Prefazione

Che cosa intendiamo quando parliamo di architettura?
Come per ogni metafora, descrivere il software attraverso le lenti dell'architettura può finire per celare più dettagli di quanti ne riveli. Può spingere a promettere più di quanto sia possibile e a dire più di quanto è stato promesso.
L'ovvio interesse per l'architettura riguarda la struttura, la quale predomina nei paradigmi e nelle discussioni che ruotano attorno allo sviluppo software: componenti, classi, funzioni, moduli, livelli e servizi, a un livello micro o macro. Ma la struttura grezza di fin troppi sistemi software spesso sfida ogni possibilità di comprensione: schemi aziendali "sovietici", da accettare come intoccabili, improbabili torri di mattoncini di legno alte più delle nubi, stratificazioni archeologiche sepolte sotto grosse pozze di fango. Non è affatto detto che la struttura del software obbedisca a quello che pensiamo dovrebbe essere una struttura architettonica.
Gli edifici hanno ovviamente una loro struttura fisica: fondamenta in pietra o cemento, estensione verticale o orizzontale, grandi o piccoli, ammirevoli o banali. La loro struttura ha pochi vincoli, se non rispettare le leggi fisiche della gravità e dei materiali impiegati. Nel nostro caso, invece, il software ha poco a che fare con la gravità, e molto con la "serietà". E di che cosa è fatto il software? A differenza degli edifici, che possono essere fatti di mattoni, cemento, legno, acciaio e vetro, il software è fatto… di software. I più grossi costrutti software sono fatti di tanti piccoli componenti software, che a loro volta sono fatti di componenti software ancora più piccoli e così via. Fino ai più minuscoli dettagli. Quando parliamo di architettura del software, questo software ha una natura ricorsiva e frattale, la quale è "incisa" nel codice e gli dà forma. Il *tutto* è nel *dettaglio*. Mescolare i livelli di dettaglio può contribuire all'architettura di un edificio; ma nel software non ha senso parlare di scala fisica. Il software ha una sua struttura (in realtà ha più strutture e anche svariati tipi di strutture), ma la sua varietà va ben oltre il confronto con la struttura fisica rinvenibile negli edifici. Possiamo certamente affermare che vi sia più attività progettuale e strutturale nel software che nell'architettura degli edifici: in questo senso, non è irragionevole considerare l'architettura del software più legata all'architettura della stessa architettura degli edifici!
Ma la scala fisica è qualcosa che gli esseri umani possono contemplare e osservare in giro per il mondo. Sebbene gradevoli e visivamente parlanti, i riquadri di un grafico di PowerPoint non sono l'architettura di un sistema software. Senza dubbio essi rappresentano una determinata visione di un'architettura, ma confondere i riquadri con il loro

significato, ovvero con l'architettura, significa perdersi sia il significato sia l'architettura: L'architettura del software non somiglia a nulla. Una determinata scelta visuale non è un dato di fatto. È solo una scelta basata su tutto un insieme di altre scelte: che cosa includere; che cosa escludere; che cosa evidenziare con una forma o un colore; che cosa celare impiegando scelte di uniformità o vere e proprie omissioni. Non vi è nulla di naturale o di intrinseco nel passaggio da una vista a un'altra.
Sebbene possa non avere senso parlare di leggi fisiche e di scala fisica nell'architettura del software, tuttavia apprezziamo e ci curiamo di alcuni vincoli fisici. La velocità del processore e l'ampiezza di banda della rete possono determinare senza appello le prestazioni di un sistema. La memoria e lo spazio di archiviazione possono limitare le ambizioni di ogni base di codice. Il software potrà anche essere fatto dei nostri sogni, ma opera comunque nel mondo fisico.

> Questo c'è di mostruoso nell'amore, mia dolcezza: che la volontà è infinita e ciò che si può fare è limitato, che il desiderio è sconfinato e l'atto è schiavo del limite.
>
> *William Shakespeare*, Troilo e Cressida, Atto III, Scena II

Nel mondo fisico operiamo noi, si muovono le aziende per le quali lavoriamo e insiste l'intera economia. Ciò ci dà un'altra unità di misura tramite la quale valutare l'architettura del software, altre forze e quantità meno fisiche tramite le quali possiamo comunicare e discutere.

> L'architettura rappresenta le decisioni progettuali significative che danno forma a un sistema, dove la significatività si misura in base al costo del cambiamento.
>
> *Grady Booch*

Il tempo, il denaro e l'impegno ci danno una sorta di scala di misura del grande e del piccolo, per distinguere gli aspetti architettonici da tutto il resto. Questa misura ci dice anche come determinare se un'architettura è "buona" o meno: non solo una buona architettura risponde, istante per istante, alle esigenze dei suoi utilizzatori, sviluppatori e proprietari, ma lo fa anche sostenendo le prove del tempo.

> Se pensate che una buona architettura sia costosa, provate a usare una cattiva architettura.
>
> *Brian Foote e Joseph Yoder*

I tipi di interventi sperimentati, in genere, a partire dallo sviluppo di un sistema non dovrebbero essere modifiche costose, difficoltose, che richiedono progetti a se stanti; piuttosto dovrebbe trattarsi di piccoli interventi, che rappresentano la normalità del flusso di lavoro quotidiano e settimanale.
Questo punto ha un problema non da poco conto legato alla fisica: il viaggio nel tempo. Come possiamo conoscere oggi quali saranno questi interventi, in modo da poter selezionare le decisioni significative che li guideranno? Come possiamo ridurre i prossimi impegni e costi di sviluppo non essendo in possesso né di una sfera di cristallo, né di una macchina per viaggiare nel tempo?

> L'architettura è costituita dalle decisioni che vorresti poter prendere nelle fasi preliminari di un progetto, ma che non necessariamente riuscirai a prendere prima di tante altre.
>
> *Ralph Johnson*

Comprendere il passato è già abbastanza complesso; la nostra capacità di cogliere il presente è, quanto meno, parziale; predire il futuro non è affatto banale.
Questo perché molte strade si biforcano, e in più punti.
Lungo un percorso oscuro della storia ci può giungere l'idea che la solidità e stabilità di un'architettura dipendano dall'imposizione di un'autorità e dell'applicazione della massima rigidità. Se una modifica è costosa, viene eliminata, e le sue cause vengono sottomesse o piegate a esigenze burocratiche. Il mandato del responsabile dell'architettura è totale e totalitario, e l'architettura stessa diviene una sorta di distopia per i suoi sviluppatori e una costante fonte di frustrazione per tutti.
Lungo un altro percorso si percepisce più un aroma di liberalità speculativa. Un percorso costellato di elementi *hard-coded*, innumerevoli parametri, lapidi di codice "morto" e molta più complessità diffusa di quanta ne possa sopportare una qualsivoglia attività di manutenzione.
Il percorso cui siamo più interessati è quello più "pulito". Che riconosce la leggerezza del software e punta a preservarla come una delle priorità del sistema. Che riconosce che stiamo operando sulla base di conoscenze incomplete, ma che comprende anche che, in quanto esseri umani, operare con conoscenze incomplete è qualcosa che facciamo normalmente, qualcosa che ci appartiene. Ha più a che fare con i nostri punti di forza che con le nostre debolezze. Creiamo cose e ne scopriamo di nuove. Poniamo domande e conduciamo esperimenti. Una buona architettura deriva dalla consapevolezza che essa è più un viaggio che una destinazione, più un processo evolutivo che un artefatto congelato.

> L'architettura è un'ipotesi, la quale deve essere dimostrata dall'implementazione e dalle misure.
>
> *Tom Gilb*

Per imboccare questo percorso occorono cautela e attenzione, riflessione e osservazione, pratica e principi. Questo all'inizio può sembrare un percorso lento, ma tutto dipende dal modo in cui si cammina.

> L'unico modo per procedere rapidamente, è fare le cose per bene.
>
> *Robert C. Martin*

Quindi, buon viaggio.

Kevlin Henney
Maggio 2017