

Indice

INDICE	2
1. INTRODUZIONE	4
1.1 CONTESTO	4
1.2 OBIETTIVI	9
1.3 PERCORSO DI LAVORO	10
1.4 STRUTTURA DELLA TESI	14
2. STATO DELL'ARTE	16
2.1 MODEL DRIVEN DEVELOPMENT	16
2.1.1 <i>Model Driven Architecture (MDA)</i>	18
2.2 TECNICHE E LINGUAGGI DI TRASFORMAZIONE DI MODELLI	26
2.2.1 <i>Trasformazioni in MDA</i>	26
2.2.2 <i>Linguaggi per MDA</i>	26
2.2.3 <i>Tool per MDA</i>	28
2.3 MODEL DRIVEN DEVELOPMENT DI APPLICAZIONI WEB	29
2.4 I PATTERN	33
2.4.1 <i>Classificazione dei Pattern</i>	35
2.4.2 <i>Pattern e Applicazioni Web</i>	38
3. MODELLAZIONE CONCETTUALE E METAMODELLO UWA	41
3.1 IL RUOLO DELLA MODELLAZIONE CONCETTUALE	41
3.2 L'APPROCCIO UWA/UWAT+	43
3.2.1 <i>Requirements Elicitation</i>	44
3.2.2 <i>Hypermedia e Operation Design</i>	44
3.2.3 <i>Transaction Design</i>	48
3.2.4 <i>L'estensione UWAT+</i>	51
3.3 IL METAMODELLO UWA	52
3.3.1 <i>Struttura del Metamodello</i>	53
3.4 FORMATI DI MEMORIZZAZIONE E TOOLS SCELTI	88
3.4.1 <i>Standard Meta Object Facility (MOF)</i>	88
3.4.2 <i>Eclipse Modeling Framework (EMF)</i>	94
3.4.3 <i>Topcased Ecore Editor</i>	100
4. MODELLAZIONE LOGICA E METAMODELLO UML-MVC	101
4.1 IL RUOLO DELLA MODELLAZIONE LOGICA	101
4.2 IL PATTERN MVC	103
4.2.1 <i>I componenti Model, View e Controller</i>	106
4.3 UN ESEMPIO DI MODELLO LOGICO UML PER APPLICAZIONI MVC	108
4.3.1 <i>Una semplice Web application per la condivisione di Post-it</i>	109
4.3.2 <i>Il Modello Logico UML-MVC prodotto</i>	110
4.4 IL METAMODELLO UML-MVC	124
4.4.1 <i>Struttura del Metamodello</i>	126
5. TRASFORMAZIONE DI MODELLI CONCETTUALI UWA IN MODELLI LOGICI UML-MVC	138
5.1 PREMESSA	138
5.2 LE EURISTICHE PER LA TRADUZIONE DI MODELLI CONCETTUALI UWA IN MODELLI LOGICI UML-MVC	139
5.2.1 <i>Information Design</i>	140
5.2.2 <i>Transaction Design</i>	145
5.2.3 <i>Navigation Design</i>	146

5.2.4 <i>Publishing Design</i>	147
5.2.5 <i>Conclusioni</i>	150
5.3 IMPLEMENTAZIONE DELLE EURISTICHE INDIVIDUATE	151
5.3.1 <i>Premessa</i>	151
5.3.2 <i>I linguaggi di trasformazione e i tools scelti</i>	152
5.3.3 <i>Le regole di trasformazione in linguaggio ATL</i>	155
5.4 AUTOMATIZZAZIONE DELLA TRASFORMAZIONE DEL MODELLO UWA NEL MODELLO LOGICO UML-MVC	164
6. CASO DI STUDIO-UN'APPLICAZIONE PER LA CONDIVISIONE DI POST-IT	169
6.1 DESCRIZIONE DELL'APPLICAZIONE E REQUISITI DI MODELLAZIONE	169
6.2 MODELLAZIONE CONCETTUALE MEDIANTE UWA	170
6.2.1 <i>Struttura del modello prodotto</i>	171
6.2.2 <i>Descrizione del modello prodotto</i>	172
6.3 MODELLAZIONE LOGICA UML-MVC	194
6.3.1 <i>Modello UML-MVC prodotto mediante trasformazione manuale</i>	194
6.3.2 <i>Modello UML-MVC prodotto automaticamente</i>	199
6.4 CONFRONTO FRA I MODELLI LOGICI	215
6.5 IMPLEMENTAZIONE DELL'APPLICAZIONE PROGETTATA	220
6.5.1 <i>Framework MVC oriented: Java Server Faces</i>	220
6.5.2 <i>Ambienti di sviluppo</i>	224
6.5.3 <i>Apache Jakarta Tomcat</i>	225
6.5.4 <i>DBMS MySql</i>	225
6.6 IL PROTOTIPO REALIZZATO	225
7. CONCLUSIONI E SVILUPPI FUTURI	232
BIBLIOGRAFIA	240
8. APPENDICE	243
8.1 METODOLOGIE PER LO SVILUPPO DI APPLICAZIONI WEB	243
8.2 UTILIZZO DELL'ADD-IN UWA PER RATIONAL ROSE	252
8.2.1 <i>Introduzione al modello</i>	252
8.2.2 <i>Information Model</i>	253
8.2.3 <i>Navigation Model</i>	261
8.2.4 <i>Publishing Model</i>	265

1. Introduzione

1.1 Contesto

Nel settore dei servizi informatici si continua ad assistere alla crescente produzione di applicazioni software orientate allo sfruttamento delle risorse di Internet, che forniscono agli utenti della stessa servizi nuovi e sempre più innovativi. Il fenomeno porta alla nascita di applicazioni Web sempre più complesse e sofisticate, in grado di integrare funzionalità diverse, ed è oggetto di investimenti sempre più ingenti da parte di società ed aziende.

In questo contesto, risulta di fondamentale importanza affrontare la progettazione e la costruzione di queste applicazioni avvalendosi dell'ausilio di metodologie, modelli e strumenti di sviluppo idonei, capaci di assicurare livelli qualitativi elevati, mantenendo i costi di sviluppo contenuti.

Un'azienda di sviluppo software, ha nell'attività di analisi e progettazione uno dei suoi punti cardine. La disponibilità di metodologie di progettazione abili nel considerare e trattare tutti gli aspetti che caratterizzano un'applicazione Web e di strumenti di supporto adeguati, è di fondamentale importanza nella realizzazione di applicazioni di qualità. Da un lato l'impiego di una metodologia di sviluppo adeguata permette all'azienda produttrice di regolare opportunamente i vari processi interni, in modo tale da fornire un prodotto che effettivamente corrisponda alle esigenze del committente. Dall'altro l'impiego stesso della metodologia fornisce indirettamente a quest'ultimo la garanzia di avere un prodotto solido, funzionale e flessibile. Solido perché, se l'applicazione è progettata opportunamente, sarà meno vulnerabile agli attacchi sferrati da altri programmi; funzionale perché sarà in grado di fornire servizi con una qualità maggiore e in modo più rapido; flessibile perché potrà essere facilmente estesa con nuove funzionalità o mediante barriere che contribuiscano a renderla ancora meno vulnerabile.

Di pari passo con il fenomeno dello sviluppo del Web e delle sue applicazioni, si sta facendo strada l'esigenza di rendere queste applicazioni uniformi sulla base di standard che permettano lo scambio di informazioni e la loro integrabilità.

Particolarmente importante in questo senso è il lavoro svolto dall'Object Management Group (OMG) nella definizione di una serie di importanti standard tecnologici, che costituiscono dei solidi punti di riferimento per tutte le società informatiche che producono software di qualità.

Negli ultimi anni inoltre, le applicazioni Web si sono evolute rapidamente da semplici siti statici, in grado solo di fornire all'utente informazioni e contenuti, verso applicazioni Web data- e operation-intensive capaci di provvedere alle funzionalità di business. In termini semplicistici, un'applicazione Web può essere intesa come un sistema software in grado di consentire ai suoi utenti l'esecuzione di business logic attraverso un Web browser [Conallen]. Naturalmente, all'aumentare della complessità delle applicazioni Web è cresciuta anche la necessità di modellare gli aspetti che le caratterizzano. La complessità di questi sistemi, infatti, ne rende difficoltosa la progettazione e realizzazione.

La modellazione incoraggia a scomporre il problema in pezzi gestibili ed ad utilizzare meccanismi di comunicazione basati sull'uso di un linguaggio comune. I modelli possono essere considerati come semplificazioni della realtà ed possono essere costruiti a diversi livelli di astrazione [RuJaBo]. Ogni modello fornisce una semplificazione di un sistema complesso secondo un particolare punto di vista. Per tale ragione, è utile che nel processo di sviluppo di un sistema software siano coinvolti più modelli, ognuno dei quali sia in grado di fornire, dal proprio punto di vista, una vista semanticamente completa del sistema.

Nell'ambito dell'Ingegneria del Software, dunque, le tecniche di sviluppo basate sui modelli hanno riscontrato un interesse via via crescente tale da spingere gradualmente all'adozione di questo approccio molte delle grandi aziende software operanti in diversi settori importanti dell'industria, del finance e dei trasporti. L'elevato numero di applicazioni Web e la loro crescente complessità ha portato alla necessità per gli sviluppatori di applicazioni Web di disporre di strumenti che ne semplifichino la progettazione, lo sviluppo e la manutenzione nel tempo. Nel caso particolare dei siti informativi, nei quali i contenuti devono essere modificati con continuità da parte di personale, generalmente non tecnico, devono essere disponibili strumenti che ne facilitino l'aggiornamento. Lo

sviluppo di applicazioni Web, in relazione alle sue caratteristiche, è un dominio specifico nel quale possono essere applicate con successo tecniche Model-based.

L'approccio UWA/UWAT+ alla progettazione di applicazioni Web [UWA D6-D9] [Distante1] prevede che, dopo aver effettuato la Requirements Elicitation per la definizione dei requisiti dell'applicazione, il problema complessivo venga suddiviso nei seguenti sottoproblemi: Information Design, Transaction Design, Navigation Design, Operation Design, Publishing Design e Customization Design. Per ciascuno dei precedenti aspetti di design è previsto l'impiego di un modello che, tramite l'uso di appositi concetti e primitive di modellazione, fornisca la descrizione dell'applicazione relativamente a quel particolare aspetto.

L'obiettivo principale della progettazione UWA è quello di descrivere un'applicazione Web così come sarà percepita dalle diverse categorie di utenti che sono previste per essa. Il punto di vista adoperato nella realizzazione dei modelli dell'applicazione è infatti quello dell'utente e non del progettista software o dello sviluppatore. In tal senso si parla di modello concettuale user-centered dell'applicazione [UWA D2]. Invece, il problema legato alla descrizione del modello logico dell'applicazione, intendendo con questo la descrizione di come l'applicazione si presenta dal punto di vista architetturale e implementativo, non costituisce una prerogativa di UWA.

Se da un lato la metodologia UWA/UWAT+ si presta alla progettazione/descrizione di una qualunque applicazione Web, quale che siano l'architettura e le tecnologie software di implementazione, ed è in grado di ritrarre la prospettiva utente, i modelli che con essa si producono risultano essere ad un livello di astrazione troppo distante dal livello dell'implementazione. Attraverso i diversi modelli di design proposti da UWA si riescono a fornire le specifiche sui contenuti dell'applicazione, la loro struttura e organizzazione, la navigazione e la strutturazione dell'applicazione in pagine, le operazioni e le transazioni che verranno offerte all'utente. Di contro, la trasformazione di questi concetti di modellazione in codice per l'implementazione, non sempre è immediata e semplice. In assenza di livelli di design più prossimi all'implementazione, in grado di fornire specifiche sulle componenti software e il codice da produrre, spesso non rimane altra scelta che affidarsi alle capacità interpretative del singolo sviluppatore lasciandogli, in tal modo,

un margine di scelta troppo ampio che può condurre alla realizzazione di applicazioni poco corrispondenti alla modellazione effettuata.

La distanza esistente tra il design concettuale e l'implementazione, infatti, fa sì che in assenza di modelli di design a livelli di astrazione intermedi, quest'ultima proceda in maniera quasi del tutto indipendente dalla prima. Risulta pertanto opportuno prevedere un altro tipo di design, da far seguire a quello concettuale: il design logico. Si tratta di un livello di design più vicino all'implementazione e quindi al modo con cui viene strutturata l'architettura di sistema e il codice. Compito del design logico è quello di stabilire un legame fra il design concettuale e l'implementazione dell'applicazione, consentendo l'allineamento fra modelli (a tutti i livelli di astrazione) e codice per tutto il ciclo di vita dell'applicazione e in particolare nelle fasi di manutenzione ed evoluzione successive alla sua messa in esercizio.

Il risultato finale sarà quello di avere due tipi di design, ognuno con uno scopo ben preciso: il design concettuale che descrive ad un alto livello di astrazione, cosa fa l'applicazione dal punto di vista dell'utente in termini di fruizione delle informazioni, di processi da svolgere, attività da eseguire e percorsi navigazionali disponibili, ed il design logico, che descrive come è fatta l'applicazione al suo interno e, pertanto, come è implementata.

L'introduzione di un livello di design aggiuntivo complica il processo di progettazione dell'applicazione e può richiedere (almeno in linea di principio) maggiori risorse. Per contro però, esso garantisce la realizzazione di applicazioni Web che aderiscono meglio alle specifiche di design concettuale (quindi vicine all'utente) e che sono più facili da mantenere ed evolvere grazie alla disponibilità di documentazione coerente ed aggiornata. Questo giustifica e ripaga il maggior sforzo iniziale di progettazione.

L'approccio allo sviluppo di sistemi software che prevede l'uso sistematico di modelli e trasformazione di modelli nell'intero ciclo di vita del sistema, viene definito Model Driven Engineering (MDE). MDE si differenzia nettamente dall'approccio code-centric in quanto sposta l'attenzione dal codice ai modelli, attorno ai quali ruota l'intero processo di design ed implementazione. Con MDE un sistema software viene sviluppato attraverso un set di modelli a diversi livelli di astrazione. L'obiettivo è incrementare la produttività e ridurre i tempi e i costi di

sviluppo di sistemi complessi promettendo miglioramenti significativi nella direzione di un alto livello di astrazione dei linguaggi e del livello di riuso¹.

Il Web Engineering è uno specifico dominio nel quale l'approccio model-driven è da qualche tempo impiegato con successo. In tal senso si parla di Model-Driven Web Engineering (MDWE). Diverse metodologie di sviluppo, tra cui OO-H [CaGòPa], UWE [KoKr-02], OOHDM [ScRo-98] e WebML [Ceri-03], si sono recentemente mosse in questa direzione fornendo un set di metodi e tools per il design e lo sviluppo di applicazioni Web in ottica Model-Driven. MDWEnet [MDWEnet] è una recente iniziativa avviata da un gruppo di ricercatori di diversi atenei europei nell'ambito del Model-Driven Web Engineering. L'obiettivo è quello di definire un metamodello comune alle diverse metodologie di sviluppo basate sull'approccio MDWE e di condividere esperienze e conoscenze maturate in questo campo.

I concetti definiti da MDE rappresentano una generalizzazione del Model Driven Development (MDD) e della Model Driven Architecture (MDA) definita dall'Object Management Group (OMG). Quest'ultima, in relazione all'evoluzione dei linguaggi di modellazione del software (UML), intende far sì che le applicazioni software vengano generate attraverso un processo di trasformazione di modelli: un approccio aperto e non proprietario alla sfida del cambiamento tecnologico e di business; un'evoluzione del software engineering che fa leva sulla forte e formale separazione tra modelli funzionali (cosa deve fare il sistema) e modelli tecnologici (come usare le tecnologie), costruiti utilizzando standard non proprietari definiti e mantenuti da OMG®, da cui possono essere generate le applicazioni eseguibili, virtualmente su qualunque piattaforma, aperta o proprietaria, incluso ma non limitato a Web-Services, .NET®, CORBA®, Enterprise JavaBeans™, C, C++ e Java™.

Benché l'uso corretto di MDA/UML ha dimostrato di facilitare la produzione di software di qualità, la sua applicabilità va valutata situazione per situazione: non tutte le applicazioni hanno lo stesso grado di complessità o presentano problematiche di integrazione/interoperabilità, mentre in alcune realtà

¹ C'è chi profetizza che il divario tra utente e programmatore sta via via diminuendo e che, con tutta probabilità, un giorno scomparirà.

aziendali potrebbe essere già consolidato l'uso di particolari altre tecniche e processi di sviluppo.

1.2 Obiettivi

Il presente lavoro di tesi si inserisce in un più ampio progetto di ricerca che mira ad estendere in ottica MDD il processo di sviluppo di applicazioni Web secondo la metodologia UWA/UWAT+. L'obiettivo ultimo di questo progetto è quello di sfruttare i vantaggi offerti dai processi di sviluppo Model-Driven da un lato per rendere più agevole l'impiego della metodologia UWA/UWAT+ e dall'altro per ottenere applicazioni Web più fedeli alle specifiche di progetto concettuale e quindi meglio rispondenti ai requisiti utente e più facilmente manutenibili ed evolvibili.

Nell'ambito di questo progetto di ricerca, in particolare, il presente lavoro di tesi si propone di definire e strutturare un livello di progettazione intermedio, che chiameremo livello logico, tra la progettazione concettuale UWA e l'implementazione, nonché di definire un processo e un tool di supporto per la traduzione semi-automatica di modelli concettuali UWA nei corrispondenti modelli logici.

Più in dettaglio, il lavoro di tesi si prefigge i seguenti obiettivi:

- 1) Scelta dell'architettura software da impiegare per l'implementazione delle applicazioni Web progettate e proposta di un modello logico per descrivere le specifiche di implementazione delle stesse secondo questa architettura.
- 2) Definizione del metamodello associato al modello logico definito al punto precedente.
- 3) Costruzione del metamodello UWA a partire dalla documentazione esistente sulla metodologia e da precedenti esperienze di ricerca in questa direzione.
- 4) Definizione di regole di trasformazione fra modelli concettuali UWA e modelli logici e implementazione di un tool di supporto per l'automazione delle trasformazioni stesse.

1.3 Percorso di lavoro

La metodologia di lavoro seguita per il raggiungimento degli obiettivi preposti è quella rappresentata in Figura 1.1, dove si rappresenta la suddivisione delle attività svolte in attività principali e di supporto.

Come si può osservare dallo schema, le fasi iniziali sono state più che altro di supporto per le successive, in quanto hanno consentito di acquisire il background di conoscenze necessarie per poter affrontare il lavoro proposto. Si è prestata particolare attenzione ai più noti paradigmi di modellazione di applicazioni Web esistenti in letteratura, ed alla metodologia UWA.

Basandosi sui risultati ottenuti nelle fasi di supporto si è potuto procedere con la parte centrale del lavoro. Stabilito l'utilizzo della notazione UML per la rappresentazione del nostro modello logico, è stato necessario individuare l'architettura di riferimento da utilizzare per la sua implementazione. La scelta è stata condotta sulla base dell'analisi di alcuni dei più noti pattern esistenti in letteratura, prestando particolare attenzione ai Web Presentation Pattern. Tale analisi ha portato alla scelta di MVC (Model-View-Controller) come pattern di riferimento. A questo punto si è stati in grado di definire il nostro modello logico, che abbiamo chiamato modello UML-MVC, attraverso la combinazione delle specifiche del pattern MVC con l'utilizzo di diagrammi UML stereotipati in grado di descrivere le componenti dell'applicazione da sviluppare in termini di class diagram (Model Class Diagram, View Class Diagram).

Il passo successivo è stato quello della definizione delle euristiche per la trasposizione della modellazione concettuale UWA verso il modello logico UML-MVC individuando per ogni primitiva di modellazione impiegato da UWA il corrispondente elemento all'interno del modello logico in maniera del tutto indipendente dalle possibili tecnologie di implementazione.

Fatto ciò, si è passati alla selezione di uno fra i possibili framework implementativi di MVC. L'analisi condotta ha portato alla scelta del framework JavaServer Faces (JSF) [JSF]. Una volta scelto il framework implementativo, si è reso necessario comprendere come distribuire i componenti del modello UML-MVC su tale tecnologia al fine di realizzare la loro implementazione. Questo ha portato all'ideazione di un insieme di raccomandazioni utili per il passaggio verso l'implementazione.

A questo punto si è preso in esame un caso di studio: “Post-it sharing”. Per tale caso di studio si è costruito prima lo schema concettuale UWA dell’applicazione e poi si è passati alla realizzazione del suo schema logico UML-MVC mediante le regole precedentemente definite. Il modello così ottenuto è stato infine implementato mediante il framework JSF. Ciò ha consentito di trarre le opportune conclusioni in merito alla validità ed all’applicabilità dell’approccio proposto.

Fatto ciò, si è passati alla definizione del metamodello UWA, sulla base dei concetti di modellazione definiti dalla metodologia, e del metamodello UML-MVC definito partendo dall’esperienza acquisita mediante il caso di studio “Post-it sharing” e di altri modelli logici realizzati come esempio. Si è scelto come formato di memorizzazione dei metamodelli quello Ecore gestito dal framework EMF (Eclipse Modeling Framework) di Eclipse.

La fase successiva è consistita nel raffinare ed implementare le regole di mapping tra il metamodello concettuale UWA ed il metamodello logico UML-MVC. Come linguaggio per l’implementazione di tali regole è stato scelto ATL (ATLAS Transformation Language) [ATL] ed il relativo tool ADT (Eclipse development tools for ATL) [ADT] che ad oggi sono senza dubbio gli strumenti migliori a supporto del Model Transformation.

Al tool ADT sono stati forniti in input il metamodello di partenza (metamodello UWA), il modello da trasformare (modello concettuale UWA ottenuto come istanza del rispettivo metamodello), il metamodello di destinazione (metamodello logico UML-MVC) e le regole di trasformazione implementate in ATL. In output è stato ottenuto il modello logico UML-MVC, istanza del rispettivo metamodello UML-MVC, e relativo al particolare modello concettuale UWA fornito in input.

E’ stato possibile quindi fare un confronto tra il modello logico prodotto manualmente per il caso di studio e quello ottenuto dal processo di trasformazione automatizzato. Ciò ha consentito di validare le regole di mapping definite fra i due metamodelli, nonché la loro implementazione mediante ADT.

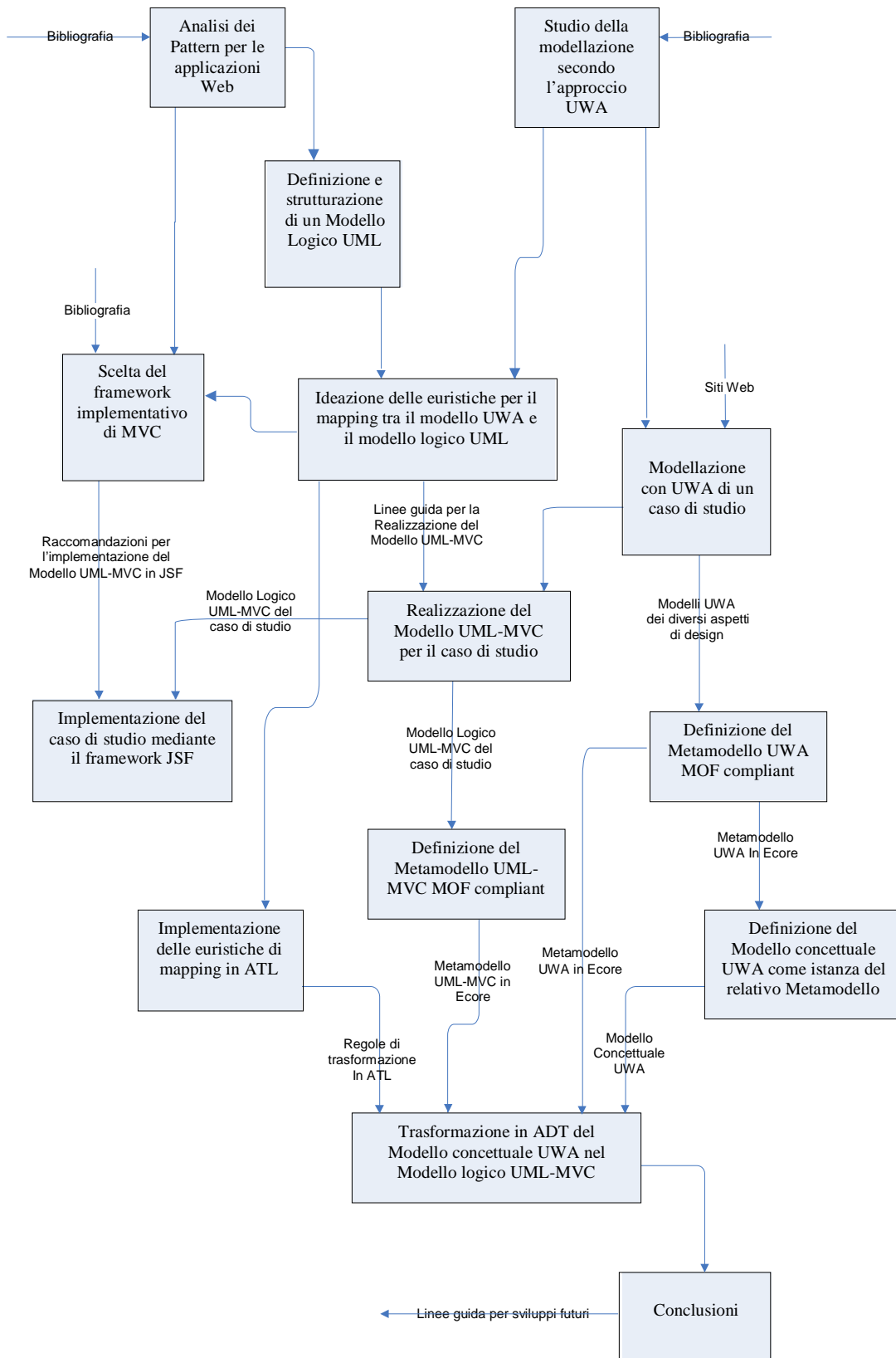


Figura 1.1: Metodologia di lavoro

In sintesi il percorso di lavoro si è sviluppato attraverso i seguenti passi:

- Analisi e scelta dell'architettura software da impiegare per l'implementazione delle applicazioni Web progettate. Qui la scelta è ricaduta sull'architettura MVC.
- Definizione manuale di un modello logico di una applicazione Web attraverso l'utilizzo di diagrammi UML stereotipati in grado di descrivere le componenti dell'applicazione da sviluppare in termini di class diagram seguendo l'approccio definito dal pattern MVC (*Model View Controller*).
- Definizione del metamodello UML-MVC in un profilo MOF sulla base del relativo modello definito manualmente per avere un riferimento preciso e rigoroso su come gli elementi del modello interagiscano fra di loro e per una loro comprensione più approfondita.
- Definizione del metamodello UWA in un profilo MOF sulla base dei concetti definiti dalla metodologia di modellazione UWA e su metamodelli già definiti in letteratura.
- Sulla base dei metamodelli definiti, ideazione delle euristiche per la trasposizione dalla modellazione concettuale UWA di un'applicazione Web nella sua modellazione logica UML-MVC, specificando per ogni primitiva di modellazione UWA l'elemento corrispondente nel modello logico.
- Implementazione delle euristiche individuate con l'ausilio dei linguaggi di trasformazione definiti in letteratura e di tools a supporto degli stessi in modo da automatizzare il processo di trasformazione di un modello concettuale UWA nel rispettivo modello logico UML-MVC.
- Sperimentazione della metodologia ideata attraverso la sua applicazione ad un caso di studio reale, al fine di validare e raffinare l'approccio proposto, fornendo prima un esempio completo di modellazione concettuale UWA e quindi di modellazione logica in grado di guidare l'implementazione.
- Traduzione del modello concettuale UWA prodotto, nel linguaggio scelto a supporto del Model Transformation in modo tale da automatizzare il

processo di trasformazione. Tale modello sarà definito come istanza del rispettivo metamodello in modo tale da garantire che sia ad esso conforme.

- Trasformazione del modello UWA , istanza del rispettivo metamodello, nel modello logico UML-MVC con l'ausilio di un tool a supporto del Model Transformation. Tale tool prenderà in input il metamodello UWA, il relativo modello concettuale UWA ed il metamodello UML-MVC e fornirà in output il corrispondente modello logico UML-MVC, istanza del metamodello UML-MVC, sulla base delle euristiche di trasformazione definite.
- Sperimentazione della metodologia proposta attraverso la implementazione di alcuni casi di studio.

1.4 Struttura della tesi

La tesi è organizzata in sette capitoli brevemente riassunti di seguito.

Nel presente Capitolo 1 si sono esposti il contesto in cui è inserito il lavoro di tesi, gli obiettivi preposti e la metodologia seguita per raggiungerli.

Nel Capitolo 2 si analizza lo stato dell'arte nel settore della progettazione e del Model Driven Development di applicazioni Web di ultima generazione. Si analizzano inoltre tecniche, tools e linguaggi a supporto del Model Transformation.

Nel Capitolo 3 si approfondisce il concetto di modellazione concettuale di un'applicazione Web, attraverso la descrizione della metodologia UWA e della sua estensione UWAT+. Questa analisi ha permesso di evidenziare le motivazioni che portano alla necessità di introdurre un livello ulteriore di design, detto design logico dell'applicazione. Viene quindi definito il metamodello UWA in un profilo MOF utilizzando gli strumenti a supporto del Model Driven Development.

Nel Capitolo 4 si approfondisce il concetto di modellazione logica individuando in UML lo strumento più adatto alla sua rappresentazione. Scopo del capitolo è

determinare l'architettura di riferimento da utilizzare per strutturare il modello logico. Lo studio condotto ha portato alla scelta del pattern MVC che permette la strutturazione di un'applicazione attraverso la suddivisione in componenti (detti Model, View e Controller) che garantiscono la separazione tra logica di business e logica di presentazione. Viene quindi definito manualmente un modello logico UML-MVC per una particolare applicazione Web di condivisione di note (Post-it sharing) e quindi di deriva da questo esempio e da altre esperienze di modellazione il metamodello UML-MVC, definendolo in un profilo MOF utilizzando gli strumenti a supporto del Model Driven Development.

Nel Capitolo 5 vengono illustrate le euristiche proposte per la definizione del modello logico UML-MVC di un'applicazione Web a partire dal suo modello concettuale UWA. Il modello così ottenuto risulta ancora essere indipendente dalla tecnologia di implementazione che si intende utilizzare per realizzare l'applicazione finale. Viene quindi illustrata l'implementazione delle euristiche individuate mediante il linguaggio ATL (ATLAS Transformation Language) [ATL] ed il tool ADT (Eclipse development tools for ATL) [ADT] e descritto il processo di automatizzazione della trasformazione di modelli.

Nel Capitolo 6 si sperimenta la metodologia di sviluppo Model-Driven risultante applicandola al caso di studio "Post-it sharing". Dopo avere realizzato il modello concettuale UWA dell'applicazione si è proceduto con la generazione del modello logico UML-MVC mediante il tool di trasformazione ADT le regole di mapping implementate in ATL. Il modello logico così ottenuto è stato confrontato con quello prodotto manualmente nel Capitolo 4. Il capitolo si chiude con l'implementazione attraverso la tecnologia JSF dell'applicazione progettata, la presentazione del prototipo realizzato e la descrizione degli strumenti utilizzati per l'implementazione stessa.

Il Capitolo 7 propone infine una sintesi del lavoro complessivo svolto e dei risultati ottenuti, nonché l'indicazione di possibili sviluppi futuri.

2. Stato dell’Arte

Nel presente capitolo si analizza lo stato dell’arte nel settore della progettazione e del Model Driven Development di applicazioni Web di ultima generazione e si descrive lo standard MDA di OMG. Si analizzano inoltre tecniche, tools e linguaggi a supporto del Model Transformation.

2.1 Model Driven Development

Lo sviluppo di software viene spesso paragonato, in termini di maturità, allo sviluppo e alla progettazione di Hardware. Mentre in questo ambito molti sono stati gli sviluppi negli ultimi 20 anni, non si può dire altrettanto nello sviluppo del software. Progressi nello sviluppo del software sono evidenti nella velocità con cui si possono sviluppare applicazioni mainframe, senza interfaccia grafica e non collegate ad altri sistemi, ma questo non è ciò che normalmente viene realizzato.

Nel software alcuni problemi di diversa natura hanno ancora bisogno di essere risolti. Programmare è un lavoro difficile e faticoso, per ogni nuova tecnologia spesso è necessario un lavoro di adattamento spesso anche ripetitivo. I sistemi non sono mai stati costruiti con una sola tecnologia e devono sempre essere collegati ad altri sistemi. C’è anche il problema del continuo cambiamento dei requisiti.

Il processo di sviluppo del software normalmente si articola nelle seguenti fasi:

- Definizione dei requisiti
- Specifica e Analisi Funzionale
- Design
- Coding
- Testing
- Deployment

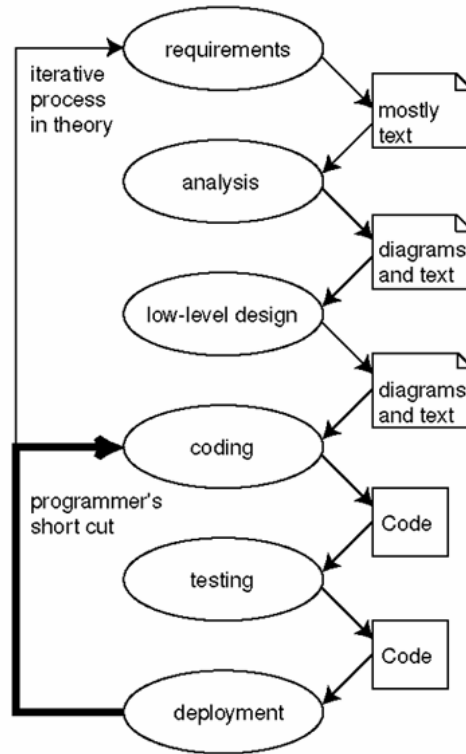


Figura 2.1 - Ciclo di vita del software

Nelle prime fasi si producono documenti. Relazioni, specifiche, diagrammi come use cases, class diagrams, interaction diagrams, activity diagrams, ecc... ma tutto questo rimane solo carta. Tutta la massa della documentazione è relativa alle prime 3 fasi del ciclo appena descritto.

Quando passiamo alle fasi successive, il valore proprio della documentazione diminuisce dato che man mano che la codifica e lo sviluppo avanzano il codice si distacca da ciò che i documenti stabiliscono e descrivono. Figure e diagrammi allora non rappresentano più le specifiche del sistema, ma alcune figure correlate.

Man mano poi che il sistema cambia, nel tempo, per adattamenti, nuove richieste, cambiamenti degli scenari, il codice viene ritoccato, mentre la documentazione tende a diventare disallineata dal codice, a diventare incompleta. E' come se il ciclo di vita fosse comandato dal codice, come se fosse questo il motore dell'ideazione e dell'evoluzione di un'applicazione.

Altre questioni riguardano problemi tipici dei sistemi software; fra questi sono inclusi la crescente complessità dei sistemi, il crescente bisogno di portabilità, il problema dell'interoperabilità.

Come risposta a tutta questa serie di questione si sono sviluppate diverse metodologie e soluzioni.

2.1.1 Model Driven Architecture (MDA)

L’Object Management Group (OMG) [OMG] è un organizzazione che contribuisce alla definizione di standard e di specifiche relative ai linguaggi di specifica del software. La Model Driven Architecture (MDA) [MDA] è un framework per lo sviluppo di applicazioni. Per MDA il concetto chiave è la modellazione. Il processo di sviluppo del software in MDA è descritto e portato avanti grazie a modellizzazioni del proprio sistema software.

MDA prevede una serie di modelli che descrivono il sistema e il loro utilizzo in un ciclo di vita del software MDA, non troppo dissimile da quello standard:

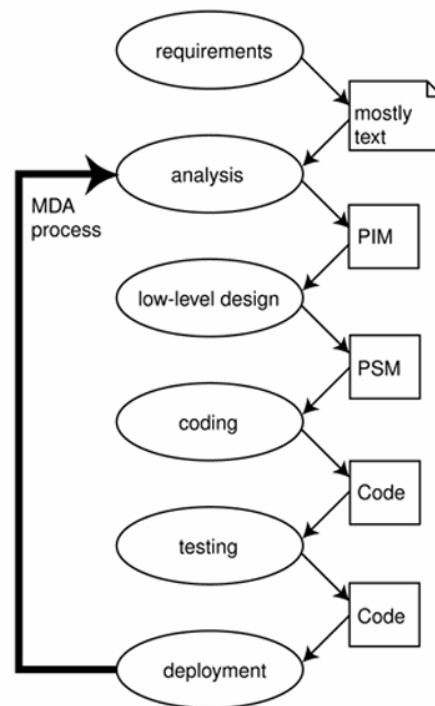


Figura 2.2 - Ciclo di vita del software in MDA