Prefazione

Adam Bosworth
Capo Progettista e Vice Presidente Senior di Engineering, BEA

Inizio la stesura della prefazione di questo libro confessando un grande senso di inadeguatezza. Non sono certamente un accademico, anzi, ho sviluppato le mie conoscenze e abilità in ambito industriale e mi piace ricordare che sono un semplice ragazzo della campagna del Vermont. Gli autori di questo libro sono brillanti, con una profonda cultura cosmopolita, formatasi a Milano così come a Stanford. Ho incontrato per la prima volta gli autori nel 2000, quando ero nel mezzo di una lotta per fondare una nuova azienda; vennero a presentarmi un modello molto semplice ed elegante per la progettazione di applicazioni basate sui dati. E mentre vedevo questa proposta spalancarsi di fronte a me, tutta la mia vita professionale scorreva davanti ai miei occhi. Ho speso un terzo di quella vita aiutando a costruire basi di dati relazionali e strumenti per utilizzarle (OLAP, Reflex, Access, ODBC, SQL Server, Data Access Objects), un altro terzo contribuendo alla costruzione di interfacce utente e strumenti per realizzare applicazioni (VB, controlli OLE, Form Designers, Quattro) e l'ultimo e più recente terzo costruendo PLUMBING e strumenti per usare il Web e costruire applicazioni per il Web (motore HTML di Internet Explorer, Active Server Pages e WebLogic Workshop). Ed ecco lì, costruita da un gruppo di professori universitari: una sintesi di tutti questi elementi che riduceva il problema a qualcosa di semplice ed elegante come il calcolo relazionale. Ero contemporaneamente deliziato e sofferente.

Gli autori si erano fatti carico di una enorme sfida nella realizzazione di WebML. Puntavano a creare un linguaggio formale che potesse descrivere qualunque interfaccia utente che potesse avere un senso per percorrere, visualizzare e modificare i dati. Qualcuno potrebbe sorprendersi del fatto che non ho limitato la mia frase precedente ad affermare che "potesse avere senso per il Web", ma di fatto credo che l'obiettivo fosse realizzare un formalismo per *tutte* le interfacce utente centrate sui dati. Questo era, e rimane, un obiettivo audace. Se avesse successo, potrebbe avere sulle interfacce utente centrate sui dati lo stesso effetto che il calcolo relazionale ha avuto sull'accesso ai dati. In un'epoca in cui le interfacce utente dovranno affrontare la loro più grande sfida, cioè essere in grado di modificarsi per adattarsi a differenti formati e paradigmi, questo approccio è perfettamente calzante.

In aggiunta, ogni capitolo di questo libro spiega con cura il modo in cui il formalismo così ideato può essere codificato in UML; ciò fa sì che possa essere

sviluppato un grande insieme di strumenti software di sviluppo, che possono collaborare ad alto livello. Il potenziale per la produttività, se questo si verificasse, sarebbe enorme. Se alla standardizzazione si aggiunge il fatto che tutte le "operation unit" possono essere definite come Web service, diviene immediatamente possibile disporre di siti Web realmente portabili e interoperabili, in cui diversi gruppi di sviluppatori possono cooperare in modo armonico, senza conflitti e con elevata produttività. In breve, sono molto entusiasmato da questo potenziale.

Dunque, come hanno fatto?

Molti problemi che noi (nell'industria) non abbiamo mai ben risolto nei prodotti sviluppati fino ad ora (per esempio, costruire liste per la selezione di oggetti da visualizzare), sebbene avessimo a disposizione la ricchezza di un ambiente applicativo basato su interfaccia grafica (GUI), sono stati brillantemente risolti attraverso le "Index Unit". Prodotti come Access, PowerBuilder e Delphi avevano intuito l'idea di tali mattoni elementari (tipicamente chiamati *form* in modo poco elegante e noti come *unit* in WebML), ma non formalizzarono mai tutti i possibili modi in cui i collegamenti tra le unit potevano verificarsi.

Un altro problema che ha messo in ginocchio il mondo delle applicazioni centrate sui dati è stato quello dei parametri opzionali. Se l'utente specifica una condizione di selezione dei dati, come un intervallo di prezzi per un prodotto, questo dovrebbe essere usato per filtrare l'insieme dei risultati. Ma se non viene specificata nessuna condizione, dovrebbero essere mostrati tutti i prodotti, qualunque sia il loro prezzo. Gli utenti hanno dovuto passare le pene dell'inferno costruendo le complesse logiche condizionali in SQL per gestire i casi di parametri che possono essere nulli (nel qual caso nessun predicato deve essere applicato) o non nulli (nel qual caso bisogna introdurre il predicato, spesso complicato dalla presenza di diverse condizioni). Il concetto WebML di "predicato opzionale", specificato semplicemente con la parola chiave "implied", esonera lo sviluppatore dall'accollarsi una notevole fatica.

Probabilmente il più grande problema mai risolto completamente nel mondo di VB, Access, PowerBuilder e Delphi è quello del contesto. Nel mondo delle applicazioni reali, gli utenti si aspettano che l'applicazione sia in grado di comprendere il contesto nel quale stanno lavorando. Per esempio, giungere alla pagina che elenca gli impiegati, avendo prima attraversato la pagina di un dipartimento e poi la pagina di una sezione del dipartimento, significherà tipicamente poter vedere l'elenco degli impiegati di quel dipartimento e di quella specifica sezione. D'altra parte, navigare da una pagina di musicisti a una pagina di titoli di pezzi musicali, per poi giungere alla pagina di un impiegato, significherà tipicamente visualizzare l'insieme di musicisti per un certo titolo. Semplice, vero? Ovvio? In realtà, è piuttosto complicato da realizzare in un mondo "normalizzato" come il nostro. In WebML questo è risolto semplicemente ed elegantemente facendo uso di selettori parametrici, link contestuali, parametri globali e, occasionalmente, predicati opzionali o "impliciti". L'idea di consentire il passaggio di valori multipli attraverso parametri è squisitamente elegante. I meccanismi di default sono ragionevoli e intuitivi e consentono di risparmiare una grande mole di lavoro. Leggendo con attenzione la sezione 3.4 vi renderete immediatamente conto di quanto lavoro potete risparmiare.

Tutto questo ricorda una verità fondamentale a noi che lavoriamo nell'industria. Ci conviene lavorare costantemente in collaborazione con chi lavora in ambito accademico, perché i risultati avranno un livello di chiarezza, formalismo ed eleganza senza i quali le soluzioni spesso sembrano degli accrocchi che alla fine risultano insoddisfacenti. Come "padre" di Access, ero, e sono, sconcertato.

Dopo aver raggiunto l'obiettivo nell'area delle interfacce utente centrate sui dati in generale, con i concetti di unit e link, WebML non riposa sugli allori. Procede considerando in profondità i paradigmi di interfacce utente per il Web e fornisce concetti per i componenti di più alto livello, le pagine (inclusi i concetti di *home*, *default* e *landmark*) e le aree. Così come per i formalismi per la navigazione attraverso i dati, questi formalismi aprono un grande spettro di possibilità e risparmiano una grande quantità di lavoro agli sviluppatori. Per esempio, la tipica soluzione industriale attualmente proposta per includere sempre un link alla "home page" consiste nell'usare un *template* per tutte le pagine. Ma spesso il modello è annidato in modo gerarchico e in questo caso il modello a template crolla, perché è troppo statico. WebML fornisce invece un modello molto elegante per includere questi link navigazionali senza nessun doloroso appesantimento del lavoro del programmatore. La stessa cosa vale per le pagine annidate, che mostrano chiaramente la ricchezza delle interfacce utente che possono essere assemblate in modo molto semplice, come quando da piccoli eravamo abituati a montare una costruzione di mattoncini. Questo è, ovviamente, il sogno di qualsiasi sviluppatore. Uno degli aspetti più interessanti di questo modello è che lo sviluppatore può sempre decidere in modo molto chiaro quanto dovrebbe essere fatto in una pagina (risparmiando così passi di navigazione) e quanto dovrebbe essere spezzato nelle varie pagine. Gli autori hanno pensato in modo molto attento alle operazioni intra-pagina, non solo a quelle inter-pagina, e tutto ciò è materia per la buona progettazione di siti Web e interfacce utente.

Cosa dire a proposito del realizzare concretamente un progetto? Questo modello descrive le azioni che devono essere svolte? E' possibile sviluppare applicazioni per il mondo reale? Storicamente, il formalismo per tutto ciò è stato il codice di programmazione, o le "azioni predefinite" nei nostri componenti per la definizione delle interfacce utente. E' sempre stato difficile capire le azioni che dovevano essere svolte nel contesto di una "coreografia". Per esempio, cosa succede se l'immissione dei dati dell'utente fallisce oppure ha successo? Un paio di concetti, operazioni con link per il successo (OK) e il fallimento (KO) - che mostrano o che gli autori sono appassionati di pugilato o che hanno senso dell'umorismo -, sono stati usati per un incredibile numero di azioni: inserimento, cancellazione, modifica, chiamata di codice generico con il trasporto dell'intero contesto. La mia azienda, BEA, è particolarmente interessata da quest'idea in quanto è parallela ad alcune attività che abbiamo svolto riguardo alla navigazione delle pagine per aiutare i nostri clienti a lavorare meglio sulla struttura e sulla gestione dei siti Web; i primi riscontri di questo lavoro sembrano molto promettenti. Questo modello che consente di alternare operazioni ed elementi di interfaccia utente è molto potente. Se poi si aggiunge un modello per accedere ai dati in ingresso e per pubblicare i risultati (ruoli per i quali credo che i Web Services

si riveleranno perfettamente adatti)-e *voilà!*- ecco l'estendibilità istantanea. Nello stesso tempo, le operazioni predefinite di base abilitano la realizzazione di un enorme quantità di lavoro standard con interoperabilità e portabilità totale che, in presenza di strumenti appropriati, potrà essere svolto dallo stesso tipo di utenti che oggi usano Access.

WebML può essere visto anche come un foglio elettronico per la definizione di interfacce utente. Essenzialmente, il ricalcolo viene attivato ogni volta che un dato in ingresso alla pagina cambia. Infatti, come avviene per i fogli elettronici, questo processo può essere non deterministico e addirittura circolare. Alcuni anni fa ho costruito un foglio elettronico chiamato Quattro. Come per tutti questi prodotti, consisteva in un motore di calcolo complesso, progettato per associare tutte le espressioni al loro "valore corretto", se possibile. L'interazione tra questo modello dichiarativo di costruzione di applicazioni (così chiaramente comprensibile da decine o centinaia di milioni di utenti) e quello della logica procedurale (ugualmente necessario, ma comprensibile solo da un milione di programmatori) mi ha sempre affascinato. Di fatto, sarebbe veramente un grande risultato se WebML riuscisse a mettere la costruzione di interfacce utente centrate sui dati alla portata dello stesso numero di persone che hanno imparato a far uso dei fogli elettronici.

Un altro modo, molto diverso, per pensare a WebML è come a una specie di super-workflow. Concettualmente, queste "operazioni", le sequenze transazionali e i workflow possono essere fusi insieme per costituire una visione globale di siti e workflow. È come se i mondi della descrizione dei siti, della strutturazione delle pagine e dei processi di *business* venissero fusi e semplificati senza discrepanze in un unico processo. Molto tempo fa, un'azienda chiamata Metaphor intraprese questa strada con un prodotto chiamato Capsules. Quello che qui è stato costruito è una serie di linee guida per completare quel viaggio, almeno nel contesto della costruzione di interfacce utente basate sui dati.

Ci sono ancora alcune questioni complesse che restano irrisolte:

- Come modellare un'interfaccia utente che rifletta la storia? Se, ad esempio, volessi che le ancore di navigazione comparissero solo man mano che l'utente le percorre, come capita in molti wizard, come potrei specificarlo in WebML? Gran parte di questo problema può essere gestito attraverso i parametri globali, ma la visualizzazione condizionata degli elementi navigazionali non sembra essere coperta.
- Come interagisce con questo modello il mondo dell'asincronismo? C'è un semplice esempio dell'addebito su carta di credito nel Capitolo 4 che assume sia sufficiente semplicemente bloccare l'applicazione e attendere l'accettazione o il rifiuto prima di passare alla pagina successiva; tuttavia, sappiamo che questo non è sempre vero.
- Come si modifica l'interfaccia utente per riflettere i vari ruoli che l'utente può assumere? La metodologia descritta nel Capitolo 7 e la Sezione 9.4.4 sembrano ipotizzare che ci sia una differente mappa di sito per ogni possibile classe di utenti. In pratica, questo diviene spesso ingestibile e le pagine dovrebbero modificarsi per riflettere i diritti dell'utente; ciò significa che in alcuni casi i dati e i link saranno visibili o meno e i dati saranno modificabili o meno. Per essere

chiari, il modello supporta la personalizzazione, ma non la modifica dinamica della struttura di sito basata sul ruolo degli utenti.

- In un mondo in cui le pagine saranno sempre più legate ai dati applicativi attraverso Web services invece che direttamente ai dati attraverso SQL, come si colloca questo modello? Quali cambiamenti sono richiesti?
- Così come brillantemente mostrato da Google, spesso il modo migliore per creare una index unit è attraverso una semplice ricerca effettuata su testo. Questo modello non può essere espresso in modo ragionevole in una base di dati relazionale. Questa non è una limitazione di WebML, ma così come i progettisti di basi di dati si sono resi conto di colpo di questa necessità ovvia e hanno iniziato a incorporarla come una funzionalità essenziale, anche i predicati WebML dovranno essere estesi per descrivere questa modalità per la costruzione di insiemi di dati.
- Gli utenti utilizzeranno realmente la metodologia di progettazione formale attentamente descritta dai Capitoli 7 - 9? Non è chiaro. Gli utenti con la grande frustrazione e disperazione dei tecnici informatici hanno spesso la tendenza all'implementazione bottom-up piuttosto che alla progettazione top-down. Certo è che, alla fine, i progettisti hanno imparato la progettazione dei dati e i diagrammi E-R e UML sono diventati uno strumento fondamentale per molte aziende di grandi dimensioni. È dunque possibile che vedremo la stessa cosa in questo caso.

Soprattutto, WebML è un risultato audace e impressionante. Riunisce insieme in modo elegante e sembra in grado di costruire qualunque cosa attraverso l'uso appropriato della composizione di pochi mattoni elementari. È impressionante come WebML disponga sia di un linguaggio testuale, sia di un modello di sviluppo visuale. Le considerazioni precedenti servono appena a stuzzicare l'appetito e suggeriscono che questo modello si dimostrerà davvero utile.

Sebbene io sospetti che l'avvento dei Web services modificherà questo modello in un modo sottile ma significativo, WebML resta comunque una delle direzioni più promettenti che abbia mai visto. Potrà svolgere, per la costruzione di applicazioni, il ruolo che SQL e ODBC/JDBC hanno svolto per l'accesso ai dati, o il ruolo che i Web services stanno svolgendo per la comunicazione tra applicazioni, fornendo un modello stabile e basato su standard, che potrà rivoluzionare in modo definitivo lo sviluppo industriale e incrementare di un ordine di grandezza o più il numero di persone che possono svolgere questo compito. Bravi!

Introduzione

Questo libro descrive metodologie e linguaggi per la progettazione di applicazioni Web basate sui dati. Con questo termine ci riferiamo a siti Web che accedono e gestiscono grandi quantità di dati strutturati, tipicamente memorizzati come tuple all'interno di sistemi di gestione di basi di dati. Oggi le applicazioni Web basate sui dati sono il tipo di applicazioni più diffuse sul Web; siti per il commercio elettronico, siti istituzionali di organizzazioni private e pubbliche, librerie digitali, portali di grandi multinazionali, siti di comunità virtuali sono tutti esempi di applicazioni Web basate sui dati.

Lo sviluppo di un'applicazione Web di questo tipo è un'attività multidisciplinare, che coinvolge un certo numero di competenze differenti, necessarie per affrontare compiti molto eterogenei, come la progettazione della struttura dati per memorizzare i contenuti, l'ideazione di interfacce ipertestuali per la navigazione e la gestione delle informazioni, la creazione di stili grafici per le applicazioni, la costruzione di architetture robuste e con buone prestazioni e l'integrazione con applicazioni preesistenti e con servizi esterni. Lo sviluppo e la manutenzione di applicazioni Web basate sui dati necessita di tutti gli strumenti e le tecniche dell'ingegneria del software, compresa la definizione di un processo di sviluppo del software ben organizzato, concetti di progettazione e notazioni appropriati e linee guida per la conduzione delle varie attività.

Se si osservano il modo in cui le applicazioni Web sono sviluppate al giorno d'oggi e gli strumenti disponibili per gli sviluppatori, ci si rende conto che i principi e le intuizioni dell'ingegneria del software non sono sfruttati al meglio. Infatti, spesso i progettisti costruiscono le applicazioni Web applicando metodi e soluzioni che hanno appreso sviluppando altri tipi di sistemi software, come sistemi informativi aziendali e applicazioni orientate agli oggetti. Questo approccio funziona bene per lo sviluppo della parte "convenzionale" dell'applicazione Web, come la struttura dei dati e la logica di business che stanno dietro l'applicazione, ma non affronta la specificità di un'applicazione "Web", che consiste nella pubblicazione di contenuti e servizi facendo uso di un'interfaccia ipertestuale. Questa carenza è particolarmente evidente nei concetti e nelle notazioni di progettazione: quando si tratta di specificare l'interfaccia ipertestuale della propria applicazione, i progettisti fanno ricorso a strumenti piuttosto rudimentali, come carta e matita o anteprime di pagine HTML. Questa situazione, che spesso si verifica anche in grandi organizzazioni ben fornite di strumenti per l'ingegnerizzazione del software, richiede un adattamento del processo di sviluppo che sia in grado di affrontare le caratteristiche peculiari delle applicazioni Web. Il ciclo di vita delle applicazioni Web dovrebbe essere costruito attorno a un solido nucleo di concetti

e notazioni centrate sugli aspetti Web e supportato da specifiche linee guida per la messa in opera di tali concetti.

Il contributo di questo libro consiste nella proposta di un insieme di concetti, notazioni e tecniche per la costruzione di applicazioni Web basate sui dati, che possono essere usati dai gruppi di sviluppo di applicazioni Web per supportare tutte le attività del ciclo di vita dell'applicazione, dall'analisi al rilascio e all'evoluzione.

L'insieme proposto unisce ingredienti tradizionali ben noti agli sviluppatori, come la progettazione concettuale dei dati attraverso il modello Entità-Relazione e la specifica dei casi d'uso attraverso UML, con nuovi concetti e metodi per la progettazione degli ipertesti, che sono essenziali per lo sviluppo di applicazioni Web. Tuttavia, il valore dell'approccio proposto non sta nei singoli ingredienti, ma nella definizione di un processo sistematico, nel quale le attività di sviluppo possono essere organizzate in base ai principi fondamentali dell'ingegneria del software, e tutti i compiti, inclusi quelli più centrati sul Web, sono adeguatamente supportati da concetti, notazioni e tecniche appropriati.

La caratteristica che distingue il processo di sviluppo proposto è l'enfasi posta sulla modellazione concettuale. La modellazione concettuale si è dimostrata efficace in molti campi dello sviluppo di software: nella progettazione delle basi di dati, dove il modello Entità-Relazione costituisce una notazione di alto livello ed efficace per la comunicazione dei requisiti dei dati tra progettisti e personale non tecnico, ed è il punto di partenza per la creazione di schemi di basi di dati di buona qualità; nelle applicazioni orientate agli oggetti, in cui le notazioni come UML (Unified Modeling Language) hanno innalzato in modo notevole il livello su cui i progettisti documentano e discutono i loro lavori. La nostra tesi è che questi benefici dovrebbero essere sfruttati anche per la progettazione di applicazioni Web basate sui dati, che dovrebbero essere specificate attraverso notazioni visuali di alto livello, intuitive, facilmente comunicabili a utenti non tecnici e utili per gli sviluppatori delle applicazioni.

Pertanto, questo libro propone un linguaggio di modellazione di alto livello per la specifica di applicazioni Web, chiamato Web Modeling Language (WebML). Essenzialmente, WebML consiste in un insieme di concetti visuali per descrivere un ipertesto come un insieme di pagine costituite da unit di contenuto e operazioni collegate tra loro e associate con i dati ai quali si riferiscono.

WebML segue lo stile di linguaggi di modellazione noti, come il modello Entità-Relazione e UML: ogni concetto ha una rappresentazione grafica e la specifica avviene attraverso diagrammi visuali. Il lettore non deve pertanto preoccuparsi della necessità di apprendere un nuovo linguaggio. Così come per i costrutti del modello Entità-Relazione, anche i diagrammi WebML possono essere rappresentati attraverso la sintassi UML, forse con minor chiarezza ed immediatezza, ma senza perdite di potere espressivo.

Tuttavia, facciamo presente che i concetti sono più importanti delle notazioni e che i metodi per applicare i concetti sono ancor più importanti; il libro guiderà quindi il lettore sia nell'apprendimento dei concetti di modellazione richiesti (Entità-Relazione e WebML), sia nell'applicazione di tali concetti alla specifica

e alla progettazione di un'applicazione Web, attraverso fasi come la raccolta dei requisiti, la progettazione dei dati e la progettazione dell'ipertesto. In aggiunta, nonostante il grande peso dato alla modellazione concettuale, una parte del libro si focalizzerà sui numerosi problemi di implementazione e rilascio di applicazioni Web basate sui dati. In particolare, il primo capitolo e l'ultima parte del libro sono interamente dedicati alle questioni tecnologiche e mostrano al lettore interessato come trasformare il progetto concettuale di un'applicazione Web in componenti software funzionanti sulle tecnologie delle basi di dati attuali. Gli aspetti considerati includono HTTP, HTML, XML, XSL, basi di dati relazionali e SQL, linguaggi di scripting a lato server e librerie di tag, application server e architetture di cache.

Come ultimo aspetto, il libro termina menzionando uno strumento CASE (Computer Aided Software Engineering) che supporta il ciclo di vita proposto; infatti i benefici della modellazione concettuale e di un processo di sviluppo strutturato si moltiplicano se si fa uso di strumenti software adeguati. Tutte le notazioni proposte si adattano perfettamente agli strumenti software noti agli sviluppatori, come editor di modelli Entità-Relazione e UML e generatori di codice. In particolare, WebML può essere facilmente supportato rappresentando i diagrammi WebML in UML oppure realizzando strumenti appositamente pensati per la notazione WebML, come nel caso del prodotto presentato nell'ultimo capitolo del libro.

Organizzazione del libro

Il libro è strutturato in quattro parti. La prima parte introduce il contesto tecnologico in cui si colloca lo sviluppo Web; la seconda parte presenta i linguaggi di modellazione usati nel libro, il modello Entità-Relazione e WebML; la terza parte definisce il processo di sviluppo del software; la quarta parte si focalizza sull'implementazione di applicazioni Web basate sui dati, presentando soluzioni per le attuali architetture per il Web.

Tutti i capitoli hanno una struttura regolare, con un'introduzione che annuncia il problema trattato nel capitolo, una parte centrale che descrive la soluzione proposta e una conclusione che riassume i risultati presentati. Nei capitoli dedicati al processo di sviluppo, le fasi di progettazione sono applicate a un caso applicativo, che viene progressivamente sviluppato dall'analisi dei requisiti all'implementazione.

La Parte I, che comprende il Capitolo 1, riassume le tecnologie rilevanti per lo sviluppo di applicazioni Web basate sui dati.

Il Capitolo 1 contiene un'ampia descrizione delle tecnologie fondamentali impiegate nella costruzione di applicazioni Web centrate sui dati. Il capitolo illustra brevemente il protocollo e i linguaggi fondamentali per il Web (HTTP, HTML, scripting e componenti a lato client); si focalizza inoltre su XML, il nuovo paradigma per la strutturazione e lo scambio delle informazioni, e sui suoi standard collaterali per la trasformazione dei documenti (XSL e XQuery); presenta poi il secondo ingrediente delle applicazioni Web basate sui dati: le basi di dati relazionali, con il relativo linguaggio di interrogazione (SQL) e gli standard di

interoperabilità (ODBC e JDBC). In conclusione, descrive le architetture e i linguaggi per la costruzione di pagine Web dinamiche, come Java servlet, linguaggi di scripting a lato server come ASP e JSP, librerie di tag e architetture basate su application server. Il capitolo termina con la discussione dei meccanismi di pubblicazione di contenuti su differenti dispositivi.

La Parte II, che comprende i Capitoli 2-5, è dedicata alla presentazione dei linguaggi di modellazione usati nel libro.

Il Capitolo 2 descrive le primitive del linguaggio di modellazione dei dati Entità-Relazione. Gli elementi fondamentali del modello dei dati sono *entità*, definite come contenitori di dati, e *relazioni*, definite come associazioni semantiche tra entità. Le entità hanno delle proprietà chiamate attributi, descritte da un nome e da un tipo. Le entità possono essere organizzate in gerarchie di generalizzazione e le relazioni possono essere ristrette per mezzo di vincoli sulla cardinalità. Il capitolo mostra anche come specificare attributi e relazioni il cui contenuto può essere determinato da altri elementi informativi, scrivendo delle espressioni dichiarative nel linguaggio OCL (Object Constraint Language).

Il Capitolo 3 descrive WebML, il linguaggio di modellazione di ipertesti basato sui concetti di unit, pagine e link. Le *unit* costituiscono le unità informative elementari che possono essere pubblicate, le *pagine* indicano come le unit devono essere composte tra loro e i *link* descrivono le connessioni tra unit e/o pagine. Diversi ipertesti, chiamati *site view*, possono essere definiti sugli stessi contenuti, allo scopo di mostrare diverse viste sui dati indirizzate a diversi tipi di utenti. Le primitive di modellazione sono introdotte gradualmente, facendo uso di molti esempi ispirati a configurazioni di ipertesti usate frequentemente.

Il Capitolo 4 descrive l'estensione del modello di ipertesto per supportare le funzioni di gestione dei contenuti, come la modifica delle informazioni personali, il riempimento del carrello della spesa e così via. Vengono introdotti nuovi costrutti per rappresentare operazioni, che possono essere predefinite o generiche. Le operazioni predefinite rappresentano le tipiche funzioni di gestione dei contenuti e di utilità normalmente presenti nei siti Web, come la creazione, la cancellazione e la modifica degli oggetti, le operazioni di login e logout dell'utente e l'invio di messaggi di posta elettronica. Le operazioni generiche rappresentano invece funzioni "a scatola chiusa" e consentono l'integrazione delle applicazioni WebML con servizi esterni.

Il Capitolo 5 si occupa di chiarire il significato di ipertesti con struttura arbitraria in termini di pagine, unit e link. Il capitolo presenta inoltre un algoritmo di alto livello, semplice ma completo, per il calcolo dei contenuti delle pagine ipertestuali; ciò consente di evidenziare la semantica operazionale di WebML e prepara la strada per la discussione sull'implementazione dei costrutti ipertestuali, che è l'argomento della Parte IV del libro.

La Parte III, comprendente i Capitoli 6-9, presenta il processo di sviluppo delle applicazioni Web basate sui dati.

Il Capitolo 6 è un'introduzione al ciclo di vita dell'applicazione. Discute le attività di specifica, progettazione e implementazione, richieste per costruire un'applicazione Web centrata sui dati. Di ogni fase vengono brevemente descritti

gli obiettivi e le azioni da svolgere.
Il Capitolo 7 si focalizza sull'analisi dei requisiti, un'attività dedicata alla raccolta e alla specifica dei requisiti dell'applicazione, preliminare alle fasi di modellazione e progettazione. La raccolta dei requisiti comprende l'identificazione di utenti e gruppi, la definizione dei requisiti funzionali, dei dati e di personalizzazione, oltre ai requisiti non funzionali riguardanti presentazione, usabilità, prestazioni, disponibilità, scalabilità, sicurezza e manutenibilità. I requisiti funzionali sono formalizzati per mezzo dei diagrammi dei casi d'uso UML (*use case diagram*; i concetti fondamentali e le site view sono descritti per mezzo di un dizionario dei dati e delle mappe di sito; infine, le linee guida per lo stile grafico sono espresse nella forma di *mock-up* di pagine HTML.

Il Capitolo 8 affronta l'attività di progettazione dei dati, tenendo conto delle particolari sfumature che questa fase assume nel contesto della progettazione Web. La struttura dati di un'applicazione Web spesso presenta un'organizzazione regolare, in cui è possibile identificare diversi sotto-schemi tra loro interconnessi; ciascuno di essi risulta essere basato su un'entità "core", che rappresenta un oggetto fondamentale delle applicazioni. Come conseguenza, anche il processo di progettazione assume una forma regolare: parte dalla specifica degli oggetti core, che costituiscono lo scheletro essenziale dello schema dei dati, e procede in modo iterativo aggiungendo quattro tipi di sottoschemi, che rappresentano i componenti interni dei concetti core, le interconnessioni per supportare la navigazione, gli oggetti ausiliari, per facilitare l'accesso agli oggetti core, e gli oggetti per il supporto della personalizzazione.

Il Capitolo 9 descrive le attività di progettazione dell'ipertesto. La progettazione procede in modo *top-down*: inizialmente si realizza una bozza di schema di ipertesto, ottenuta partizionando in aree ogni siteview identificata nel corso dell'analisi dei requisiti e assegnando ad ogni area un insieme di funzioni, che supportano la navigazione degli oggetti core, di accesso o di interconnessione, oltre alle operazioni di gestione dei contenuti. In seguito, lo schema iniziale di ogni area viene raffinato, dando origine allo schema dettagliato, specificato in WebML; in questa fase il progettista stabilisce concretamente quali unit, link, operazioni e pagine costituiranno ogni siteview. La progettazione dell'ipertesto è facilitata dall'uso dei pattern di progettazione, che offrono soluzioni collaudate per le più tipiche configurazioni di pagina richieste.

La Parte IV, costituita dai Capitoli 10-14, è dedicata all'implementazione e al rilascio delle applicazioni Web basate sui dati.

Il Capitolo 10 si concentra sulla progettazione dell'architettura e precede la discussione dell'implementazione vera e propria. Questo capitolo illustra le architetture di riferimento che possono essere usate per la costruzione di applicazioni Web basate sui dati e fornisce alcuni criteri per scegliere tra le varie alternative. Il capitolo affronta in dettaglio i requisiti non funzionali di prestazioni, sicurezza, disponibilità e scalabilità, ed evidenzia le scelte di progettazione e i compromessi che devono essere affrontati per garantire il livello di servizio richiesto. Il capitolo si chiude con una sezione dedicata alla valutazione delle prestazioni e al caching, due aspetti importanti della progettazione di architetture Web.

Il Capitolo 11 tratta il tema dell'implementazione degli schemi concettuali

dei dati in sorgenti dati fisiche. Vengono presentati alcuni scenari differenti, con un diverso livello di riuso dei dati e degli schemi preesistenti. Il capitolo inizia presentando le regole standard di mapping per trasformare un diagramma Entità-Relazione in uno schema di base di dati relazionale. Successivamente, affronta l'implementazione dello schema relazionale nel contesto di un'infrastruttura dati aziendale, compito che presenta molte scelte progettuali relative ai problemi di integrazione di schemi, integrazione dei dati e gestione della replicazione.

Il Capitolo 12 descrive come implementare le pagine WebML in programmi a lato server. Come riferimento, la spiegazione adotta il linguaggio di scripting JSP (Java Server Pages) e l'interfaccia di connessione alla base di dati JDBC, ma la discussione può essere facilmente adattata ad altre piattaforme, come l'architettura Microsoft .NET o il linguaggio PHP. La spiegazione delle tecniche di implementazione parte da configurazioni di pagina molto semplici, che corrispondono a template di pagina JSP abbastanza immediati, procedendo poi a coprire un ampio spettro di caratteristiche di pagine ipertestuali dinamiche.

Il Capitolo 13 presenta una strategia di implementazione più sofisticata, basata sul pattern di progettazione MVC (Modello-Vista-Controllore), che garantisce una distribuzione bilanciata delle responsabilità tra i componenti software che collaborano alla costruzione della pagina. In aggiunta, il capitolo illustra alcune tecniche di implementazione pensate per applicazioni di grandi dimensioni, come per esempio la definizione di servizi di unit e di operazioni generici che fanno uso di descrittori XML, lo sviluppo di oggetti di business distribuiti secondo lo standard Enterprise JavaBeans e la gestione centralizzata della presentazione con l'aiuto di regole CSS e XSL.

Infine, il Capitolo 14 descrive un esempio di strumento CASE, chiamato WebRatio Site Development Studio, che supporta la progettazione di applicazioni Web basate sui dati e la generazione automatica di codice a partire dalle specifiche del modello Entità-Relazione e WebML. Il capitolo illustra l'architettura e le funzioni dello strumento, che copre il ciclo di sviluppo dalla progettazione dei dati e dell'ipertesto fino alla loro implementazione. Le note bibliografiche forniscono una serie di riferimenti ad altri strumenti che supportano la specifica e la produzione di applicazioni Web.

Il libro è corredato da diverse appendici, che riassumono gli elementi del modello WebML, la sintassi di WebML e di OCL (Object Costraint Language), e le tecniche di implementazione per trasformare le specifiche di ipertesto in pagine dinamiche e interrogazioni alla base di dati.

A chi è rivolto il libro

Questo libro ha l'ambizioso obiettivo di proporre un innalzamento del livello nella modalità in cui le applicazioni Web sono sviluppate, seguendo la tradizione della modellazione concettuale e dell'ingegneria del software. Il libro non è diretto soltanto agli specialisti delle tecnologie informatiche, ma anche a tutti i professionisti coinvolti nella costruzione di un'applicazione Web, che costituiscono un

pubblico tanto ampio quanto lo spettro di problemi affrontati dagli sviluppatori di applicazioni Web.
Per raggiungere questo obiettivo, ci siamo sforzati di eliminare dal testo ogni formalismo e discussione accademica non necessari, cercando d'altra parte di fare uso di esempi pratici per spiegare ogni nuovo concetto proposto al lettore. Il libro è quindi alla portata di qualunque lettore con una limitata conoscenza di basi di dati, sviluppo di software e tecnologie Web. Nei vari capitoli, i concetti di modellazione vengono concretamente applicati alla descrizione di siti Web reali. Allo stesso modo, gli aspetti di sviluppo sono mostrati con l'aiuto di un caso applicativo ispirato a un caso industriale reale. Nelle nostre intenzioni, questo libro dovrebbe puntare a "mostrare" i concetti proposti con l'aiuto di esempi, piuttosto che "dire" come le cose dovrebbero essere fatte.

Il libro può anche essere usato in corsi di informatica che trattano metodi di progettazione guidati dai dati, specialmente ora che le scuole di informatica e le università stanno orientando il curriculum di studio verso le tecnologie e le applicazioni Web. Come illustrato nella prossima sezione, materiali aggiuntivi e di supporto ai docenti e agli studenti sono disponibili sul sito Web del libro.

Risorse online

Al libro sono associate numerose risorse online, reperibili sul sito http://www-.webml.org, disponibile in italiano e in inglese. Sul sito sono disponibili molti materiali relativi allo sviluppo di applicazioni Web guidato da modelli di alto livello e, in particolare, a WebML. Sono presenti esempi di modellazione di ipertesti, articoli tecnici e di ricerca, materiale didattico e risorse per gli sviluppatori (per esempio, stencil WebML per Microsoft Visio, il famoso software per la rappresentazione di diagrammi). Inoltre, è presente un'ampia sezione interamente dedicata al libro. Essa contiene il codice dei programmi JSP presentati nei Capitoli 12 e 13 e un ampio numero di esercizi risolti. Dal sito è inoltre possibile contattare gli autori per ottenere ulteriori materiali didattici aggiornati o più estesi.

Il sito Web http://www.webratio.com descrive WebRatio Site Development Studio, lo strumento CASE presentato nel Capitolo 14; è possibile prelevare e provare il software grazie ad un apposito programma di valutazione e, su richiesta dei docenti che desiderano utilizzare il software nei loro corsi, è possibile ottenere delle licenze accademiche per il prodotto.

La storia di WebML

L'approccio modellistico allo sviluppo di applicazioni Web che sta alla base di questo libro è il risultato di più di cinque anni di ricerca presso il Politecnico di Milano, affiancato da un'intensa attività di sviluppo nell'industria. Il primo prototipo di ricerca di un software CASE per le applicazioni Web, chiamato AutoWeb, fu progettato tra il 1996 e il 1998 da Piero Fraternali e Paolo Paolini. Questo strumento, operativo già dal 1997, è stato utilizzato per sviluppare numerose applicazioni Web e ha consentito di dimostrare la possibilità di automatizzare la pro-

duzione di siti Web basati sui dati, specificati attraverso un linguaggio concettuale di alto livello.

WebML è nato all'interno del progetto Esprit "Web-Based Intelligent Information Infrastructures" (W3I3, 1998-2000), finanziato dall'Unione Europea, con la partecipazione di cinque partner (le italiane Politecnico di Milano e TXT e-solutions, l'olandese KPN Research, la finlandese Digia Inc. e la tedesca Otto Versand). Il progetto ha prodotto anche un prototipo di ambiente di sviluppo, chiamato ToriiSoft. Dal 1999, WebML è stato utilizzato per lo sviluppo di applicazioni Web industriali, sia all'interno di contratti di ricerca con aziende come Microsoft e Cisco Systems, sia in progetti industriali con aziende come TXT e-solutions e Acer Europe. Nel 2001, un gruppo di progettisti e sviluppatori WebML ha fondato una società con l'obiettivo di sviluppare e distribuire il prodotto WebRatio Site Development Studio, uno strumento completamente basato su WebML.

Ringraziamenti

I primi ringraziamenti e riconoscimenti vanno doverosamente a un enorme numero di sviluppatori, ricercatori e studenti che hanno contribuito alla progettazione di WebML e al conseguente sviluppo dei prodotti AutoWeb, ToriiSoft e WebRatio. Tra gli altri, vogliamo ringraziare Fabio Surini, Nicola Testa, Paolo Cucco, Roberto Acerbis, Stefano Butti, Claudio Greppi, Carlo Conserva, Fulvio Ciapessoni, Giovanni Toffetti, Marco Tagliasacchi, Andrea Rangone, Paolo Paolini, Stefano Paraboschi, Ioana Manolescu, Andrea Maurino, Marco Guida, Giorgio Tornielli, Alvise Braga Illa, Wim Timmerman, Pekka Sivonen, Stefan Liesem, Ingo Klapper, Daniel Schwabe e Graham Robson.

Un ringraziamento speciale va a Adam Bosworth, che è stato uno dei primi ad apprezzare il nostro sforzo di "cambiare il modo in cui la gente concepisce lo sviluppo di applicazioni Web". Dobbiamo a lui preziose discussioni tecniche, avvenute da entrambi i lati dell'Atlantico.

Ringraziamo Gianpiero Morbello, Massimo Manzari e Emanuele Tosetti di Acer per aver concesso il permesso di utilizzare l'applicazione Acer-Euro nelle Parti III e IV del libro.

Calorosi ringraziamenti vanno anche a tutto il personale del gruppo IKF di CISCO Systems, tra cui Mike Kirkwood, Shirley Wong, Deepa Gopinat, Seema Yazdani e Irene Sklyar. Certamente per loro è ben chiaro il concetto di applicazione Web di "grandi dimensioni"!

Siamo infine profondamente riconoscenti a Prahm Mehra e Paolo Atzeni, che ci hanno supportato con annotazioni e commenti molto precisi, rivelatisi utilissimi nella revisione del testo.