Università degli Studi di Salerno
Dipartimento di Informatica

Corso di Laurea in Informatica

TiveJS

Un'applicazione Javascript per il riconoscimento di linguaggi diagrammatici

A Javascript application for the recognition of Diagrammatic languages

Relatori
Prof. Gennaro Costagliola
Dott. Mattia De Rosa

Candidato
Dario Tecchia
Matr. 0512102581

Anno Accademico 2018/2019

Questa tesi la dedico alla mia famiglia.

Ringraziamenti

Abstract

La comunicazione visiva è in molti casi più diretta ed immediata rispetto alla comunicazione verbale: disegni, foto e mappe sono esempi di frasi visive che necessitano di un contesto per essere descritte in modo naturale.
In questa tesi presento TiveJS, un'estensione della piattaforma draw.io, che sfrutta simboli e definizioni sematiche per il riconoscimento dei linguaggi diagrammatici e la traduzione di questi in altri linguaggi. Il tool applica delle definizioni semantiche ad un diagramma e restituisce una traduzione di quest'ultimo. La traduzione avviene attraverso due fasi principali: il riconoscimento del grafo e l'applicazione delle definizioni.
Il mio lavoro di tesi si basa su strumenti precedentemente sviluppati: LoCoModeler e Tive. Precedentemente suddiviso in lato client e lato server, Tive è stato re-implementato completamente in JavaScript, prendendo il nome di TiveJS, eliminando così la necessità del server.

Indice

Ringraziamenti **2**

Abstract **3**

Elenco delle figure **6**

1 Introduzione **1**
1.1 Motivazioni 1
1.2 Lavori Correlati 2
1.2.1 draw.io e mxGraph 2
1.2.2 LoCoMoTiVE 2
1.3 Organizzazione della Tesi 5

2 Linguaggi visuali **6**
2.1 Linguaggio verbale 6
2.2 Componenti 6
2.3 Vantaggi 7

3 Local Context **8**
3.1 Sintassi 8
3.2 Semantica 8

4 TiveJS **9**
4.1 Funzionamento 9
4.2 Implementazione 9
4.2.1 Tecnologie usate 9

5 Contesti d'utilizzo **10**
5.1 Entitiy Relationship 10
5.2 Flowchart 10

5.3 Tree 10

6 Conclusioni **11**

6.1 Sviluppi futuri 11

Bibliografia **12**

Elenco delle figure

1.1 Schermata di draw.io 2
1.2 Schermata di LoCoModeler 3
1.3 Schermata di TiVE 4

Capitolo 1

Introduzione

La comunicazione fra individui avviene in svariati modi, ad esempio attraverso il linguaggio verbale ed il linguaggio visivo (o linguaggio visuale).
Un linguaggio visuale non è altro che una forma di comunicazione, detta comunicazione visuale, che fa uso di simboli grafici o immagini. Simboli grafici, immagini e mappe sono esempi di elementi utilizzati all'interno della comunicazione visiva (o comunicazione visuale) che necessitano di un contesto per essere descritte in modo naturale. Spesso quest'ultima risulta essere molto più immediata e di facile comprensione rispetto alla tradizionale comunicazione verbale coposta di lettere e parole.
In questa tesi presento **TiveJS**, un'estensione della piattaforma draw.io, che sfrutta simboli e definizioni sematiche per il riconoscimento dei linguaggi diagrammatici e la traduzione di questi in altri linguaggi.

1.1 Motivazioni

La piattaforma già esistente, LoCoMoTiVE, si basa su un meccanisco client-server.
Il client è formato dalla piattaforma draw.io, opportunamente modifica, per la creazione di sentenze visuali. Il server è stato implementato in Java utilizzando i servlet per il riconoscimento e la traduzione delle sentenze visuali. Il funzionamento è molto semplice, il client esegue una chiamata HTTP di tipo POST contenute al suo interno un grafo o un diagramma, creato attraverso l'utilizzo di simboli ad hoc, in formato XML. Una volta ricevuta la sentenza visuale, il server la interpreta applicando le definizioni per poi restituire la traduzione semantica oppure dei messaggi di errore.
Le motivazioni che ci hannno portato alla creazione di un nuovo tool sono varie: rendere l'applicazione più scalabile e più veloce limitando l'interazione con il server a semplici accessi a pagine statiche; l'aggiornamento di TiVe all'ultima versione di draw.io. Essen-

do il core di TiveJS scritto completamente in JavaScript ora si integra perfettamente con la piattaforma estesa e con la manipolazione del grafo. Le definizioni dei linguaggi ora sono definite in formato JSON rendendo ancora più alta l'interoperabilità dei sistemi.

1.2 Lavori Correlati

Il mio lavoro di tesi, essendo principalmente un porting ed una rivisitazione, si basa su strumenti precedentemente sviluppati. Gli strumenti in questione sono descritti nel dettaglio nei seguenti sottoparagrafi.

1.2.1 draw.io e mxGraph

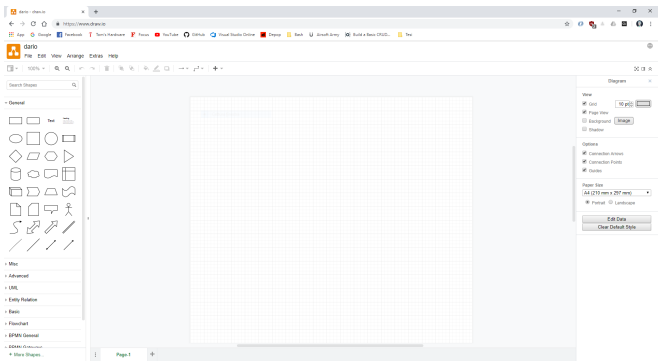

Figura 1.1 Schermata di draw.io

TiveJS, è basato su draw.io, un'applicazione web gratis che permette agli utenti di creare diagrammi e grafi direttamente dal proprio browser web, mostrato in figura 1.1. Ha un'integrazione con Google Drive e Dropbox per il salvataggio di dati che può avvenire anche con l'ausilio del localStorage del browser o attraverso il salvataggio di file sulla macchina. Draw.io è basato sulla libreria mxGraph.
Il software è stato sviluppato nel 2005 dalla JGraph Ltd.

1.2.2 LoCoMoTiVE

All'attuale stato dell'arte troviamo l'ecosistema LoCoMoTiVE, ovvero un'unione di due software, LoCoModeler e Tive. Presentato in [1] e [2], questo tool permette l'analisi semantica basata sul contesto locale, vista nel dettaglio nel capitolo 3. Nei prossimi due paragrafi andrò ad illustrare singolarmente i due componenti di cui è composto.

LoCoModeler

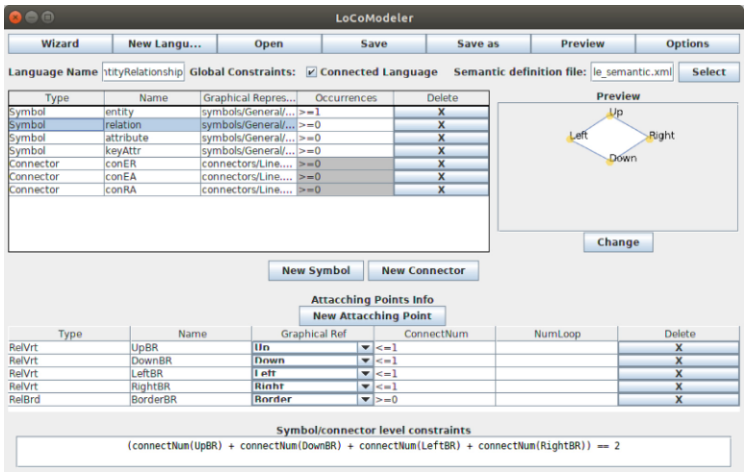

Figura 1.2 Schermata di LoCoModeler

Come descritto in [1], il modulo LoCoModeler consente ai designer la creazione e la modifica del linguaggio visivo in base al contesto locale, in maniera rapida e facile. Il suo output è la definizione in formato XML del linguaggio che verrà utilizzato durante il riconoscimento dei diagrammi. Una volta che il progettista ha completato la specifica del linguaggio, può compilarlo in un ambiente web (il modulo TiVE) per consentire agli utenti di disegnare frasi e verificarne la correttezza. Durante la definizione del linguaggio, questa funzione consente al progettista di controllare la correttezza delle specifiche.
Una schermata principale dell'interfaccia grafica del tool è mostrata nella Figura 1.2. Le sue componenti principali sono:

- Una casella di testo contenente il nome del linguaggio e una checkbox che sta ad indicare se il diagramma o grafo deve essere o non essere necessariamente connesso[1].

- Una tabella riportante le informazioni principali dei simboli e dei connettori inclusi nel linguaggio. E' possibile modificare o eliminare un elemento interagendo con la riga di questo. L'utente può aggiungere nuovi simboli o connettori usando i bottoni sottostanti la tabella.

[1]Un grafo è detto connesso se, per ogni coppia di vertici $(u, v) \in V$, esiste un cammino che collega u a v.

- Un pannello (sulla destra) mostra un'anteprima grafica del simbolo o connettore selezionato nella tabella. E' possibile cambiare la rappresentazione grafica dell'elemento utilizzando il bottone *Change*.
- Una tabella (al centro) mostra le informazioni relative al simbolo o connettore selezionato. Ogni riga mostra un punto d'attacco e i relativi vincoli. E' possibile aggiungere nuove righe utilizzando i bottoni sovrastanti la tabella.
- Un'area di testo dove è possibile specificare i vincoli per il simbolo o il connettore attraverso espressioni simili al C[2].

La definizione di un nuovo linguaggio può avvenire grazie all'ausilio di un *Wizard* diviso in tre fasi.

TiVE

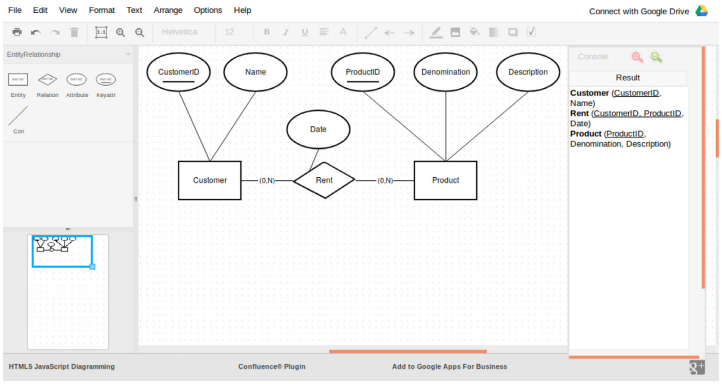

Figura 1.3 Schermata di TiVE

Una volta definito il linguaggio, i diagrammi possono essere composti utilizzando i simboli e i connettori definiti nella sua specifica. Questo può essere fatto attraverso un editor grafico TiVE, che è un'applicazione web che consente la composizione di diagrammi direttamente dal browser web.
Come mostrato nella Figura 1.3, l'applicazione è costituita da tre sezioni principali:

- Nella toolbar a sinistra troviamo la palette dei simboli e connettori utlizzabili per la creazione dei diagrammi.
- Nella zona centrale troviamo l'area di lavoro dove è possibile comporre i diagrammi trascinando gli elementi contenuti nella toolbar di sinistra.

[2]Linguaggio di programmazione.

- A destra troviamo la Console dove verrà mostrata la traduzione semantica o la lista di errori nel caso in cui si verificassero.

1.3 Organizzazione della Tesi

Nel capitolo 2 tratterò dei linguaggi visuali e dei loro componenti fondamentali. Nel capitolo 3 parlerò del Local Context e delle corrsipondenti specifiche sintattiche e semantiche. Nel capitolo 4 introdurrò il risultato del mio lavoro di tesi, TiveJS e le sue funzioni, e illustrerò i dettagli dell' implementazione e le tecnologie usate. Nel capitolo 5 illustrerò vari casi d'utilizzo da me studiati. Nel capitolo 6, presenterò le conclusioni e possibili sviluppi futuri dell'applicazione.

Capitolo 2

Linguaggi visuali

Nei precedenti capitoli ho parlato spesso di linguaggi verbali e linguaggi visuali e di quanto la comunicazione visuale può essere più efficiente della comunicazione verbale. In questo capitolo entrerò nel dettaglio senza dilungarmi andando ad illustrare quali sono le principali differenze tra un linguaggio visuale ed uno verbale, i componenti che lo compongono e in quali casi o contesti un linguaggi visivo è più efficace rispetto ad uno verbale.

2.1 Linguaggio verbale

Il linguaggio verbale è un gruppo di elementi, come suoni e parole, che messe insieme formano frasi e infine permettono la comunicazione fra individui. Da questo deriva la comunicazione verbale che è quindi costitutita dalle parole usate quando parliamo o scriviamo.

2.2 Componenti

Ogni linguaggio è formato da un proprio insieme di componenti. Un linguaggio visuale si distingue principalmente dal linguaggio verbale per i componenti di cui è formato. Il linguaggio visivo si basa su simboli grafici o immagini. Elementi che il cervello umano interpreta e trasforma in concetti, linguaggio verbale ed emozioni. Quindi se il linguaggio visuale è costitutito da testo e parole per la formazioni di frasi, il linguaggio visivo è formato da simboli e disegni per formare sentenze visive.

2.3 Vantaggi

I vantaggi possono essere molteplici, innanzitutto un linguaggio visuale può essere molto più efficace e di facile comprensione rispetto al linguaggio verbale per via della sua semplicità e naturalità. Non ha lingue o convenzioni in quanto un disegno o un'immagina non dipende da lingue o standard.

Capitolo 3

Local Context

3.1 Sintassi

3.2 Semantica

Capitolo 4

TiveJS

4.1 Funzionamento

Riconoscimento grafo

Applicazione definizioni

4.2 Implementazione

4.2.1 Tecnologie usate

Capitolo 5

Contesti d'utilizzo

5.1 Entitiy Relationship

5.2 Flowchart

5.3 Tree

Capitolo 6

Conclusioni

6.1 Sviluppi futuri

Bibliografia

[1] Vittorio Fuccella Gennaro Costagliola, Mattia De Rosa. Extending local context-based specifications of visual languages. *Journal of Visual Languages and Computing*, pages 184–195, 12 2015. doi: 10.1016/j.jvlc.2015.10.013.

[2] Vittorio Fuccella Gennaro Costagliola, Mattia De Rosa. Using the local context for the definition and implementation of visual languages. *Computer Languages, Systems and Structures*, pages 20–38, 12 2018. doi: 10.1016/j.cl.2018.04.002.