DELIVERABLE 3: APPLICAZIONE DELLA TECNICA CBAM A UNA DECISIONE PROGETTUALE

Martina De Maio

Studentessa Laurea Magistrale in Ingegneria Informatica,

Università degli Studi di Roma “Tor Vergata”, Roma, Italia

Matricola: 0296447

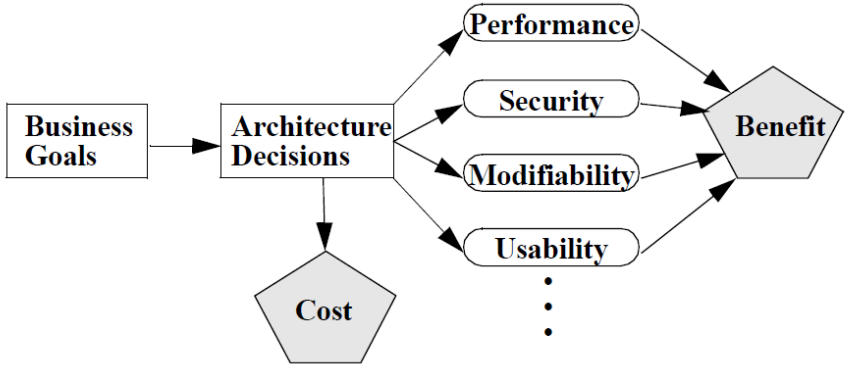
Agenda

Il lavoro è organizzato secondo la seguente scaletta:

0. Introduzione: CBAM overview
1. Step 1
2. Step 2
3. Step 3
4. Step 4
5. Step 5
6. Step 6
7. Step 7
8. Step 8

CBAM OVERVIEW

- Un'analisi ***costi-benefici*** è un processo utilizzato dalle organizzazioni per analizzare quali decisioni prendere e quali rinunciare.
- Prima di intraprendere un nuovo progetto, è consigliabile condurre un'analisi ***costi-benefici*** per supportare un'organizzazione sulla fattibilità delle scelte progettuali. L'esito dell'analisi determinerà se il progetto è finanziariamente fattibile, con rischi calcolati, o se sia necessario modificare le scelte iniziali o addirittura perseguire un altro progetto.
- La tecnica ***CBAM*** (*Cost Benefit Analysis Method*) permette di confrontare i costi totali di un programma o di un progetto con i suoi benefici, valutando diverse alternative o linee d'azione, in modo da trovare i punti di forza e di debolezza che si sarebbero potuti realizzare scegliendo un'alternativa rispetto a un'altra, e in modo da garantire che le decisioni prese riducano i rischi e massimizzino i profitti.
- Nel ciclo di sviluppo del software, tale approccio consente di anticipare il più possibile nel tempo la valutazione di potenziali problematiche, che, se gestite invece direttamente durante la fase di sviluppo, potrebbero portare a sprechi in termini di risorse umane ed economiche.



CBAM OVERVIEW

Durante le prime fasi del processo di sviluppo viene progettata l'architettura del software, sulla base di specifici requisiti funzionali, non funzionali (attributi di qualità) e business goals.

Un attributo di qualità è definito come una proprietà non funzionale desiderata, ad esempio sicurezza, prestazioni, riutilizzabilità del software, affidabilità, etc.

In generale, alternative diverse soddisferanno in maniera differente un determinato attributo di qualità. Ad esempio, un linguaggio di programmazione può avere performance più elevate rispetto a un altro.

Step 1

Nel caso in questione si applica la tecnica CBAM ad un applicativo di analisi in un contesto ***Big Data*** simile al lavoro svolto sul Deliverable2, che prevede l'analisi di una significativa quantità di dati .

In particolare, la scelta progettuale che si andrà ad analizzare è quella relativa al linguaggio di programmazione.

Le varie alternative analizzate sono:

- Linguaggio ***Java***
- Linguaggio ***Python***
- Linguaggio ***C++***

Step 2: Quality attributes weight

Gli attributi di qualità individuati, e la loro importanza, sono:

- Performance: 20
- Security: 15
- Simplicity : 10
- Interoperability: 20
- Integrability: 15
- Portability: 20

Essendo in un contesto big data, le performance di un linguaggio di programmazione hanno una notevole importanza, in quanto condizionano notevolmente i tempi di esecuzione del programma, cosi come altrettanto importante è la portabilità, ossia la dipendenza dalla piattaforma specifica, la semplicità di programmazione e di debugging, la disponibilità di librerie per una efficace e veloce *interoperability* con altri ambienti software.

Step 3: Score

Per ogni alternativa, si definisce, con uno score che va da -1 a 1, quanto l'alternativa soddisfa l'attributo di qualità.

Quality Attributes	Java	Python	C++
Performance	0.7	0.6	0.8
Security	0.9	0.6	0.4
Semplicity	0.8	0.8	-1
Interoperability	0.7	0.8	0.2
Integrability	0.7	0.7	0.2
Portability	0.9	0.9	-0.5

Step 4: Risk

Per ogni alternativa si definisce, con uno score crescente che va da 0 a 1, il corrispondente rischio.

	Java	Python	C++
Risk	0.4	0.4	0.9

Step 5: Benefit score

$$Benefit(AS_i) = (\sum_j (AS_{ij} * QAscore_j)) * |Risk_i - 1|$$

Quality Attributes	Java	Python	C++
Performance	0.7*20 = 14	0.6*20 = 12	0.8*20 = 16
Security	0.9*15 = 13.5	0.6*15 = 9	0.4*15 = 6
Semplicity	0.8*10 = 8	0.8*10 = 8	-1*10 = -10
Interoperability	0.7*20 = 14	0.8*20 = 16	0.2*20 = 4
Integrability	0.7*15 = 10.5	0.7*15 = 10.5	0.2*15 = 3
Portability	0.9*20 = 18	0.9*20 = 18	-0.5*20 = -10
Benefit	**46.8**	**44.1**	**0.9**

Step 6: Cost

Si assume un costo di licensing per ogni linguaggio di programmazione pari a 1, essendo essi disponibili gratuitamente.

Step 7: Desirability

Avendo ipotizzato un costo pari a 1 per ogni linguaggio di programmazione, il parametro ***Desirability*** assume lo stesso valore del parametro Benefit.

$$Desirability(ASi) = Benefit(ASi)/Cost(ASi)$$

Step 8: Ranking

In accordo con la Desirability, e quindi il Benefit, il ranking finale è il seguente:

- Java: **46.8**
- Python: **44.1**
- C++: **0.9**