UML

Uno standard dell'Object Management Group (OMG).

Definizione ufficiale:

"un linguaggio per specificare, visualizzare, e realizzare i prodotti di sistemi software, e anche per il business modeling. L' UML rappresenta una collezione di "best engineering practices" che si sono dimostrate utili nella progettazione di sistemi complessi e di grandi dimensioni."

Versione minimalista: una notazione standard (comune a livello internazionale) per rappresentare in modo visuale (grafico) le caratteristiche dei sistemi software.

Storia:

Il processo di standardizzazione ed unificazione metodologica che ha portato ad UML ha avuto origine nel 1994 con l'ingresso di Jim Rumbaugh nella società Rational, in cui operava Grady Booch, ed è proseguito con l'arrivo nella stessa società, a fine 1995, di Ivar Jacobson.

- ottobre 1995: Unified Method versione 0.8 (Booch e Rumbaugh)
- giugno 1996: Unified Modeling Language (UML) versione 0.9 (Booch, Rumbaugh, Jacobson)
- ottobre 1996: UML 0.91 (Booch, Rumbaugh, Jacobson)
- gennaio 1997: UML 1.0 (Booch, Rumbaugh, Jacobson e partner UML, tra cui Microsoft, Oracle, Hewlett-Packard, Digital, Texas Instruments), in risposta alla richiesta avanzata dall' OMG (Object Management Group) per un framework di interoperabilità tra strumenti di analisi e disegno. All'OMG arrivano anche altre proposte, tra cui quelle firmate da IBM e Platinum
- settembre 1997 1997: Dopo accordi con Platinum, IBM e le altre "cordate", la versione 1.1 dello Unified Modeling Language viene sottoposta all'approvazione di OMG. E' una proposta congiunta di Rational Software, Microsoft, Hewlett-Packard, Oracle, Sterling Software, MCI Systemhouse, Unisys, ICON Computing, IntelliCorp, i-Logix, IBM, ObjecTime, Platinum Technology, Ptech, Taskon, Reich Technologies, Softeam, e altri.
- 16 novembre 1997: UML 1.1 diventa ufficialmente uno standard OMG (con il nome ufficiale di "UML OMG 1.0")
- dicembre 1998: UML 1.2 - revisione editoriale, correzione di errori tipografici e grammaticali
- giugno 1999: UML 1.3 - variazioni minori ai diagrammi delle classi, dei casi d'uso, di interazione; specifica degli aspetti semantici e notazionali per modelli e sottosistemi.
- maggio 2001 : UML 1.4 - variazioni minori
- marzo 2003: UML 1.5 - vengono inglobate le "Action Semantics"
- giugno 2003: UML 2.0 approvata in bozza - molte variazioni significative, completa ristrutturazione del metamodello
- marzo 2005: UML 2.0 diventa la versione ufficiale

Può essere usato per:

1. **Pensare**.
   Nel corso delle attività di sviluppo o di evoluzione di un sistema, UML può servire agli sviluppatori per ragionare sui problemi e sulle soluzioni.
2. **Comunicare (e documentare)**.
   Tra soggetti diversi, distanti nello spazio (ad esempio da analisti separati fisicamente dai progettisti e dagli sviluppatori) o nel tempo (posso trovarmi a riprendere in mano modelli che io stesso ho creato in passato). Tra aziende diverse, che devono collaborare su basi contrattuali.

Naturalmente, c'è qualche differenza tra l'utilizzo di UML per pensare e per comunicare. Se lo uso per ragionare, posso permettermi di fare solo i diagrammi che mi servono al momento, al livello di dettaglio e di precisione che mi interessa, senza pormi problemi di comunicazione. Se invece lo uso per comunicare e documentare, è necessario che chiarisca prima di tutto con chi voglio comunicare, che cosa esattamente voglio comunicare, e a quale livello di dettaglio e di precisione.

Obbiettivo:

Lo Unified Modeling Language (UML) vuole essere, secondo le intenzioni dei suoi autori, Booch Jacobson e Rumbaugh, "il linguaggio per la creazione di modelli software". Un linguaggio, cioè, “universale per la progettazione (modeling) dei sistemi, nel senso che può esprimere modelli di varia tipologia e creati per obiettivi diversi, proprio come un linguaggio di programmazione o un linguaggio naturale possono essere usati in molti modi diversi".

Perché l’UML?

Perché nel campo delle metodologie di analisi e disegno object oriented la confusione regna ancora sovrana. Qualcuno ne aveva contate più di cinquanta, ciascuna con i propri modelli ed i propri formalismi. Sebbene le più diffuse non superino la decina, non è facile per le aziende intenzionate a muoversi nel settore OO scegliere la più adeguata. La confusione esistente a livello metodologico si ripercuote poi a livello di strumenti CASE per la progettazione object oriented: sebbene l’orientamento prevalente tra i produttori sia quello di supportare notazioni diverse, l’assenza di uno standard riconosciuto rende rischiosa ogni valutazione di investimento in prodotti di questo genere. In questo confuso panorama, le proposte metodologiche avanzate individualmente da Booch, Rumbaugh e Jacobson risultavano già tra le più diffuse, il che costituisce un fattore positivo per le prospettive di accettazione dell’Unified Modeling Language.

Caratteristiche dell’UML

L'UML intende essere, appunto, un linguaggio universale mediante il quale descrivere i diversi modelli prodotti nel corso dello sviluppo software, a prescindere dalle procedure utilizzate per realizzarli. Booch Jacobson e Rumbaugh precisano che definire "un unico processo per tutti gli stili di sviluppo non sembra possibile e neppure desiderabile. [...] Comunque, l'UML può venire utilizzato per esprimere gli output di tutti i diversi processi, cioè i modelli che vengono prodotti". Le fondamenta dell'UML sono costituite da un metamodello, che definisce le caratteristiche e le relazioni esistenti tra le diverse componenti di un progetto software (es. classi, attributi, procedure, moduli, ...). Il metamodello costituisce la base per l'implementazione dell'UML da parte dei produttori di strumenti per lo sviluppo (CASE tools, ambienti visuali), e per l'interoperabilità tra i diversi strumenti. L'intenzione degli autori è di produrre una specifica formale del metamodello al momento della sottomissione dell'Unified Modeling Language all'OMG per la sua approvazione come standard (fine '96 - inizio '97).

Sulla base di quanto definito nel metamodello, vengono poi proposti alcuni modelli diagrammatici:

Livello LOGICO

- Diagramma dei casi d'uso (Use Case Diagram)
- Diagramma delle classi (Class Diagram)
- Diagramma di sequenza (Sequence Diagram)
- Diagramma di collaborazione tra oggetti (Collaboration Diagram)
- Diagramma di transizione di stato (State Diagram)

Livello FISICO

- Diagramma delle componenti software (Component Diagram)
- Diagramma di allocazione delle componenti (Deployment Diagram)

I modelli diagrammatici presentano ovviamente una notazione unificata, frutto di un compromesso tra quelle originarie proposte singolarmente da Booch Jacobson e Rumbaugh, e già adottate dalla maggior parte degli strumenti CASE Object Oriented.In realtà, il progetto dei tre metodologi non limita l'utilizzo dell'UML al solo ambito Object Oriented. Il loro obiettivo era quello di definire un linguaggio di progettazione che fosse in grado di supportare anche linguaggi e componenti non OO, come file e moduli software tradizionali, ed ogni tipo di componente software a livello fisico.

Anche in questo caso i risultati non si sono fatti attendere, in quanto Microsoft e Hewlett Packard hanno aderito al "progetto UML", che acquista così una ulteriore capacità di attrazione anche per le realtà, praticamente tutte, non ancora strettamente object oriented.

A livello Aziendale:

Molte aziende, in Italia e nel resto del mondo, stanno adottando UML, innanzitutto con l'obiettivo di migliorare la comunicazione relativa ai progetti software, all'interno dell'organizzazione e nei confronti di terze parti.

In quanto notazione universale, UML costituisce un linguaggio comune per l'analisi, il disegno, la documentazione dei sistemi. Può quindi agevolare il flusso comunicativo sia all'interno delle aziende che sviluppano software, che nel rapporto tra clienti e fornitori di sistemi. Non a caso, uno dei campi nei quali l'utilizzo di UML si sta maggiormente diffondendo è quello della gestione delle subforniture nell'ambito della progettazione di sistemi complessi. Dove società diverse concorrono allo sviluppo, producendo soluzioni parziali da integrare in un'architettura complessiva, la documentazione in formato UML agevola la comunicazione tra le parti.

Il secondo motivo che favorisce l'adozione di UML è legato a un rinnovato interesse per le attività di progettazione (analisi e disegno). I sistemi distribuiti diventano sempre più complessi, e, proprio per poterne governare la complessità, risulta necessario dedicare tempo alla definizione della loro architettura, logica e fisica, sotto forma di rappresentazioni sintetiche (modelli), prima di buttarsi a capofitto nella scrittura del codice. Inoltre, le tecnologie utilizzate per l'implementazione sono sempre più basate sull'object oriented e sull'integrazione di componenti, ed è importante che anche le attività di progettazione siano basate sugli stessi princìpi. UML, essendo un linguaggio di modellazione basato sull'approccio object oriented, risponde perfettamente ad entrambe queste esigenze.

La spinta all'adozione di UML può quindi provenire dall'esterno (allineamento alle richieste dei clienti, ad uno standard internazionale), o dall'interno dell'organizzazione (miglioramento della comunicazione, adozione di approcci di progettazione object-oriented). In ogni caso, è probabile che chi intende utilizzare UML decida di avvalersi di uno strumento di visual modeling, cioè di un software commerciale per la produzione di modelli. Certo, i diagrammi UML si possono disegnare anche con carta e matita; ma è sicuramente più comodo, e forse più economico, utilizzare uno strumento software per crearli e gestirli.

Le soluzioni disponibili sono di due tipi: diagrammatori puri (come Flowcharter o Visio), oppure strumenti dedicati alla produzione di modelli UML.

I diagrammatori puri consentono di creare ogni genere di diagramma, avvalendosi di mascherine (template), legate ad ambiti notazionali specifici, e ne esistono alcuni capaci di rappresentare tutti i simboli grafici di UML. Il limite di questo tipo di strumenti è che consentono appunto di creare solo diagrammi, cioè rappresentazioni grafiche, ma non di documentare gli aspetti semantici (il significato) degli elementi rappresentati, né di

controllare la correttezza sintattica delle associazioni tra elementi. Nulla vieta, ad esempio, di rappresentare graficamente una "gerarchia di aggregazione" che leghi più "attività" ad una "classe", anche se si tratta di un'associazione che UML non consente.

Gli strumenti di visual modeling veri e propri, invece, offrono una gestione completa dei modelli UML, sintattica e semantica, e non solo la produzione di diagrammi. Consentono la generazione di codice a partire dai modelli, ed il reverse engineering (riproduzione di modelli) a partire da codice esistente.