di strumenti per la piattaforma	13, 16, 17
Importanza degli sviluppatori esterni	1, 4, 14, 15
Legame con l' innovazione dei processi	1, 4, 11, 13, 15, 19
Impatto sulle piccole e medie imprese	9, 12, 14, 18
Evoluzione e interdipendenza con altri sistemi (servizi, applicazioni, ecc.)	2, 5, 9, 16, 18

4.2 Benefici degli ecosistemi software

Tabella V: Benefici SECO

Benefici	Articoli
Collaborazione all'interno delle comunità	2, 3, 8, 9, 10
Condivisione della conoscenza e cooperazione tra più entità indipendenti	4, 7, 17, 18
Anche sviluppatori poco esperti sono motivati a creare contenuti	13
Condivisione tra gli attori delle linee guida, delle strategie e dei modelli di business	1, 4, 8, 9, 10, 13, 20
Il successo del software aumenta l'attrattiva per i suoi attori	2, 4, 17

L'aumento della qualità del prodotto comporta un aumento della produttività e delle transazioni monetarie	4, 8, 10, 13, 20
Produzione di prodotti e servizi diversi per diversi utenti	8, 14, 16
Soddisfa le esigenze diversificate degli utenti	1, 6, 8, 14
La piattaforma guida l'acquisto da parte degli utenti	6, 14
Rapida evoluzione dei mercati	9, 11, 15, 19
L'evoluzione del software guida i processi decisionali	3, 11, 15, 17, 19
Flessibilità nella scelta della piattaforma da utilizzare	3
Riduzione dei costi per lo sviluppo del software e per la sua distribuzione	3, 10, 12, 20
Facile espansione mediante l'utilizzo di API	9, 10, 14

4.4 Principali entità ed attori di un ecosistema software

Gli attori di un ecosistema software possono assumere diversi ruoli, si distinguono:

- **Orchestrator:**

  Si tratta di colui che gestisce le interazioni di tutto il sistema, nell'esempio precedente riguardante l'ecosistema Android è la Google, che ha investito nella creazione e nello sviluppo della piattaforma e che consente tramite lo store l'accesso al mercato da parte degli sviluppatori, talvolta compie anche indagini di mercato che servono da feedback.

- **Keystone:**

  Ovvero la chiave di volta, cioè l'attore su cui si basa l'intero sistema, è in questo caso lo sviluppatore della piattaforma hardware sulla quale viene installato il sistema operativo.

- **Niche players:**

  Operatori di nicchia, si tratta degli sviluppatori di app, i quali sono indipendenti e concorrono tra loro, guadagnando dalle vendite nello store.

L'altra componente è il software, il quale può essere o la piattaforma sulla quale poggiano i vari servizi o applicazioni o le applicazioni ed i servizi stessi.

Nel caso di Android questo tipo di struttura è abbastanza evidente: Il layer Application Framework permette di gestire le attività sulle quali si basano le applicazioni stesse attraverso vari manager, inoltre è garantita l'interoperabilità fra i servizi grazie al Binder ed agli altri meccanismi che permettono la IPC.

4.5 Aspetti architetturali

Definiamo l'architettura di un ecosistema software come l'insieme delle strutture necessarie per il suo funzionamento, cioè attori ed elementi software che ne fanno parte, includendo le relazioni tra di essi e le loro proprietà [20].

L'insieme di tutte le componenti software dell'ecosistema costituisce il cuore dell'ecosistema stesso, può essere analizzato per poter determinare la tipologia di ecosistema, possono anche essere considerate le variazioni di questo nel corso del tempo, determinando la reazione degli attori a queste.

Un ecosistema software può essere schematizzato secondo molti possibili criteri, tre possibili strutture identificabili sono:

4.5.1 Struttura organizzativa

Pone l'accento sui ruoli coperti dai singoli attori e sul diverso grado di necessità che questi hanno nel sistema, vengono inoltre mostrate le interazioni intercorrenti fra i diversi attori.

4.5.2 Struttura del Business

In questo caso viene analizzata la variazione del valore causata dall'intervento degli attori nell'ecosistema, i quali possono catturare tale valore, sfruttandolo per i propri scopi, ed anche accrescerlo ulteriormente.

Sebbene questo avvenga principalmente per scopi di guadagno e lo schema si presti bene all'analisi degli aspetti finanziari, viene utilizzato anche in contesti non orientati al profitto, in quanto il suo oggetto di studio è il valore del prodotto e le sue variazioni, non soltanto il profitto dei singoli attori.

La struttura del Business viene concepita attraverso la Business Model Ontology, la quale attraverso dei modelli di business rappresenta lo schema in questione.

Un modello di business si distingue in quattro aspetti fondamentali:

- **Prodotti**

  Vengono descritte le peculiarità di un nuovo prodotto o di una nuova innovazione attraverso una Value Proposition, ovvero la proposta da parte di un'azienda ad un gruppo di acquirenti [17].

- **Interfacce utente**

  Si tratta della descrizione del tipo di utenza (channel segment) e della relazione che si ha con questa, ovvero di come si arrivi all'acquirente.

- **Gestione di infrastruttura**

  Descrive il coinvolgimento nel mercato, in particolare dei ruoli chiave KR (Key Roles), dei partner commerciali KP (Key Partners) e le risorse chiave KR (Key Resources), le quali possono includere anche il personale qualificato.

- **Aspetto finanziario**

  Tratta i costi sostenuti per alimentare il Business (investimenti, stipendi …) ed i ricavi ottenuti.

4.5.3 Struttura del software

Questa struttura identifica come attori principali gli sviluppatori ed i vari elementi del software, il software può essere interpretato o come semplice sorgente che viene scritto, nel suo comportamento durante l'esecuzione del codice o nell'interazione con altro software.

Lo standard ISO/IEC 42010 prevede diverse viste architetturali, le quali permettono ai vari stakeholder di ottenere una maggiore comprensione dalla descrizione del software, in particolare:

- Le **Developement View**, le quali mostrano lo sviluppo del software, interessano principalmente l'azienda che lo sviluppa, la quale deve effettuare decisioni

critiche circa il processo software da adottare e le scelte architetturali da intraprendere.

- La **Functional View** mostra il comportamento del software durante la sua esecuzione ed interessa ad esempio all'acquirente che vuole una chiara visione del funzionamento del software.

- La **Deployment View** mostra l'integrazione del software con l'hardware ed interessa, soprattutto nel caso degli ecosistemi software, al costruttore della piattaforma fisica.

Il comportamento dei costruttori della piattaforma fisica e quello dei costruttori della piattaforma software tende ad influenzarsi vicendevolmente.

Un esempio a supporto dell'affermazione precedente risiede nell'analisi delle conseguenze relative all'aumento della potenza di calcolo dei dispositivi portatili, la quale ha portato all'adozione di nuove funzionalità software o al miglioramento delle performance di quelle già esistenti, al fine di sfruttare appieno l'aumento della capacità di calcolo. A sua volta tale aumento della potenza di calcolo è stato incentivato dal mercato, disposto a pagare un prezzo superiore al fine di ottenere piattaforme hardware più performanti.

4.6 Problematiche e soluzioni relative ai SECO

4.6.1 Cause principali

Le problematiche relative agli ecosistemi software derivano principalmente dalla suddivisione del lavoro fra i vari attori, la quale comporta inevitabilmente problemi per

la comunicazione immediata di modifiche. Una soluzione già esplorata è quella di comunicarle attraverso l'architettura, sia di natura hardware che software.

4.6.2 API ed estensioni

La propagazione di tali modifiche per attori e sviluppatori esterni può risultare talvolta problematica, come ha mostrato lo studio [21] per quello che concerne le API. In particolare si è sperimentato che "l'effetto di strappo" (ripple effect), può comparire anche a mesi di distanza e lasciare in stato inconsistente alcuni strati del sistema, i quali non propagheranno le modifiche per un certo periodo di tempo.

Al fine di determinare un maggior grado di autonomia da parte degli sviluppatori esterni, le case produttrici di software fanno un massiccio uso delle API, le quali permettono agli sviluppatori di avere degli strumenti da usare con una certa libertà per la creazione delle loro applicazioni. Tuttavia le modifiche di uno o più metodi delle API, specialmente se legate a funzionalità base del sistema, come quelle del kernel, possono avere ricadute negative sugli altri attori, che conseguentemente ridurranno la loro partecipazione all'attività dell'ecosistema. Un metodo semplice per evitare questi problemi consiste nel definire come "deprecati" quei metodi che sono sconsigliati agli sviluppatori in quanto, con tutta probabilità, non verranno inseriti nelle versioni successive.

4.6.3 Stabilità e compatibilità delle interfacce

Un altro problema fondamentale risiede nella stabilità delle interfacce, le quali mettono in comunicazione software prodotto da sviluppatori diversi. Le interfacce devono seguire determinati standard previamente concordati ed essere retrocompatibili.

E' infatti molto probabile che la compatibilità tra le interfacce possa costituire un grosso problema, a seguito dell'estensione del mercato globale e del completamento

dell'ingresso in questo di nazioni con apparati di produzione hardware e software meno avanzati.

4.6.4 Vulnerabilità

L'apertura è sicuramente un fattore positivo, persino vitale per un ecosistema, tuttavia questo significa una maggiore vulnerabilità a codice e programmi malevoli. In particolare diviene possibile che l'utente installi involontariamente un'applicazione maligna, ad esempio intenzionata a rubare i dati personali. Android notifica all'utente in fase di installazione i permessi che un'applicazione possiede una volta installata, aiutando a scongiurare tale eventualità. Inoltre i permessi sono strettamente integrati con la progettazione dell'app stessa, la quale non può compiere certe azioni se prima non ha specificato la richiesta delle rispettive autorizzazioni.

4.6.5 Aggiornamenti e riprogettazioni della piattaforma

Per mantenere vivo l'ecosistema, è necessario un continuo aggiornamento ed ampliamento dei servizi offerti. Le modifiche apportate dagli altri attori devono poter essere comunicate in forme e tempistiche adeguate agli sviluppatori esterni. Una situazione in cui questo non avviene, o avviene in maniera scarsamente efficiente, è certamente deleteria per l'ecosistema in quanto una volta forniti tali aggiornamenti, nessuno può usufruirne e ciò li renderebbe inutili e costosi.

In una piattaforma molto longeva può anche presentarsi il rischio che questa diventi troppo complessa a seguito delle modifiche, infatti queste possono essere dettate dalla variazione della domanda del mercato più che da una pianificazione iniziale. E' necessario mantenere "snella" la piattaforma, revisionando attentamente il codice dopo ogni cambiamento e lasciando solo quanto effettivamente necessario.

4.6.6 Privacy e trattamento dei dati personali

In un ecosistema software in cui la maggior parte dei servizi viene fornita gratuitamente, il valore fornito dall'utente non è più costituito da una transazione monetaria ma dai dati contenenti le sue preferenze, i quali vengono rivenduti a società che si occupano di indagini di marketing.

Se da un lato questo permette la creazione di un mercato più stabile, dove le entità che offrono i servizi hanno un quadro generale della domanda, dall'altro il trattamento adeguato dei dati personali costituisce un problema di natura sia etica che legale. Questo problema è di natura particolare in un ecosistema software, data la suddivisione delle responsabilità e dei ruoli in caso di danno.

4.6.7 Livello di integrazione degli sviluppatori esterni

Risulta necessario determinare il livello di accesso che i soggetti esterni possono avere alla piattaforma, in particolare sono possibili tre tipi di integrazione:

- **Accesso ai dati memorizzati** nella piattaforma da parte delle applicazioni esterne. Questo potrebbe però rendere il sistema vulnerabile ad attacchi.

- **Estensione delle funzionalità** della piattaforma per mezzo dell'integrazione nel flusso di lavoro, vengono cioè estesi i casi d'uso. Anche in questo caso, un'integrazione così profonda potrebbe lasciar entrare codice dannoso o malevolo.

- **Costruzione di un framework** per l'interfaccia utente, garantendo un isolamento rispetto al cuore della piattaforma, a cui si può solo accedere solo per mezzo dell'interfaccia.

4.7 Aree di interesse principali

Grazie all'analisi delle keywords è stato possibile determinare quattro aree di interesse fondamentale relative agli ecosistemi software, ovvero Global Positioning System, Wireless Sensor Network, Cloud computing e Data mining.

4.7.1 Global Positioning System

Il sistema di posizionamento globale comprendeva a maggio 2016 un numero di satelliti pienamente attivi pari a 31 [22] i quali permettono la localizzazione in un punto qualsiasi della superficie terrestre. Il funzionamento prevede l'interazione fra un ricevitore e la rete satellitare, la posizione viene identificata elaborando il segnale del ricevitore GPS, in particolare il suo tempo di propagazione.

Il sistema GPS è composto da tre segmenti: il segmento spaziale (i satelliti) il segmento di controllo (costituito dalle stazioni di controllo principali ed ausiliarie e dalle antenne per comunicare con i satelliti) ed il segmento utente (i ricevitori in dotazione agli utenti).

Attualmente la maggioranza degli smartphone integra un GPS, sono disponibili inoltre molte applicazioni che fanno uso di tale funzionalità, come quella ufficiale prodotta da TomTom. Pertanto si può osservare quanto altri ecosistemi come Android risultino strettamente collegati con il sistema GPS e che quest'ultimo possa essere definito a sua volta come un ecosistema costituito dai segmenti precedentemente descritti. Tuttavia l'ecosistema GPS è più chiuso di quello Android, in particolare il segmento di controllo è gestito dall'Aeronautica Militare degli Stati Uniti. Inoltre il sistema GPS è nato, a differenza di Android, con applicazioni militari, tutt'ora presenti.

E' possibile identificare come orchestrator il governo statunitense, il motivo risiede nel fatto che tale entità è l'investitore e regola l'interazione fra i vari attori del sistema. Un esempio di tale influenza risiede nel fenomeno della Service Availability [23], la quale

prevedeva addirittura l'inserimento intenzionale di errori di posizione nelle prime applicazioni civili del GPS, rendendolo molto meno efficiente della sua controparte militare.

Il ruolo di keystone era coperto dai produttori di GPS, in particolare TomTom. Oggi sono subentrati per questo ruolo i produttori di smartphone, inglobando nei loro dispositivi le funzionalità GPS.

L'integrazione con l'ecosistema Android ha permesso agli sviluppatori indipendenti di implementare funzionalità che impiegano il GPS nelle loro applicazioni. In tale modo è stato possibile introdurre nell'ecosistema combinato anche i niche players.

Android include le funzionalità GPS nell'hardware abstraction layer, tale strato ha l'importante ruolo di interfaccia fra le richieste di utilizzo hardware e le effettive chiamate a tali componenti fisiche, in questo modo risulta incrementata la compatibilità del software nelle diverse configurazioni hardware che gli smartphone possono avere.

L'uso del GPS è molto popolare in ambito Android grazie ad applicazioni di vario tipo, fra cui quelle per i social network, per le chat e per le mappe.

Tuttavia l'attivazione del GPS tende a scaricare molto rapidamente la batteria del dispositivo, in particolare lo studio [24] dimostra che tale assorbimento energetico risulta superiore laddove il segnale GPS presenta un SNR inferiore.

4.7.2 Wireless Sensor Network

Le reti wireless di sensori vengono utilizzate al fine di raccogliere grandi quantità di dati, tuttavia, l'etere è un mezzo particolarmente oggetto a disturbi ed interferenze, in particolare da segnali radio, i quali condividono la banda ISM impiegata dalle reti wireless [25]. Data la vulnerabilità ai disturbi, spesso i sensori wireless lavorano in reti

ad hoc. Tuttavia, in caso di un adeguato supporto fornito da strutture fisiche per le telecomunicazioni a grandi distanze, tali vincoli di prossimità possono essere rimossi.

Gli smartphone di ultima generazione dispongono di moltissimi sensori, fra cui si annoverano accelerometri, giroscopi, magnetometri, sensori di prossimità, di luminosità e di temperatura [26].

Dato che fanno un grande affidamento sulle infrastrutture wireless, gli smartphone sono ottimi candidati per creare una Wireless Sensor Network.

Anche nel caso della comunicazione wireless si è in presenza di una caratteristica fondamentale dell'ecosistema che è l'evoluzione dei singoli attori in base ai cambiamenti apportati dalle terze parti. In particolare è facilmente rilevabile la reazione del mercato alla transizione quasi completa da 2G a 3G e a quella tuttora in corso fra 3G e 4G. Tale cambiamento coinvolge tutti gli attori principali:

- Gli erogatori del servizio di telefonia mobile devono aggiornare le proprie infrastrutture sul territorio.
- I creatori del sistema operativo su cui poggiano le applicazioni devono implementare ed aggiornare interfacce adeguate con il comparto hardware.
- I produttori hardware devono garantire la compatibilità con i nuovi tipi di rete disponibili.
- Gli sviluppatori possono sfruttare l'incremento delle prestazioni per le loro applicazioni.

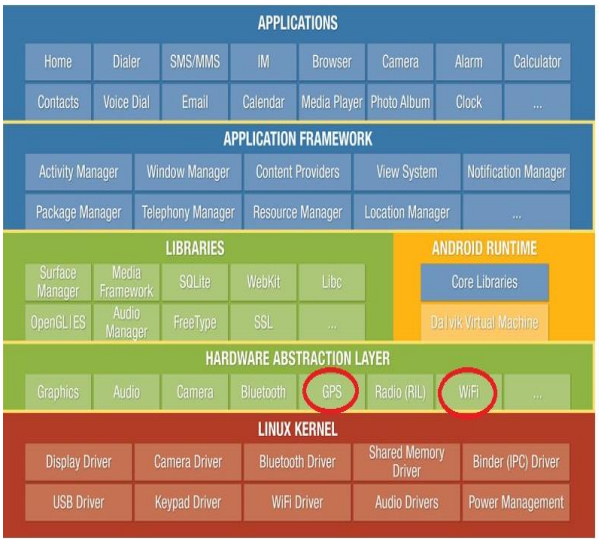

Figura 4.1: Struttura a layer di Android

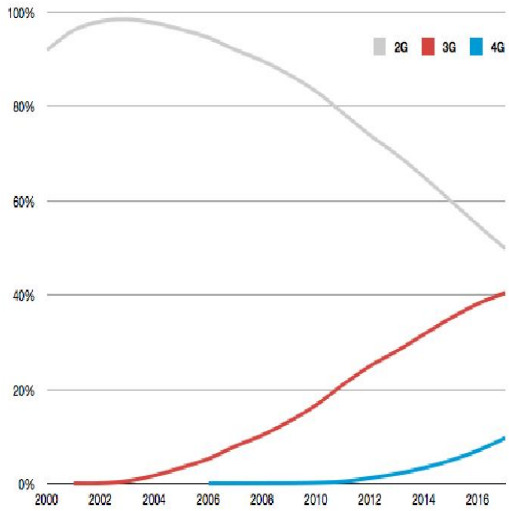

Figura 4.2: Evoluzione delle reti

4.7.3 Cloud computing

Il Cloud computing è definito come modello che permette un accesso conveniente ad un insieme di risorse computazionali, inoltre ha la peculiarità di permettere l'allocazione ed il rilascio delle risorse con costi di gestione minimi [27].

Molteplici sono le tipologie di servizi erogati per mezzo del Cloud:

- **Saas** (Software as a Service) in cui il servizio erogato consiste in uno o più programmi, i quali elaborano i dati che vengono inviati al server remoto sul quale operano.
- **Paas** (Platform as a Service) che è di livello superiore al precedente, prevede la possibilità di usare una piattaforma software, costituita da diverse componenti, come programmi o librerie.
- **Daas** (Data as a Service) mette a disposizione dati a cui accedere in remoto.
- **Haas** (Hardware as a Service) mette a disposizione risorse hardware per il calcolo computazionale.

- **Iaas** (Infrastructure as a Service) è quella tipologia di servizio che mette a disposizione sia risorse hardware sia virtuali, tali risorse vengono erogate su richiesta.
- **Caas** (Communication as a Service) permette la comunicazione in tempo reale.
- **Naas** (Network as a Service) serve per fornire all'utente dei servizi di connettività, come delle reti virtuali private (VPN).

Il Cloud rappresenta un ecosistema software sia individualmente, sia nell'interazione con altri ecosistemi.

L'orchestrator, che coincide spesso col gestore del Cloud, decide quanto l'ecosistema debba essere aperto e quali servizi erogare, determinando anche eventuali variazioni dell'area di interesse.

Nonostante il Cloud sia prevalentemente un'entità virtuale, è necessario che i produttori di dispositivi abilitino ed ottimizzino l'accesso ai servizi da questo erogati. I produttori di apparecchiature hardware in grado di beneficiare dei servizi offerti dal Cloud ricoprono il ruolo di keystone.

Partner esterni possono collaborare col gestore del Cloud al fine di provvedere all'erogazione dei propri servizi. Tale strategia avvantaggia entrambe le entità: il partner vedrà ampliato il suo raggio d'azione ed il gestore avrà un insieme maggiore di servizi da poter fornire all'utente.

Nell'integrazione con altri ecosistemi come Android e Windows, il Cloud può permettere il salvataggio di una grande quantità di informazioni relative agli utenti, come i file di log. E' stato determinato che il tasso di crescita delle informazioni relative al singolo utente aumenta ad un ritmo del 35% l'anno [28].

4.7.4 Data mining

Il Data mining è definito come il processo in grado di scoprire informazioni e relazioni utili a partire da un grande insieme di dati [29].

E' possibile definire quattro attori fondamentali nel processo di Data mining [30]:

- Il **Data provider** fornisce i dati desiderati
- Il **Data collector** raccoglie i dati
- Il **Data miner** applica gli algoritmi per l'estrazione dei pattern
- Il **Decision maker** compie delle scelte in base ai dati raccolti

All'interno di un ecosistema software, ogni attore deve conoscere in maniera precisa le necessità degli altri attori con i quali interagisce. La tecnica del Data mining permette di costruire il profilo della domanda, in maniera tale da rendere stabile l'ecosistema. Analisi periodiche e frequenti del mercato permettono di adattare l'offerta sulla base della domanda e costituiscono la base della capacità di reazione di un ecosistema software. Sono state già descritte precedentemente le problematiche che possono scaturire dal cattivo utilizzo dei dati raccolti. Nel corso degli anni sono state sviluppate delle contromisure software, come estensioni browser che evitano il tracciamento o bloccano gli annunci pubblicitari e tool per la codifica[30].

4.8 Applicazioni degli ecosistemi software

Gli ambiti di utilizzo per i SECO sono i più svariati e riguardano ambiti che possono spaziare dalla medicina alla viabilità stradale, seguono alcuni esempi:

4.8.1 Ecosistemi software per uso medico

La e-health indica l'insieme delle soluzioni informatiche destinate alla cura della salute dei pazienti, dalla comunicazione a distanza fra il medico ed il paziente all'archiviazione e condivisione online delle cartelle cliniche. Grazie all'ausilio informatico, diventa possibile migliorare l'accesso ad adeguate diagnosi e cure sanitarie da parte di un gran numero di pazienti. La World Health Organization ha già espresso il suo interesse a riguardo [31].

Risulta evidente che, al fine di gestire una tale mole di dati medici, superiore alla capacità di elaborazione del singolo individuo, sia necessario l'aiuto di una struttura software complessa, costituita da diversi attori, un ecosistema appunto. Il sistema Watson Health [32] è un esempio di tale applicazione. Questo progetto della IBM mira a creare un programma di intelligenza artificiale allo scopo di fornire aiuto ai medici nella formulazione di diagnosi per i pazienti.

Lo scenario finale prevede che il paziente venga monitorato attraverso dispositivi mobili, come lo smartphone, collegato ad appositi sensori. Lo smartphone, col consenso del paziente, invia i suoi dati nel cloud, i quali possono venire elaborati al fine di fornire una soluzione in tempo reale, con l'invio di messaggi sul telefono personale del paziente o contattando il suo medico curante. Inoltre la raccolta dei dati può aiutare la ricerca in molti settori. Il sistema è già stato adottato in diverse strutture ospedaliere statunitensi, specializzate in oncologia [32].

Figura 4.3

Un altro studio [33] propone un'ulteriore applicazione medica, che consiste nell'aiuto di pazienti con problemi di memoria. Attraverso la realtà aumentata sarà possibile migliorare la qualità della vita di persone che soffrono di questo tipo di disturbi. Il visore interagisce con altre apparecchiature, come lo smartphone, comunicando via internet con il Cloud, che contiene i dati del paziente, il Cloud a sua volta comunica con dei programmi che permettono di elaborare le informazioni in ingresso e fornirne di utili in uscita. Si viene a creare così un ecosistema software in grado di processare molte informazioni e fornire un valido aiuto nella vita quotidiana

4.8.2 Integrazione dei cloudlet

Lo studio [33] espone anche l'importanza dei cloudet, i quali sono presentati come intermediari fra i singoli dispositivi ed il Cloud. Se presenti in maniera distribuita sul territorio e con un adeguato algoritmo di gestione dei dati, questi possono temporaneamente sostituirsi al server per le funzioni basilari, garantendo un minore tempo di accesso ma soprattutto lasciando libera più banda. Con l'avvento dell'IOT, il quale si propone di collegare in rete un altissimo numero di dispositivi, i cloudlet avranno un ruolo fondamentale nel garantire un accesso adeguato alla rete.

I cloudlet possono anche essere usati per garantire la comunicazione in ambienti con scarse infrastrutture per la telecomunicazione , come i paesi in via di sviluppo o i luoghi in cui è avvenuto un disastro di grandi proporzioni, aiutando a coordinare i soccorsi. Un altro aspetto di grande pregio è la ridondanza dei dati che questi possono fornire, subentrando in caso di guasto o di attacco.

4.8.3 Riduzione delle distrazioni alla guida

Apple e Google hanno manifestato interesse ad estendere i loro ecosistemi alle autovetture. Android Auto [34] è un progetto il cui scopo è quello di ottimizzare l'interazione tra guidatore e smartphone, riducendo le distrazioni e diminuendo di

conseguenza il numero di incidenti. Il progetto mira a fornire solo informazioni utili al guidatore, che può interagire per mezzo di comandi vocali.

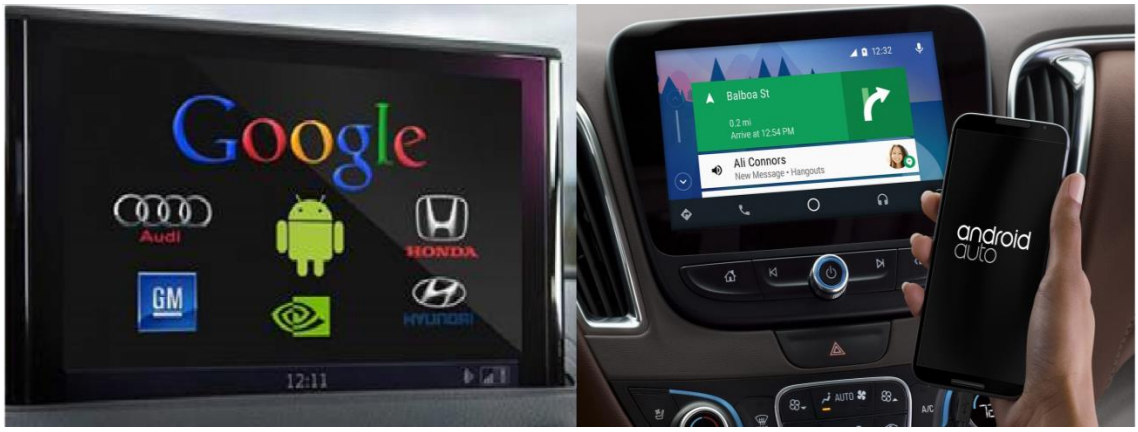

Figura 4.4

4.8.4 Self-Driving cars

Quello della guida autonoma è un obiettivo molto ambizioso. Queste auto necessitano non solo di sensori avanzati per poter analizzare l'ambiente circostante, ma anche dell'interazione con un ecosistema software, da cui ricevere informazioni per poter garantire una guida adeguata. Il Cloud può rivelarsi una valida soluzione al fine di gestire le grandi quantità di dati analizzati da una vettura autoguidata [35].

4.8.5 Vendita ed acquisto di prodotti online

Un esempio di ecosistema software dedicato all'e-commerce è quello che possiede **Amazon** come **orchestrator**. All'interno di questo ecosistema è possibile individuare facilmente gli attori principali:

L'azienda mette in contatto per mezzo della sua piattaforma **acquirenti** e **venditori** (gli **utenti finali**), i quali beneficiano dell'accesso ad un vastissimo mercato. Viene impiegato anche un servizio di consegna e localizzazione dei pacchi, i quali a loro volta

per essere gestiti utilizzano un proprio insieme di apparecchiature hardware e software.

Amazon fa affidamento su dispositivi con accesso ad internet, fra cui gli smartphone, i **produttori** di tali dispositivi ricoprono collettivamente il ruolo di **keystone**.

Inoltre Amazon presenta un proprio app store, aperto agli **sviluppatori indipendenti** [36]. Avendo uno store autonomo rispetto a Google, le due aziende competono per il dominio del mercato delle app. Tuttavia è interessante notare come comunque, nonostante la competizione reciproca, non costituiscano degli ecosistemi isolati, in quanto presentano entrambe come base gli stessi produttori hardware.

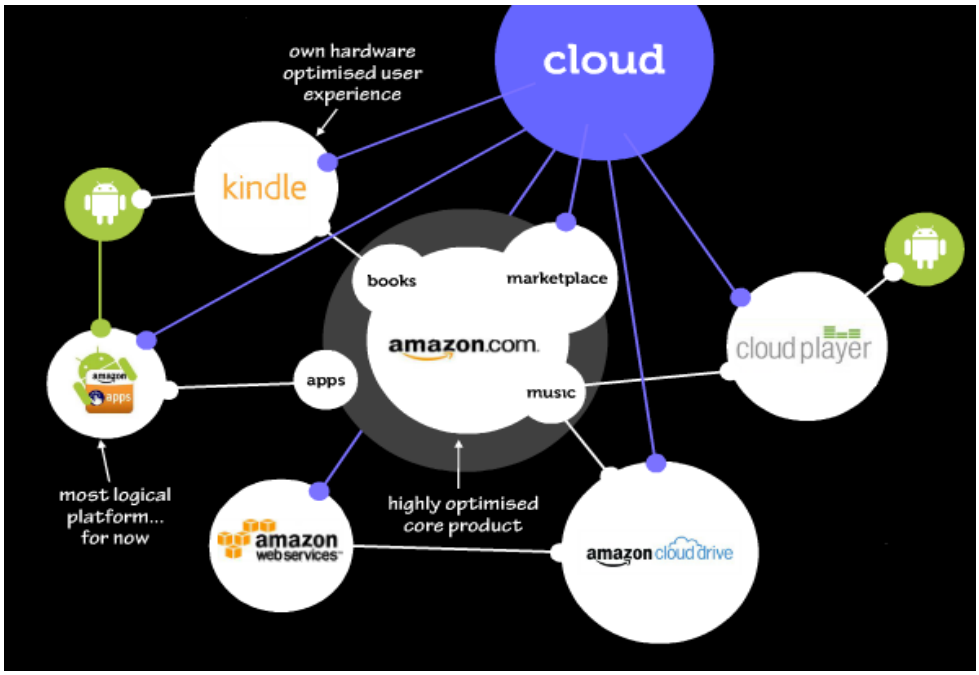

Figura 4.5: Ecosistema Amazon

CONCLUSIONI

In questo lavoro è stato illustrato il Systematic Mapping Study e confrontato con la Systematic Literature Review, descrivendone le fasi e le linee guida.

Successivamente è stata presentata una panoramica sugli ecosistemi software, con riferimenti alla situazione attuale del mercato.

Per mezzo del SMS si sono ottenuti dei documenti la cui analisi ha permesso l'individuazione delle aree di maggiore interesse ed il loro andamento nel corso degli ultimi anni. Sono stati poi selezionati dei documenti per condurre un' ulteriore analisi approfondita, determinando le caratteristiche salienti dei SECO. Sono state presentate infine le problematiche e le applicazioni degli ecosistemi software.

Come mostrato nel lavoro, gli ecosistemi software hanno trovato un ampio riscontro nel mercato attuale. Le aziende investono ingenti capitali nella manutenzione e gestione di tali ecosistemi, che come veri e propri organismi complessi, si adattano ed evolvono in base alle necessità.

I principali competitor hanno deciso di agire in larga parte autonomamente nell'estensione dei loro ecosistemi anziché affidarsi ad un'azienda già affermata in un particolare ramo, lo testimonia ad esempio l'esistenza di store differenti per le app e di diverse strutture per il Cloud.

Analizzando il trend dei SECO, si è inoltre osservato come l'interesse nei loro confronti sia in crescita negli ultimi anni, ad ulteriore testimonianza della crescente influenza nel mercato che questi hanno.

Molte delle sfide future dell'Ingegneria del Software hanno come oggetto la pianificazione adeguata di attività coordinate in un ecosistema software.

Come già visto dalle loro applicazioni, i SECO avranno un ruolo fondamentale nel migliorare la qualità della vita nei prossimi anni, offrendo soluzioni rapide ed efficaci a problemi medici, logistici o commerciali.

BIBLIOGRAFIA E SITOGRAFIA

[1] Kitchenham, B., Dybå, T. & Jørgensen, M. (2004), Evidence-based software engineering, in "Proceedings of ICSE 2004", IEEE Computer Society Press, pp. 273-281.

[2] P. A. M. S. Neto, I. C. Machado, J. D. Mcgregor, E. S. Almeida, and S. R. L. Meira. "A systematic mapping study of software product lines testing". Inf. Softw. Technol. 53, 5, 2011.

[3] B. Kitchenham, and S. Charters, "Guidelines for performing systematic literature reviews in software engineering". EBSE Technical Report, version 2.3, 2007.

[4] K. Petersen, R. Feldt, S. Mujtaba, and M. Mattsson, "Systematic mapping studies in software engineering," Proc. of the 12th international conference on Evaluation and Assessment in Software Engineering (EASE'08), 2007.

[5] Dixon-Woods, M., Agarwal, S., Jones, D., Young, B. & Sutton, A. (2005) "Synthesising qualitative and quantitative evidence: a review of possible methods", Journal of Health Services Research and Policy 10(1), 45–53.

[6] Object Oriented Design Map (Bailey et al. 2007)

[7] Wieringa, R., Maiden, N. A. M., Mead, N. R. & Rolland, C. (2006), "Requirements engineering paper classification and evaluation criteria: a proposal and a discussion", Requir. Eng. 11(1), 102–107.

[8] Kai Petersen, Robert Feldt, Shahid Mujtaba, Michael Mattsson. "Systematic Mapping Studies in Software Engineering", pp. 8-9.

[9] T. Dybå, T. Dingsøyr, "Empirical studies of agile software development: a systematic review", Information and Software Technology, 50 (9–10) (2008), pp. 833–859.

[10] T. Dybå, T. Dingsøyr, "Strength of evidence in systematic reviews in software engineering", ESEM '08: Proceedings of the Second ACM – IEEE International

Symposium on Empirical Software Engineering and Measurement, ACM, New York, NY, USA (2008), pp. 178–187

[11] B.A. Kitchenham, E. Mendes, G.H. Travassos, “Cross versus within-company cost estimation studies: a systematic review”, IEEE Transactions on Software Engineering, 33 (5) (2007), pp. 316–329

[12] http://www.treccani.it/enciclopedia/ecosistema/

[13] Jan Bosch, “From Software Product Lines to Software Ecosystems”

[14] K. Manikas, K.M. Hansen, “Software ecosystems – a systematic literature review”, J. Syst. Softw., 86 (5) (2013), pp. 1294–1306

[15] https://www.netmarketshare.com/operating-system-market share.aspx?qprid=10&qpcustomd=0&qptimeframe=M&qpsp=168&qpnp=4

[16] http://www.idc.com/prodserv/smartphone-os-market-share.jsp

[17] http://www.investopedia.com/terms/v/valueproposition.asp

[18] http://www.androidcentral.com/jelly-bean-now-67-android-devices

[19] http://www.janbosch.com/jan_bosch/Composition.html

[20] P. C. R. K. L. Bass, “Software Architecture in Practice”, third ed.,, Boston: Addison-Wesley, 2013.

[21] Romain Robbes, Mircea Lungu , “A Study of Ripple Effects in Software Ecosystems”.

[22] http://www.gps.gov/governance/advisory/meetings/2016-05/zinn.pdf

[23] http://www.navipedia.net/index.php/GPS_Services

[24] Lo’ai A. Tawalbeh, A. Basalamah, R. Mehmood, H. Tawalbeh, “Greener and Smarter Phones for Future Cities: Characterizing the Impact of GPS Signal Strength on Power Consumption”

[25] C. A. Boano and K. Römer, “External radio interference,” in Radio Link Quality Estimation in Low-Power Wireless Networks, pp. 21–63, Jul. 2013

[26] https://www.mistralsolutions.com/building-wireless-sensor-network-using smartphones/

[27] http://www.nist.gov/itl/cloud/index.cfm

[28] Leavitt, N., “Is Cloud Computing Really Ready for Prime Time?” Computer, Vol.42, No.1, pp.15-20, 2009.

[29] J. Han, M. Kamber, and J. Pei, “Data Mining: Concepts and Techniques”. San Mateo, CA, USA: Morgan Kaufmann, 2006.

[30] L. Xu, C. Jiang, J. Wang, J. Yuan, and Y. Ren, “Information security in big data: Privacy and data mining”, in IEEE Access, vol. 2, pp. 1149–1176, Oct. 2014.

[31] http://www.who.int/topics/ehealth/en/

[32] Focus, “Un software ci aiuterà a restare sani”, Agosto 2016, pag. 17.

[33] Mahadev Satyanarayanan, Rolf Schuster, Maria Ebling, Gerhard Fettweis,Hannu Flinck, Kaustubh Joshi, and Krishan Sabnani, “An Open Ecosystem for Mobile Cloud Convergence”, 2015.

[34] https://www.android.com/auto/

[35] Naveen Shivaramu Yeshodara ,Namratha S. Nagojappa, Nikhitha Kishore, “Cloud Based Self Driving Cars”, 2014.

[36] https://developer.amazon.com