Questo breve documento contiene qualche idea volutamente descritta in modo sintetico. Il lavoro effettivo verrà poi concordato con lo studente in base agli interessi specifici e ai tempi d'esecuzione. La tesi potrà essere scritta/svolta in inglese, ma per semplicità le tracce proposte sono in italiano.

Lista aggiornata febbraio 2021

Gestione automatizzata dello stato per applicazioni mobili complesse

La complessità di un'app mobile richiede sempre più spesso tecniche sofisticate per la gestione dello stato. In modo abbastanza disordinato sono nate quindi soluzioni quali: MVC, The Composeable Architecture (TCA), MVVM, Block, Cubit e chiaramente Redux. Pur avendo elementi simili, ogni modello propone la propria architettura e richiede, spesso, lo sviluppo di codice specifico (boilerplate code) o l'adozione di framework particolari. Una prima fase del lavoro dovrebbe, quindi, identificare un insieme di soluzioni alternative, studiarle al meglio per evidenziarne pregi e difetti e definire un unico framework concettuale. La seconda parte del lavoro potrebbe poi sfruttare il lavoro precedente per definire un modello astratto di riferimento e generare in automatico tutto il codice richiesto, scegliendo chiaramente una piattaforma di riferimento (ad esempio, Flutter, Swift o simili).

Da Figma a Kotlin per automatizzare la realizzazione di applicazioni mobili

Lo sviluppo di un'app mobile coinvolge almeno due figure professionali diverse: il designer e lo sviluppatore. Il primo lavora spesso con strumenti tipo Figma o simili per realizzare quello che chiama il progetto dell'app. Il secondo parte spesso dall'ambiente di sviluppo e realizza l'app stessa. Questo lavoro cerca di unire i due mondi e di definire una soluzione model-based che possa partire da Figma, o da uno strumento simile, e generare in automatico tutto il codice possibile, ed implicitamente definito dal designer. Questo richiede chiaramente anche la definizione di un modello unificato di app. Come ambiente target si potrebbe usare Kotlin Multiplatform Mobile, altra soluzione in rapida diffusione per la realizzazione di app cross-platform.

Monitoraggio avanzato di applicazioni Android

La qualità di un'app Android è un problema decisamente complesso: ciò che funziona su un dispositivo e con una versione del sistema operativo non è detto che funzioni altrettanto bene su un dispositivo diverso e con una versione (leggermente) diversa del sistema operativo. Un sistema di monitoraggio delle app potrebbe aiutare a raccogliere informazioni sul campo e cogliere, quindi, al meglio eventuali malfunzionamenti e le loro cause. Due tesi precedenti hanno affrontato il problema partendo da due prospettive diverse: una libreria esterna da usare durante la realizzazione dell'applicazione e una versione estesa del sistema operativo. Oltre a proporre, eventualmente altre soluzioni, che chiaramente dovrebbero tenere in considerazione le specificità di Android, un nuovo lavoro di tesi dovrebbe rivedere, reingegnerizzare ed estendere i prototipi esistenti alla luce delle nuove versioni di Android e dovrebbe poi valutare in modo dettagliato e sufficientemente esteso le diverse soluzioni.

Reingegnerizzazione di un sistema di guida autonoma

Tra i tanti sistemi proprietari, esiste oggi un sistema di guida autonoma open chiamato openpilot (comma.ai). Il risultato è un sistema decisamente complesso e di difficile comprensione. Il lavoro proposto avrebbe quindi l'obiettivo di fare *reverse engineering* del codice per identificare l'organizzazione del sistema, la sua architettura software e le responsabilità dei diversi componenti. Potrebbe poi essere interessante studiare il processo di gestione del progetto open, la risoluzione dei problemi ed i pattern comportamentali degli sviluppatori coinvolti. L'obiettivo del lavoro sarebbe quindi di applicare le prassi tipiche dell'ingegneria del software

ad un progetto complesso in un contesto innovativo con lo scopo di capirlo meglio e, magari, identificare un'architettura di riferimento per applicazioni di questo tipo.

Soluzioni model-based per Machine Learning

Oggi esistono molte soluzioni diverse per realizzare soluzioni basate su machine learning. Pur essendo interessanti, molte di queste soluzioni vincolano l'utente all'uso dello strumento specifico. Questa "dipendenza" è dovuta al fatto che molte soluzioni usano formati specifici per salvare i modelli ottenuti. Ad esempio, TensorFlow, Caffe, Theano e PyTorch usano tutti soluzioni particolari. Il lavoro di tesi dovrebbe quindi definire un insieme significativo di soluzioni di riferimento, studiare i loro formati e le soluzioni adottate, studiare anche eventuali proposte di standardizzazione (ad esempio, Open Neural Network Exchang, Apple Core ML format e Neural Network Exchange Format) e proporre poi una soluzione astratta che integri l'esistente e consenta di generare (automaticamente) i diversi formati richiesti.

Gestione dinamica di sistemi di Federated Machine Learning

Federated Machine Learning si sta imponendo come una soluzione distribuita e federata per sistemi di machine learning. L'esigenza della distribuzione nasce dalle sempre maggiori moli di dati da distribuire, e quindi dalle latenze che inevitabilmente un sistema centralizzato introdurrebbe, e dal rispetto della privacy: i dati acquisiti non vengono trasferiti, ma sono elaborati direttamente sul dispositivo, senza quindi violarne la privatezza. Questo lavoro non si propone di definire nuovi e più sofisticati algoritmi per machine learning federato, ma prende in considerazione il sistema distribuito. A differenza dei lavori noti in letteratura, che gestiscono tutto in modo statico, questo lavoro si propone di gestire dinamicamente le risorse coinvolte (dai nodi di computazione ai parametri degli algoritmi) per migliorare costi, tempi e qualità dei risultati. Oltre allo sviluppo di opportuni modelli teorici, questo lavoro richiede anche lo sviluppo di un opportuno simulatore per dimostrare i vantaggi della soluzione proposta.

Definizione rigorosa dei requisiti per il deployment e la gestione di applicazioni cloud

Molte applicazioni oggi sono gestite attraverso macchine virtuali e container per avere un ambiente d'esecuzione dinamico, flessibile e in grado di gestire cambiamenti in modo facile e sicuro. Molto spesso il deployment di queste applicazioni è realizzato scrivendo direttamente script opportuni per controllare i diversi strumenti utilizzati. Questo non facilita lo studio di diverse alternative ed inoltre richiede lo sviluppo di diversi script specifici per le diverse soluzioni adottate (la possibilità di riuso è decisamente limitata). Questo lavoro, al contrario, ritiene che una definizione chiara dei requisiti, non a livello applicativo ma dell'infrastruttura di esecuzione, aiuterebbe a risolvere i problemi precedenti. L'analisi dei requisiti aiuterebbe a valutare diverse soluzioni (ad esempio, in termini di tempi d'esecuzione e costi) e un insieme di requisiti definiti in modo rigoroso potrebbe servire come punto di partenza per automatizzare la creazione degli script (codice infrastrutturale) richiesti dalle diverse tecnologie.

Notazioni grafiche per il deployment avanzato di applicazioni

L'uso di soluzioni per il deployment automatizzato di applicazioni distribuite basate su macchine virtuali e/o container richiede la creazione di script specifici per la gestione dei diversi ambienti e la generazione degli artefatti necessari (ad esempio, per poter usare Ansible, Kubernetes o altre soluzioni simili). Molto spesso, questi script richiedono conoscenze specifiche e sono difficilmente riutilizzabili. Lo scopo di questo lavoro è quindi di definire linguaggi grafici (ispirati a Blockly di Google) per semplificare l'uso degli strumenti suddetti ed automatizzare poi la creazione degli script richiesti dalle diverse tecnologie. Il lavoro consiste quindi nella

scelta delle tecnologie di interesse, nella definizione del linguaggio grafico appropriato, nella realizzazione dell'editor e nella generazione automatica degli script partendo da modelli creati con il nuovo linguaggio.

Ecoware come strumento di anomaly detection

Per anni abbiamo studiato e sviluppato una soluzione, chiamata EcoWare, per il monitoraggio di applicazioni, prima a servizi e poi semplicemente cloud-based. Il nostro interesse è sempre stato rivolto a raccogliere il maggior numero di informazioni possibili per poter analizzare il comportamento dell'applicazione con la precisione necessaria. Il lavoro potrebbe quindi evolvere in due direzioni. La prima riguarda l'analisi ed il confronto delle soluzioni oggi disponibili per fare monitoraggio di applicazioni cloud. La seconda evoluzione potrebbe riguardare l'estensione e l'uso di EcoWare come strumento di anomaly detection per applicazioni basate sul paradigma a microservizi e deployate attraverso container, Kubernetes e soluzioni simili. Questa seconda ipotesi richiederebbe il ripensamento di EcoWare per il nuovo contesto, la definizione di pattern di monitoraggio appropriati, con la definizione di un'opportuna libreria di sonde, e la verifica sperimentale del sistema realizzato in contesti realistici.

Framework model-driven per applicazioni IoT

Ogni framework per applicazioni IoT (ad esempio, UML IoT, AndroidThings, NodeRED) ha la propria notazione, il proprio runtime ed il proprio linguaggio di programmazione. Una notazione unica ed un approccio model-driven aiuterebbero a semplificare il problema e consentirebbero al progettista/programmatore di concentrarsi sul cuore dell'applicazione e non sui problemi relativi alla sua implementazione. In questo modo si darebbe anche la possibilità di fare facili sperimentazioni con tecnologie diverse. La tesi dovrebbe studiare le soluzioni disponibili, realizzare un meta-modello unificato, ovvero un'unica notazione di progetto, e realizzare le trasformazioni richieste per generare il codice richiesto dai framework target a partire dal modello (istanze del meta-modello) dell'applicazione di interesse.

Analisi formale di modelli BIM

Un modello BIM (Building Information Modeling) cattura tutti gli aspetti di un edificio. Poiché nella sostanza è possibile creare un modello ad oggetti dell'edificio di interesse e di tutte le sue parti, la tesi si propone di creare un linguaggio specifico (domain-specific) per la definizione di vincoli sul modello (ad esempio, per le distanze dai vicini, per le dimensioni minime di una camera da letto o per il volume di aria scambiata da un impianto di condizionamento) ed un motore di verifica ispirato ai tanti metodi e modelli formali disponibili per l'analisi e la progettazione del software. L'integrazione con un tool di progettazione (ad esempio, AutoCAD o Revit) potrebbe essere un plus. L'idea di fondo è di iniziare a pensare ad un gemello digitale (digital twin) vivo dell'edificio di interesse.

JML

Java Modeling Language (JML) è il linguaggio di riferimento usato per insegnare la programmazione per contratto nel corso di Ingegneria del Software. Purtroppo, dopo diversi anni, e dopo varie versioni di Java, oggi non esiste un tool utilizzabile per definire e valutazione asserzioni scritte in JML; esistono diverse soluzioni parziali che funzionano solamente in casi molto particolari. La tesi si propone sia di dare una definizione precisa, compiuta ed eventualmente estendibile di JML sia di realizzare un'opportuna libreria che consenta la valutazione delle asserzioni definite. La libreria deve essere indipendente dai diversi IDE e compilatori Java utilizzati e deve anche sfruttare le caratteristiche innovative di Java per fornire meccanismi di valutazione semplici e flessibili.