

UNIVERSITÀ DEGLI STUDI DI SALERNO

FACOLTÀ DI INFORMATICA



DIPARTIMENTO DI INFORMATICA

CORSO DI LAUREA MAGISTRALE IN INFORMATICA

DATA SCIENCE E MACHINE LEARNING

UTILIZZO DI coDIT CON SCHEMI DI GRANDI DIMENSIONI E  
STUDIO DI USABILITÀ DEGLI OPERATORI ICONICI DI coDIT

**DOCENTI**

Prof. Giuseppe Polese

Dott.ssa Loredana Caruccio

Dott. Bernardo Breve

**CANDIDATI**

Francesca Tassatone  
0522500568

Roberta Gesumaria  
0522500569

Daniele Lupo  
0522500579

---

ANNO ACCADEMICO 2018-2019

## Sommario

Introduzione .....	4
iBench .....	5
Installazione.....	5
Problemi riscontrati.....	5
Utilizzo .....	5
File di configurazione.....	5
Problemi riscontrati.....	8
CoDIT .....	11
Installazione.....	11
Problemi riscontrati durante l'installazione .....	11
Utilizzo .....	11
Struttura XMI riconosciuta da CoDIT .....	12
Problemi riscontrati durante l'utilizzo .....	14
Connessione a MySQL e creazione database .....	17
creaConnessione .....	17
creaDb .....	18
caricaDati .....	19
Utilizzo .....	20
Parser.....	21
Implementazione.....	21
Sezione iniziale .....	21
Sezione entità e attributi .....	22
Sezione relazioni .....	22
Sezione finale.....	23
Utilizzo .....	23
Riscontro sull'utilizzo di CoDIT con schemi di grandi dimensioni .....	24
Fasi e macchine utilizzate .....	24
Generazione schemi con iBench.....	24
Risultati .....	24
PC1 .....	26
PC2 .....	26
PC3 .....	26
Tempistiche di esecuzione delle fasi preliminari.....	28
PC1 .....	28
PC2 .....	28

PC3 .....	29
Studio di usabilità .....	30
Metriche utilizzate.....	30
Le fasi della procedura .....	30
Preparazione.....	30
Esecuzione .....	33
Analisi dei risultati .....	34
Conclusioni .....	39
Indice delle tabelle .....	41

**NB.** Gli artefatti prodotti durante il progetto sono disponibili al seguente [link](#).

## Introduzione

Il progetto che sarà svolto riguarderà lo studio di usabilità sull'espressività e la complessità percepita degli operatori iconici di CoDIT utilizzando scenari che coinvolgono schemi di grosse dimensioni. Per generare tali schemi ci è stato indicato di utilizzare il tool iBench.

Vediamo nel dettaglio il lavoro da svolgere.

Utilizzo di CoDIT, un tool per l'integrazione di schemi. Gli schemi su cui lavora sono schemi concettuali sui quali è possibile applicare operatori iconici per l'integrazione. I formati richiesti da CoDIT sono XMI o SVG.

Per lo studio richiesto, avremo però bisogno di schemi più grandi di quelli che si hanno a disposizione, per ottenere ciò verrà utilizzato iBench.

iBench genera schemi logici in formato XML basandosi su un file di configurazione.

Le prime differenze da notare sono le tipologie di file e di schemi differenti che si ottengono da un tool e che vengono elaborati dall'altro. Inoltre, i file XMI riconosciuti da CoDIT hanno una struttura non generica, quindi sarà necessario estrarre dallo schema logico le informazioni utili per ottenere lo schema concettuale, ottenendo, quindi, una struttura adeguata comprensibile a CoDIT.

Successivamente, verrà effettuato uno studio di usabilità sull'espressività e la complessità percepita degli operatori iconici di CoDIT, sia dai membri del gruppo che da altri utenti, sempre con competenze informatiche.

## iBench

iBench è un generatore di metadati per la creazione di mappature, schemi e vincoli di schemi arbitrariamente grandi e complessi. Può essere utilizzato con un generatore di dati per generare efficacemente scenari realistici di integrazione dei dati con diversi gradi di dimensione e complessità. Inoltre, permette di creare benchmark per diverse attività di integrazione dei dati (virtuali), integrazione dei dati, scambio di dati, evoluzione degli schemi, mappatura degli operatori come composizione e inversione e corrispondenza degli schemi.

Modificando il file di configurazione predefinito, l'utente specifica il tipo di metadati, i dati da generare e le proprietà dello scenario generato. iBench produce un file XML che memorizza i metadati generati in base alla configurazione dell'utente. Se richiesto, genera anche i dati di istanza per gli schemi generati.

### Installazione

iBench è scritto in Java, per cui c'è bisogno di utilizzare *ant* per il building e *ivy* per scaricare le dipendenze.

La prima operazione svolta è stata quella di scaricare e installare *ant*.

Eseguendo il comando *ant* nella cartella principale, è stato scaricato *ivy* ed utilizzato automaticamente per scaricare il jar delle dipendenze. Successivamente, è stata generata una cartella build in cui sono presenti vari script di shell (.sh per Linux e Mac OS e .bat per Windows).

Il tutto è riportato in modo semplice nel file README messo a disposizione.

### Problemi riscontrati

Nell'utilizzo del file .bat per l'esecuzione di iBench è stata riscontrata la mancanza, nel codice, dell'indicazione dell'acquisizione dell'argomento dato da linea di comando. Successivamente all'aggiunta di tale sezione di codice, l'esecuzione è terminata senza alcun problema.

Codice iBench.bat non funzionante:

```
java -Xmx4096m -jar ibench-fat.jar
```

Codice iBench.bat funzionante:

```
java.exe -jar ibench-fat.jar -c %2
```

Inoltre, il parametro *-Xmx4096m* limitava l'utilizzo della RAM a 4GB.

### Utilizzo

iBench permette di generare source schema e target schema.

Per poter generare questi schemi viene messo a disposizione un file di configurazione. Attraverso quest'ultimo è possibile settare le dimensioni e la struttura che si vuole ottenere per gli schemi risultanti.

I nomi dell'intera struttura e i dati vengono generati in modo casuale.

### File di configurazione

Vediamo nel dettaglio le varie sezioni del file di configurazione che abbiamo utilizzato per la generazione degli schemi.

La prima parte è relativa alla definizione della *path* in cui devono essere salvati gli schemi e i dati generati, poi viene indicato il nome del file che conterrà i metadati relativi alla struttura dello schema generato.

```
# Output Path Prefixes
SchemaPathPrefix=person
InstancePathPrefix=person

# Schema file Name
FileNames.Schemas=metadataPerson.xml
```

Successivamente troviamo la parte per la configurazione sul numero di istanze, ossia entità per ogni schema, il numero di attributi per ogni entità ed altre configurazioni relative ai dati generati.

```
# Number of Instances for each Basic Scenario Type
Scenarios.COPY = 4

# Number of attributes
ConfigOptions.NumOfSubElements = 5

# Maximum length of strings created by the data generator
MaxStringLength = 100

# Maximum numerical value created by the data generator
MaxNumValue = 1000

# Type of data generator (currently only TrampCSV supported)
DataGenerator = TrampCSV

# Type of query generator to use (Postgres, Perm)
QueryGenerator = Postgres

# Mapping language used (currently FO tgds or SO tgds are supported)
MappingLanguage = FOTgds

# Number of tuples per relation, if data is generated
RepElementCount = 100

# Number of independent tuples (not created by data exchange) generated for each target relation
#TargetTableNumRows = 50
```

Poiché i dati sono generati in modo casuale è possibile indicare il seme per il generatore casuale. Inoltre, per introdurre variabilità nella costruzione delle relazioni, è possibile indicare la deviazione relativa al numero di attributi per relazione.

```
# Seed for the random number generator, used for repeatability
RandomSeed = 3

# Deviation of attributes for each relation
ConfigOptionsDeviation.NumOfSubElements = 3
```

La definizione delle dipendenze viene stabilita tramite percentuali. La percentuale delle dipendenze viene applicata al totale degli schemi generati, considerando sia source che target; quella delle chiavi esterne viene applicata sul numero di dipendenze generate.

Per esempio, utilizzando una configurazione con i seguenti parametri: 100 entità, 50% di dipendenze e 50% di chiavi esterne avremo 50 dipendenze con 25 chiavi esterne.

```
#Generate random inclusion dependencies
ConfigOptions.SourceInclusionDependencyPerc = 50
ConfigOptions.SourceInclusionDependencyFKPerc = 50

ConfigOptions.TargetInclusionDependencyPerc = 50
ConfigOptions.TargetInclusionDependencyFKPerc = 50
```

Per la generazione dei dati viene messa a disposizione una sezione in cui è possibile definire tipologie specifiche (dati UDT - User Definition Data Type), come e-mail, numero di telefono ed altri che possono essere scelti tra le classi che si trovano al seguente [link](#). In questa sezione bisogna indicare il numero di UDT che saranno definiti, poi per ognuno di esso bisognerà specificare un nome univoco, la classPath relativa alla classe della tipologia del dato che si vuole generare, la percentuale di generazione ed il tipo in cui verrà codificato il campo in un database SQL.

```
#####  
# UDTs  
#####  
  
DataType.NumDataType = 5  
DataType.0.Name = Emails  
DataType.0.ClassPath = toxgene.util.cdata.xmark.Emails  
DataType.0.Percentage = 20.0  
DataType.0.DBType = TEXT  
  
DataType.1.Name = Phones  
DataType.1.ClassPath = toxgene.util.cdata.xmark.PhoneNumbers  
DataType.1.Percentage = 20.0  
DataType.1.DBType = TEXT  
  
DataType.2.Name = FirstNames  
DataType.2.ClassPath = toxgene.util.cdata.xmark.FirstNames  
DataType.2.Percentage = 20.0  
DataType.2.DBType = TEXT  
  
DataType.3.Name = LastNames  
DataType.3.ClassPath = toxgene.util.cdata.xmark.LastNames  
DataType.3.Percentage = 20.0  
DataType.3.DBType = TEXT  
  
DataType.4.Name = Cities  
DataType.4.ClassPath = toxgene.util.cdata.xmark.Cities  
DataType.4.Percentage = 20.0  
DataType.4.DBType = TEXT
```

Inoltre, è possibile scegliere i dati, sempre in modo casuale, ma da un set fornito tramite file csv. Per configurare questa parte deve essere indicato il file csv, il nome della colonna all'interno del file, la percentuale di generazione ed infine il tipo in cui verrà convertito il campo in un database SQL.

```
# CSV file this UDT is coming from  
CSVDataType.0.File = zip_codes_states.csv  
  
# Name of the attribute from which the values are read  
CSVDataType.0.AttrName = state  
  
# the percentage of attributes that should use this DT (this is interpreted as a probability)  
CSVDataType.0.Percentage = 20.0  
  
# when loading generated data into a database, use this SQL datatype for the column of this UDT  
CSVDataType.0.DBType = TEXT
```

Tramite i seguenti parametri è possibile gestire alcune differenze tra lo schema source e target. Utilizzandoli verrà effettuato il partizionamento (orizzontale, verticale), verranno aggiunti o rimossi alcuni attributi oppure verrà effettuato il merge di alcuni attributi.

```

# Horizontally partition a source relation into multiple target relations
Scenarios.HORIZPARTITION = 0

# Inverse of vertical partitioning, join multiple vertical fragments from the source to create a target relation
Scenarios.MERGING = 0

# Vertically partition a source relation
Scenarios.VERTPARTITION = 0

# Copies a source relation to the target and adds one or more new attributes without correspondence to the source
Scenarios.ADDATTRIBUTE = 0

# Copies a source relation to the target removing one or more new attributes
Scenarios.DELATTRIBUTE = 0

# Combination of ADDATTRIBUTE and DELATTRIBUTE
Scenarios.ADDDELATTRIBUTE = 0

```

Infine, sono presenti due sezioni relative ad opzioni sui dati di output che verranno generati.

```

#####
# Optional activation/deactivation of output options
#####

# Generate HTML schema
OutputOption.HTMLSchemas = false

# Generate source data
OutputOption.Data = true

# Generate target data that is independent of the source data
OutputOption.EnableTargetData = true

# Generate XMLSchema schemas for the source and target schemas
OutputOption.XMLSchemas = true

# Generate an HTML description of the source to target mappings
OutputOption.HTMLMapping = false

# Generate TrampXML file, an XML based metadata format storing the generated schemas, mappings, constraints, etc.
OutputOption.TrampXML = true

# Generate a Clio conformant mapping file
OutputOption.Clio = true

#####
# Optional activation/deactivation of parts of the generated Tramp XML document
#####

# Generate correspondences aka schema matches
TrampXMLOutput.Correspondences = true

# Generate transformations implementing the mappings (currently only SQL)
TrampXMLOutput.Transformations = true

# Generate data
TrampXMLOutput.Data = true

# Generate a connection info (allows Tramp tools to connect to a database, e.g., to load a schema)
TrampXMLOutput.ConnectionInfo = false

# Generate functional dependencies
TrampXMLOutput.FDs = false

```

## Problemi riscontrati

I problemi riscontrati si possono dividere in base all'obiettivo della configurazione, ovvero:

- Generazione dati per lo schema target;
- Utilizzo di csv per la generazione dei dati.



### Generazione dati per lo schema target

Avendo bisogno dei dati relativi agli schemi generati, abbiamo configurato iBench in modo da abilitare la creazione dei file csv relativi alla generazione di tali dati. Per la creazione dei dati relativi allo schema target abbiamo riscontrato dei problemi in caso di particolari configurazioni.

```
# Generate target data that is independent of the source data
OutputOption.EnableTargetData = true

# Number of independent tuples (not created by data exchange) generated for each target relation
TargetTableNumRows = 50
```

Figura 1: proprietà per l'abilitazione della generazione dei csv del target schema e il numero di tuple da creare.

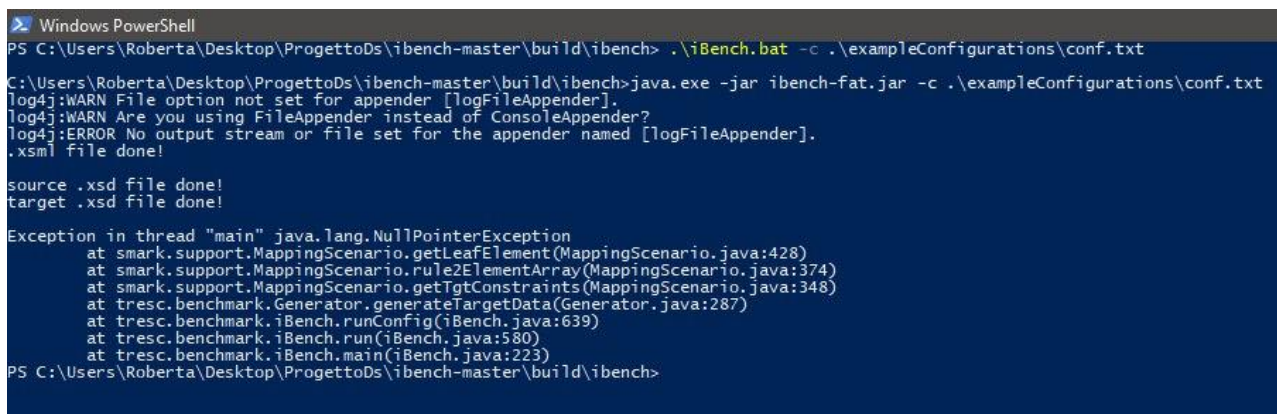
Le configurazioni che non sono risultate adatte alla generazione dei dati sono state:

- Abilitazione delle proprietà relative alle percentuali sulle dipendenze e chiavi esterne;
- Partizionamento orizzontale e verticale.

Vediamo nel dettaglio gli errori per entrambi i casi.

### Abilitazione delle proprietà relative alle percentuali sulle dipendenze e chiavi esterne

La generazione dei file csv è stata possibile solo se le proprietà sulle percentuali relative alle dipendenze e chiavi esterne non erano state abilitate. Infatti, nel caso di abilitazione di tali proprietà, non venivano generati i dati relativi allo schema target, restituendo il seguente errore:



```
Windows PowerShell
PS C:\Users\Roberta\Desktop\ProgettoDs\ibench-master\build\ibench> .\iBench.bat -c .\exampleConfigurations\conf.txt
C:\Users\Roberta\Desktop\ProgettoDs\ibench-master\build\ibench> java.exe -jar ibench-fat.jar -c .\exampleConfigurations\conf.txt
log4j:WARN File option not set for appender [logFileAppender].
log4j:WARN Are you using FileAppender instead of ConsoleAppender?
log4j:ERROR No output stream or file set for the appender named [logFileAppender].
.xml file done!
source .xsd file done!
target .xsd file done!
Exception in thread "main" java.lang.NullPointerException
    at smark.support.MappingScenario.getLeafElement(MappingScenario.java:428)
    at smark.support.MappingScenario.rule2ElementArray(MappingScenario.java:374)
    at smark.support.MappingScenario.getTgtConstraints(MappingScenario.java:348)
    at tresc.benchmark.Generator.generateTargetData(Generator.java:287)
    at tresc.benchmark.iBench.runConfig(iBench.java:639)
    at tresc.benchmark.iBench.run(iBench.java:580)
    at tresc.benchmark.iBench.main(iBench.java:223)
PS C:\Users\Roberta\Desktop\ProgettoDs\ibench-master\build\ibench>
```

Seguendo lo stack di eccezioni giungiamo al seguente snippet di codice:

```
public Vector<SMarkElement[][]> getTgtConstraints()
{
    Vector<SMarkElement[][]> tgtConstraints=new Vector<SMarkElement[][]>();
    for(int i=0;i<_target.getConstrSize();i++)
    {
        Rule thisConstraint=_target.getConstraint(i);
        if(thisConstraint instanceof ForeignKey)
        {
            //maybe this is the problem?? \//
            SMarkElement[][] thisElementArray=rule2ElementArray(thisConstraint);
            tgtConstraints.add(thisElementArray);
        }
    }
    return tgtConstraints;
}
```

Figura 2: Snippet di codice presente nella classe MappingScenario.

Sì, probabilmente è questo il problema!

## Partizionamento orizzontale e verticale

Nel generare schemi con l'abilitazione delle proprietà per il partizionamento orizzontale o verticale, i nomi delle entità che vengono generati nel target sono creati con una sezione in maiuscolo, ad esempio "HPOFR0". Con l'abilitazione delle proprietà per la generazione dei dati per il target, l'esecuzione restituisce il seguente errore:

```
Exception in thread "main" java.lang.Exception: Did not find target relation with name <connection_hp_0_n10_ce0_hp0fr0_from_0_to_4999>
    at org.vagabond.benchmark.model.TrampXMLModel.getRelForName(TrampXMLModel.java:268)
    at tresc.benchmark.dataGen.ToXScriptOnlyDataGenerator.generateUnique(ToXScriptOnlyDataGenerator.java:372)
    at tresc.benchmark.dataGen.ToXScriptOnlyDataGenerator.generateToxList(ToXScriptOnlyDataGenerator.java:273)
    at tresc.benchmark.dataGen.ToXScriptOnlyDataGenerator.generateToxTemplate(ToXScriptOnlyDataGenerator.java:139)
    at tresc.benchmark.dataGen.ToXScriptOnlyDataGenerator.generateData(ToXScriptOnlyDataGenerator.java:120)
    at tresc.benchmark.dataGen.ToDataGenerator.generateData(ToDataGenerator.java:73)
    at tresc.benchmark.dataGen.TrampCSVGen.generateData(TrampCSVGen.java:110)
    at tresc.benchmark.Generator.generateTargetData(Generator.java:289)
    at tresc.benchmark.iBench.runConfig(iBench.java:639)
    at tresc.benchmark.iBench.run(iBench.java:580)
    at tresc.benchmark.iBench.main(iBench.java:223)
```

Analizzando il codice, presumiamo che l'errore sia dovuto ad un'acquisizione con una forzatura al minuscolo del nome delle entità relative agli schemi.

```
// generate primary key constraints
private String generateUnique(SMarkElement schemaElement, String schemaName) throws Exception {
    boolean isTarget = schemaName.equalsIgnoreCase("Source") ? false : true;

    RelationType rel = scen.getDoc().getRelForName(schemaElement.getLabel().toLowerCase(), isTarget);

    if (!rel.isSetPrimaryKey())
        return "";
}
```

## Utilizzo di csv per la generazione dei dati

Nella sezione per UDT si possono specificare i file da cui prendere i valori per generare i dati. Provando con un solo file csv venivano generate righe contenenti più tuple. Inoltre, le percentuali delle chiavi esterne e delle dipendenze non erano coerenti alla configurazione specificata.

## CoDIT

CoDIT (Conceptual Data Integration Tool), supporta il processo di integrazione dei dati a livello concettuale, permettendo la costruzione di uno schema globale correlando sottoschemi delle sorgenti e, in questo modo, il DBA potrà specificare in maniera visuale come combinare le sorgenti di dati. Lo schema globale potrà essere specificato allineando i sottoschemi delle sorgenti oppure specificando nuovi costrutti e mappando i sottoschemi delle sorgenti ad essi.

A livello metodologico CoDIT si basa su un linguaggio visuale, CoDIL (Conceptual Data Integration Language).

CoDIL permette di gestire l'eterogeneità degli schemi, per far fronte alla necessità di risolvere i conflitti dovuti all'utilizzo di diversi costrutti e/o nomi per modellare la stessa realtà.

Attraverso CoDIL, il DBA potrà selezionare costrutti degli schemi sorgenti (uno per ogni schema) in relazione ed associare un operatore iconico per specificare come risolvere il conflitto.

Il linguaggio CoDIL fornisce un insieme di operatori iconici che possono essere applicati a coppie di sottoschemi concettuali. Attraverso l'utilizzo di tali operatori è possibile specificare in che modo i sottoschemi selezionati sono correlati e scegliere come questi devono essere mappati nello schema riconciliato.

### Installazione

Per l'installazione del tool è stato utilizzato il file README messo a disposizione. Il file è molto dettagliato sia per l'installazione su Mac OS che per Windows. Inoltre, la guida per il sistema Mac OS può essere eseguita anche per Windows.

### Problemi riscontrati durante l'installazione

Nel file README viene menzionata una guida per modificare il file *web.xml* della configurazione di Catalina. Purtroppo, all'interno della guida sono presenti alcuni errori, come ad esempio dei tag di apertura o chiusura mancanti. Una volta risolti i vari errori, si ottiene il file *web.xml*, disponibile al seguente [link](#).

### Utilizzo

Le funzionalità offerte da CoDIT sono:

1. Disegna schema;
2. Caricamento dinamico;
3. Create target scheme.

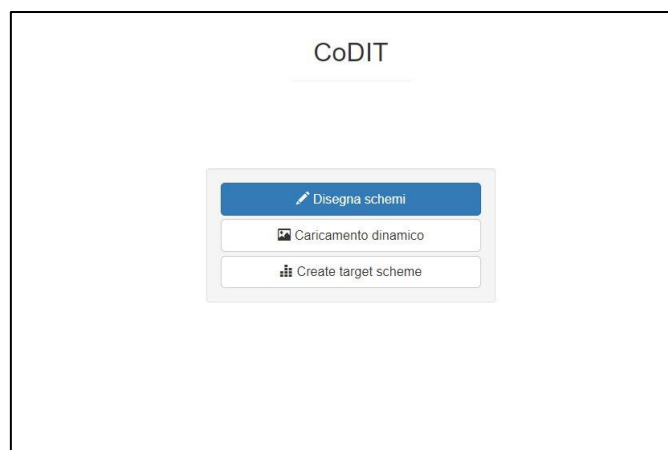


Figura 3: Home tool CoDIT.

L'operazione sulla quale ci siamo focalizzati è il caricamento dinamico. I formati riconosciuti da CoDIT sono XMI e SVG. Il file XMI, però, ha una struttura specifica che riporteremo di seguito.

### Struttura XMI riconosciuta da CoDIT

La struttura del file XMI può essere suddivisa in quattro parti:

- Una sezione iniziale;
- Una sezione in cui vengono indicate tutte le entità con i relativi attributi;
- Una sezione in cui vengono indicate tutte le relazioni;
- Una sezione finale.

#### Sezione iniziale

La seguente sezione risulta essere identica in tutti i file forniti come esempi per un primo utilizzo di CoDIT.

```
<uml:Model xmi:version="2.1" xmlns:xmi="http://schema.omg.org/spec/XMI/2.1" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns:ecore="http://www.eclipse.org/emf/2002/Ecore"
xmlns:uml="http://schema.omg.org/spec/UML/2.1.1" xsi:schemaLocation="http://schema.omg.org/spec/UML/2.1.1 http://www.eclipse.org/uml2/2.0.0/UML" xmi:id="_XrSbEdk5EdyEZoPpUv3LUw" name="Blank Model">
<packageImport xmi:type="uml:PackageImport" xmi:id="_XrSbEdk5EdyEZoPpUv3LUw">
  <importedPackage xmi:type="uml:Model" href="http://schema.omg.org/spec/UML/2.1.1/uml.xml#_0"/>
</packageImport>
```

Figura 4: Struttura XML CoDIT, sezione iniziale.

#### Sezione entità e attributi

Per indicare l'entità viene utilizzato il tag "packagedElement", con i seguenti attributi:

- xmi:type = uml:Class, per indicare che si tratta di un'entità;
- xmi:id, per indicare l'id;
- name, per indicare il nome dell'entità;
- type, per indicare il tipo di entità. Non sempre viene specificato, di solito è usato per indicare le entità deboli (WeakEntity).

```
<packagedElement xmi:type="uml:Class" xmi:id="room" name="room">
```

Figura 5: Struttura XML CoDIT, entità.

Per ogni entità, come figlio dell'elemento "packagedElement", viene indicato l'attributo "xmi:Extension" per specificare che si tratta di un'entità.

```
<xmi:Extension extender="http://www.eclipse.org/emf/2002/Ecore">
  <eAnnotations xmi:type="ecore:EAnnotation" xmi:id="activityEAnnotation" source="http://www.eclipse.org/uml2/2.0.0/UML">
    <details xmi:type="ecore:EStringToStringMapEntry" xmi:id="phpbb_f_Entity" key="Entity"/>
  </eAnnotations>
</xmi:Extension>
```

Figura 6: Struttura XML CoDIT, estensione identificativa dell'entità.

Per indicare gli attributi relativi all'entità viene utilizzato il tag "ownedAttribute" come figlio dell'elemento "packagedElement".

Per ogni attributo vengono definiti:

- xmi:type="uml:Property, per indicare che è un attributo;
- xmi:id, per indicare l'id;
- name, per indicare il nome dell'attributo;
- visibility, per indicare la visibilità dell'attributo.

```
<ownedAttribute xmi:type="uml:Property" xmi:id="roomNum" name="roomNum" visibility="private">
</ownedAttribute>
```

Figura 7: Struttura XML CoDIT, attributo.

Nel caso in cui l'attributo sia chiave primaria, viene utilizzato il tag "xmi:Extension", come figlio di "ownedAttribute", per indicare che si tratta di una chiave primaria.

```
<ownedAttribute xmi:type="uml:Property" xmi:id="id" name="id" visibility="private">
  <xmi:Extension extender="http://www.eclipse.org/emf/2002/Ecore">
    <eAnnotations xmi:type="ecore:EAnnotation" xmi:id="_Ovi-VuPdEdy8F_b8rGg0Zw" source="http://www.eclipse.org/uml2/2.0.0/UML">
      <details xmi:type="ecore:EStringToStringMapEntry" xmi:id="_Ovi-V-PdEdy8F_b8rGg0Zw" key="PK"/>
    </eAnnotations>
  </xmi:Extension>
</ownedAttribute>
```

Figura 8: Struttura XML CoDIT, attributo chiave primaria.

### Sezione relazioni

In questa sezione vengono definite le relazioni tra le entità sopra definite.

Viene usato sempre il tag "packagedElement", questa volta però è utilizzato con lo scopo di definire le associazioni, e viene specificato nell'attributo "xmi: type= uml:Association".

La relazione tra le entità è indicata attraverso l'attributo "xmi:id" in cui è specificato il nome delle due entità relazionate, usando il seguente formato E1\_\_E2\_\_id.

Inoltre, se si deve definire una specializzazione, oltre ad indicare un id in cui ci sono i nomi delle entità, viene inserito anche l'attributo "type=ISA".

```
<packagedElement xmi:type="uml:Association" xmi:id="attribute__numBeds__id" />
<packagedElement xmi:type="uml:Association" xmi:id="attribute__smokingPreference__id" />
<packagedElement xmi:type="uml:Association" xmi:id="attribute__onFloor__id" />
<packagedElement xmi:type="uml:Association" xmi:id="room__attribute__id" />
<packagedElement xmi:type="uml:Association" xmi:id="room__townHouse__id" type="ISA" />
<packagedElement xmi:type="uml:Association" xmi:id="room__suite__id" type="ISA"/>
<packagedElement xmi:type="uml:Association" xmi:id="room__oneRoom__id" type="ISA" />
```

Figura 9: Struttura XML CoDIT, sezione relazioni tra le entità.

### Sezione finale

La seguente sezione risulta essere identica in tutti i file forniti come esempi per un primo utilizzo di CoDIT.

```
<profileApplication xmi:type="uml:ProfileApplication" xmi:id="_XrSbHdk5EdyEz0PpUv3LUw">
  <xmi:Extension extender="http://www.eclipse.org/emf/2002/Ecore">
    <eAnnotations xmi:type="ecore:EAnnotation" xmi:id="_XrSbHtk5EdyEz0PpUv3LUw" source="http://www.eclipse.org/uml2/2.0.0/UML">
      <references xmi:type="ecore:EPackage" href="http://schema.omg.org/spec/UML/2.1.1/StandardProfileL2.xmi#_vzU58YinEdgtvbnfB2L_5w"/>
    </eAnnotations>
  </xmi:Extension>
  <appliedProfile xmi:type="uml:Profile" href="http://schema.omg.org/spec/UML/2.1.1/StandardProfileL2.xmi#_0"/>
</profileApplication>
```

Figura 10: Struttura XML CoDIT, sezione finale.

### Coordinate elementi

Un'ulteriore funzionalità offerta da CoDIT è quella di salvare la posizione degli elementi, memorizzando le relative coordinate all'interno del file XML. Quindi, per ogni elemento identificativo di una entità, attributo o relazione, sono presenti gli attributi x e y.

```

<packagedElement xmi:type="uml:Class" xmi:id="activity" name="activity" x="1682" y="221">
  <xmi:Extension extender="http://www.eclipse.org/emf/2002/Ecore">
    <eAnnotations xmi:type="ecore:EAnnotation" xmi:id="activityEAnnotation" source="http://www.eclipse.org/uml2/2.0.0/uml">
      <details xmi:type="ecore:EStringToStringMapEntry" xmi:id="activity_Entity" key="Entity"/>
    </eAnnotations>
  </xmi:Extension>

  <ownedAttribute xmi:type="uml:Property" xmi:id="ActivityRecordID" name="ActivityRecordID" visibility="private" x="1572" y="97">
  </ownedAttribute>

```

Figura 11: Struttura XML CoDIT, entità e attributi con coordinate.

## Problemi riscontrati durante l'utilizzo

Durante l'utilizzo preliminare di CoDIT, ancora non incentrato sullo studio di usabilità richiesto, sono stati riscontrati vari problemi, alcuni dei quali sono stati compresi e risolti per poter utilizzare schemi di grandi dimensioni.

### Limitazione della visualizzazione degli schemi

Il primo problema riscontrato era legato alla visualizzazione degli schemi dalle dimensioni sotto indicate, generati con il tool iBench e successivamente convertiti nel formato xmi leggibile dal tool CoDIT.

Gli schemi utilizzati erano di tali dimensioni:

ID schema	#Entità	#Attributi per entità <sup>1</sup>	#Tuple	Percentuale dipendenze	Percentuale chiavi esterne
Schema1	50	10	100	50	50
Schema2	80	10	100	50	50

Tabella 1: Tabella relativa agli schemi nella fase iniziale dell'utilizzo di CoDIT.

Nel caricamento di questi schemi il risultato ottenuto era di due tipi:

1. per lo Schema1 la sezione atta alla visualizzazione dello schema appariva bianca. Gli errori che venivano riscontrati sulla console del browser (Chrome) e su Eclipse erano i seguenti:

```

✖ Failed to load resource: the server responded with a status of 404 ()
{"schema1": "", "schema2": "schema1"}
⚠ [Deprecation] Synchronous XMLHttpRequest on the main thread is deprecated because of its detrimental effects to the end user's experience. For more help, check https://xhr.spec.whatwg.org/.
LOADING SCHEMAS: Si è verificato un errore!
INIZIO PARSER
✖ Uncaught TypeError: Cannot read property 'x' of undefined
    at Element.<anonymous> (parser1.js:1511)
    at Function.each (jquery.min.js:2)
    at w.fn.init.each (jquery.min.js:2)
    at Object.parseIt [as success] (parser1.js:1472)
    at u (jquery.min.js:2)
    at Object.fireWith [as resolveWith] (jquery.min.js:2)
    at k (jquery.min.js:2)
    at XMLHttpRequest.<anonymous> (jquery.min.js:2)

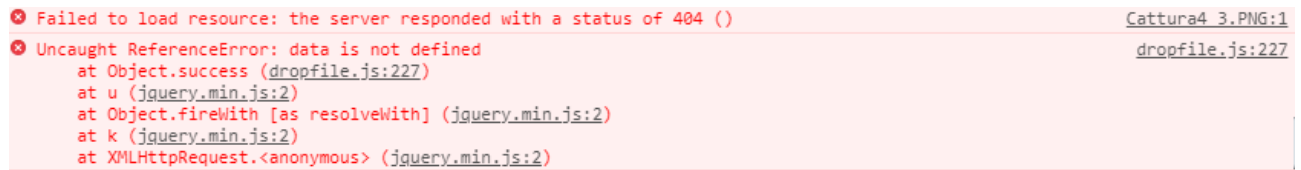
```

Figura 12: Console browser





2. per lo Schema2, la schermata atta alla visualizzazione degli schemi restava bloccata sul caricamento indicando la dimensione del file che si voleva caricare.  
L'errore veniva riportato solo nelle console del browser:

A screenshot of a browser's developer console showing two error messages. The first error is 'Failed to load resource: the server responded with a status of 404 ()' with a file icon and a link to 'Cattura4 3.PNG:1'. The second error is 'Uncaught ReferenceError: data is not defined' with a file icon and a link to 'dropfile.js:227'. The stack trace for the second error includes: 'at Object.success (dropfile.js:227)', 'at u (jquery.min.js:2)', 'at Object.fireWith [as resolveWith] (jquery.min.js:2)', 'at k (jquery.min.js:2)', and 'at XMLHttpRequest.<anonymous> (jquery.min.js:2)'.

```
✖ Failed to load resource: the server responded with a status of 404 () Cattura4 3.PNG:1
✖ Uncaught ReferenceError: data is not defined dropfile.js:227
    at Object.success (dropfile.js:227)
    at u (jquery.min.js:2)
    at Object.fireWith [as resolveWith] (jquery.min.js:2)
    at k (jquery.min.js:2)
    at XMLHttpRequest.<anonymous> (jquery.min.js:2)
```

Discutendone con il dottorando Bernardo Breve, siamo riusciti a risolvere il problema relativo allo Schema1, modificando parte dell'algoritmo usato per visualizzare gli elementi dello schema (parser1.js). Per quanto riguarda il problema relativo allo Schema2, invece, non è stato ancora risolto poiché viene ancora riscontrato nella sezione di destra ogni qualvolta dovrebbe essere visualizzato un modal di errore. Quindi, supponiamo sia legato ad un errore relativo alla creazione del modal.

#### *Limitazione sulla dimensione massima dei file caricati*

Dopo aver risolto il problema precedente legato alla mancata visualizzazione di alcuni schemi, abbiamo provato a visualizzare schemi di grosse dimensioni. Non è stato possibile a causa della dimensione del file che eccedeva quella di caricamento massima.

Anche per questo errore, trovato il punto in cui veniva definita la dimensione massima (upload\_file.php, riga 9) è stato risolto il problema portando la dimensione ad un massimo di 1Gb.

#### *Limitazione della sezione di visualizzazione degli schemi*

Riuscendo a caricare e visualizzare schemi di grosse dimensioni ci siamo resi conto che la sezione atta alla visualizzazione è limitata dato che, pur permettendo lo scorrimento, essa non risulta infinita. In conclusione, non è possibile visualizzare nel complesso schemi eccessivamente grandi.



## Connessione a MySQL e creazione database

Per alcune funzionalità messe a disposizione da CoDIT è necessario che, oltre ad essere in possesso di uno schema concettuale, ci sia anche un database popolato. Abbiamo, quindi, creato uno script in Python, che permettesse di connettersi a MySQL e ci permettesse di creare e popolare nuovi database.

```
import xml.etree.ElementTree as ET
import mysql.connector
import sys
import csv

def creaConnessione(nomeDB):...

def creaDb(tipoSchema):...

def caricaDati(cursor, path, nomeEntita, numeroAttributi):...

if((sys.argv. len ()) != 3):
    print("Uso: nomeScript nomeFile/pathFile nomeDb")
else:
    creaDb("SourceSchema")
    creaDb("TargetSchema")
```

Di seguito verranno illustrati nel dettaglio i vari metodi implementati.

### creaConnessione

```
def creaConnessione(nomeDB):
    return mysql.connector.connect(
        host="localhost",
        user="root",
        passwd="",
        database= nomeDB
    )
```

La libreria mysql.connector mette a disposizione un oggetto dove vengono salvate tutte le informazioni necessarie per la connessione. Dato che serve una connessione per ogni database, è stato implementato un metodo che si occupa di configurare la connessione al database, specificato come parametro.

## creaDb

```
def creaDb(tipoSchema):
    # Connessione a mysql e creazione db
    mydb = creaConnessione("")
    mycursor = mydb.cursor()
    nomeDb = tipoSchema + "_" + sys.argv[2]
    # Elimino il db se già esiste
    mycursor.execute("DROP DATABASE IF EXISTS '" + nomeDb + "';")
    mydb.commit()
    mycursor.execute("CREATE DATABASE " + nomeDb)
    mycursor.execute("SET GLOBAL FOREIGN_KEY_CHECKS=0;")
    mydb.commit()
    mydb.close()

    tree = ET.parse(sys.argv[1])
    root = tree.getroot()
    table = ""
    # Recupero la parte iniziale
    path = os.path.normpath(sys.argv[1])
    nomeFile = path.split(os.sep)[-1]
    inizioPath = sys.argv[1]
    inizioPath = inizioPath[: (sys.argv[1].__len__() - nomeFile.__len__())]

    # Connessione al db appena creato e generazione delle tabelle con attributi
    mydb = creaConnessione(nomeDb)
    mycursor = mydb.cursor()
    for c in root.findall("./Schemas/" + tipoSchema + "/Relation"):
        # Per ogni relazione, conto quanti attributi ci sono
        nAttr = 0
        nameEn = c.get('name')
        table = 'CREATE TABLE ' + nameEn + ' ('

        attributi = root.findall("./Schemas/" + tipoSchema + "/*[@name='" + nameEn + "']/Attr")
        for a in attributi:
            table += a.find('Name').text
            if (a.find('DataType').text == 'TEXT'):
                table += ' varchar(100), '
            elif (a.find('DataType').text == 'INT8'):
                table += ' INT(100), '
            nAttr = nAttr + 1

        table = table[:-1]

        # Aggiungo la PK
        pk = root.findall("./Schemas/" + tipoSchema + "/*[@name='" + nameEn + "']/PrimaryKey")
        if (pk is not None):
            table += ", PRIMARY KEY (" + pk[0].find("Attr").text + ")"
        table += ');'
        mycursor.execute(table)
        if (tipoSchema == "SourceSchema"):
            caricaDati(mycursor, inizioPath, nameEn, nAttr)
    mydb.commit()
```

```

# Aggiungo le chiavi esterne alle tabelle
mycursor = mydb.cursor()
query = ""
fks = root.findall("./Schemas/" + tipoSchema + "/ForeignKey")
for f in fks:
    # Tabella from
    t1 = f.find("From").get("tableref")
    # Campo from
    e1 = f.find("From/Attr").text
    # Tabella to
    t2 = f.find("To").get("tableref")
    # Campo to
    e2 = f.find("To/Attr").text
    query = "ALTER TABLE " + t1 + " ADD FOREIGN KEY (" + e1 + ") REFERENCES " + t2 + "(" + e2 + ");"
    mycursor.execute(query)
mydb.commit()
# Setto di nuovo il check delle chiavi esterne
mycursor.execute("SET GLOBAL FOREIGN_KEY_CHECKS=1;")
mydb.commit()
mydb.close()

```

Questo metodo si occupa di leggere le informazioni dal file *xml* contenente la struttura degli schemi. Una volta eliminati i database, se già esistenti, crea i nuovi database che conterranno le relazioni degli schemi *Source* e *Target*.

Leggendo la struttura delle varie relazioni, si occuperà di effettuare le *query* di creazione delle varie tabelle con i relativi attributi.

## caricaDati

```

def caricaDati(cursor, path, nomeEntita, numeroAttributi):
    """
    NOTA: Esiste un csv per ogni entità
    """
    queryRiga = ""
    with open(path + nomeEntita + ".csv", 'r') as csvfile:
        csv_data = csv.reader(csvfile)
        for row in csv_data:
            queryRiga = 'INSERT INTO ' + nomeEntita + ' VALUES('
            # I valori sono separati da una pipe
            vals = row[0].split("|")
            attributiDaInserire = numeroAttributi
            for v in range(len(vals)):
                if(attributiDaInserire < 1): break
                # Il primo è il numero della riga di excel
                if(v==0):continue
                # Concateno il valore della cella
                queryRiga += "'" + vals[v].replace("'", " ") + "',"
                attributiDaInserire = attributiDaInserire-1
            # Tolgo l'ultima virgola
            queryRiga = queryRiga[:-1]
            queryRiga += ")"
            cursor.execute(queryRiga)

```

Una volta creata la singola tabella, verranno caricati i dati inerenti letti dal csv denominato come la tabella. Leggendo il csv, verrà costruita la query di inserimento utilizzando i dati divisi per riga. Dato che iBench, con alcune configurazioni, genera più valori rispetto al numero di campi, è stato introdotto un controllo sul numero di valori utilizzati.

## Utilizzo

Per eseguire lo script Python bisogna utilizzare il seguente comando:

```
python3 ConnessioneDb.py pathFile.xml nomeDb
```

Dove:

- *ConnessioneDb.py*, è il nome dello script che si vuole eseguire;
- *pathFile.xml*, è il nome del file dal quale si vuole acquisire la struttura del database;
- *nomeDb*, è il suffisso del nome che verrà dato ai database che saranno creati per il source e il target schema.

## Parser

Come specificato nel paragrafo introduttivo, iBench genera uno schema logico su file XML, mentre CoDIT lavora su schemi concettuali su file XMI con una struttura mostrata nel paragrafo *Struttura XMI riconosciuta da CoDIT*. Per risolvere questo problema è stato creato un parser, che letto il file XML genera un nuovo file XMI con la struttura richiesta.

Per effettuare questa traduzione da logico a concettuale non è stato possibile effettuare un reengineering poiché lo schema logico generato da iBench viene costruito con dei nomi casuali, quindi la struttura generata non ha una semantica utile per poter effettuare un vero e proprio reengineering semi-automatico o manuale. Ciò che abbiamo potuto fare è stato semplicemente mappare ogni relazione, nello schema generato da iBench, in una entità nello schema concettuale risultante, ricostruendo attributi, chiavi primarie ed esterne dalle informazioni relative ad ogni relazione specificata nella struttura generata da iBench.

## Implementazione

Per implementare il parser è stato utilizzato il linguaggio di programmazione Python, il quale offre una soluzione finalizzata proprio all'interfacciamento con i file XML. Infatti, grazie all'utilizzo del modulo `xml.etree.ElementTree` è stato possibile leggere il contenuto del file e creare una struttura per il nuovo file XMI da generare.

Per individuare i nomi degli elementi all'interno del file XML è stato utilizzato XPath, un linguaggio di interrogazione, parte della famiglia XML, che permette di individuare i nodi all'interno di un documento.

Di seguito mostreremo le varie parti del parser in relazione alla struttura del file XML mostrata nel paragrafo *Struttura XMI riconosciuta da CoDIT*.

### Sezione iniziale

```
tree = ET.parse(sys.argv[1])
root = tree.getroot()

#struttura della prima sezione del nuovo file
newRoot = ET.Element('uml:Model')
newRoot.attrib['xmi:version'] = '2.1'
newRoot.attrib['xmlns:xmi'] = 'http://schema.omg.org/spec/XMI/2.1'
newRoot.attrib['xmlns:xsi'] = 'http://www.w3.org/2001/XMLSchema-instance'
newRoot.attrib['xmlns:uml'] = 'http://schema.omg.org/spec/UML/2.1.1'
newRoot.attrib['xsi:schemaLocation'] = 'http://schema.omg.org/spec/UML/2.1.1 http://www.eclipse.org/uml2/2.0.0/UML'
newRoot.attrib['xmi:id'] = '_XrSbENk5EdyEZoPpUv3LUw'
newRoot.attrib['name'] = 'Blank Model'
packageImport = ET.SubElement(newRoot, 'packageImport')
packageImport.attrib['xmi:type'] = 'uml:PackageImport'
packageImport.attrib['xmi:id'] = '_XrSbEdk5EdyEZoPpUv3LUw'
importedPackage = ET.SubElement(packageImport, 'importedPackage')
importedPackage.attrib['xmi:type'] = 'uml:Model'
importedPackage.attrib['href'] = 'http://schema.omg.org/spec/UML/2.1.1/uml.xml#_0'
```



## Sezione entità e attributi

```
#ottengo tutte le entità dall'xml e creo gli elementi per il nuovo file
for c in root.findall("./Schemas/SourceSchema/Relation"):
    nameEn = c.get('name')
    en = ET.SubElement(newRoot, "packagedElement") #elemento per la definizione delle entità
    en.attrib['xmi:id'] = nameEn
    en.attrib['name'] = nameEn
    en.attrib['xmi:type'] = 'uml:Class'
    en.attrib['visibility'] = 'private'

#estensione per la definizione di una entità
ext = ET.SubElement(en, 'xmi:Extension')
ann = ET.SubElement(ext, 'eAnnotations')
ann.attrib['xmi:id'] = 'activityEAnnotation' #controllare che vada bene che si mantenga sempre questo id
ann.attrib['source'] = 'http://www.eclipse.org/uml2/2.0.0/UML'
det = ET.SubElement(ann, 'details')
det.attrib['xmi:id'] = 'phpbb_f_Entity'
det.attrib['key'] = 'Entity'

#ottengo il nome della PK
namePk = root.find("./Schemas/SourceSchema/*[@name='"+nameEn+"']/PrimaryKey/Attr").text
```

```
# per ogni entità crea gli attributi relativi
for a in root.findall("./Schemas/"+tipoSchema+"/"+*[@name='\" + nameEn + "\"]/Attr"):
    attr = ET.SubElement(en, "ownedAttribute")
    # creo la sezione per identificare la PK
    if (a.find('Name').text == namePk):
        extPK = ET.SubElement(attr, 'xmi:Extension')
        annPK = ET.SubElement(extPK, 'eAnnotations')
        annPK.attrib['xmi:id'] = '_Ovi-VuPdEdy8F_b8rGg0Zw'
        annPK.attrib['source'] = 'http://www.eclipse.org/uml2/2.0.0/UML'
        detPK = ET.SubElement(annPK, 'details')
        detPK.attrib['xmi:id'] = '_Ovi-V-PdEdy8F_b8rGg0Zw'
        detPK.attrib['key'] = 'PK'

    attr.attrib['xmi:type'] = 'uml:Property'
    attr.attrib['xmi:id'] = a.find('Name').text
    attr.attrib['name'] = a.find('Name').text
    attr.attrib['visibility'] = 'private'
```

## Sezione relazioni

```
# generiamo la sezione per le relazioni tra le entità
list = root.findall("./Schemas/"+tipoSchema+"/ForeignKey")
for fk in list:
    #il replace è stato utilizzato per non avere problemi con iBench nella creazione
    #delle relazioni dato che utilizza un formato de tipo nomeEntità1 nomeEntità2_id,
    #quindi se nel nome di una delle relazioni incorre in un " " va in errore.
    nomeEnt1 = fk.find("From").get("tableref").replace(" \" \" ")
    nomeEnt2 = fk.find("To").get("tableref").replace(" \" \" ")

    if (nomeEnt1 != '' and nomeEnt2 != ''):
        rel = ET.SubElement(newRoot, 'packagedElement')
        rel.attrib['xmi:type'] = 'uml:Association'
        rel.attrib['xmi:id'] = nomeEnt1 + "___" + nomeEnt2 + "___id"
```

## Sezione finale

```
# struttura della sezione finale del nuovo file
profileApplication = ET.SubElement(newRoot, 'profileApplication')
profileApplication.attrib['xmi:type'] = 'uml:ProfileApplication'
profileApplication.attrib['xmi:id'] = '_XrSbHdk5EdyEZoPpUv3LUw'
extension2 = ET.SubElement(profileApplication, 'xmi:Extension')
ann2 = ET.SubElement(extension2, 'eAnnotations')
ann2.attrib['xmi:type'] = 'ecore:EAnnotation'
ann2.attrib['xmi:id'] = '_XrSbHtk5EdyEZoPpUv3LUw'
ann2.attrib['source'] = 'http://www.eclipse.org/uml2/2.0.0/UML'
references = ET.SubElement(ann2, 'references')
references.attrib['xmi:type'] = 'ecore:EPackage' # da provare a togliere
references.attrib['href'] = 'http://schema.omg.org/spec/UML/2.1.1/StandardProfileL2.xmi#_y2U58YinEdqtvbnfB2L_5w'
appliedProfile = ET.SubElement(profileApplication, 'appliedProfile')
appliedProfile.attrib['xmi:type'] = 'uml:Profile'
appliedProfile.attrib['href'] = 'http://schema.omg.org/spec/UML/2.1.1/StandardProfileL2.xmi#_0'

#scrivo l'albero creato sul nuovo file
newTree = ET.ElementTree(newRoot)
newTree.write('conversioneXmi.xmi')
```

## Utilizzo

Per eseguire lo script Python bisogna utilizzare il seguente comando:

```
python3 Parser.py pathFile.xml nomeFile
```

Dove:

- Parser.py, è il nome dello script che si vuole eseguire;
  - pathFile.xml, è il nome del file xml che si vuole convertire;
  - nomeFile, è il suffisso del nome che verrà dato ai file finali xmi per il source e il target schema.
- Cosa importante da ricordare è che, per non avere problemi con l'utilizzo dei file con il tool CoDIT, in nomeFile non utilizzare il segno di punteggiatura "punto".

## Riscontro sull'utilizzo di CoDIT con schemi di grandi dimensioni

Di seguito saranno riportati i test svolti durante le fasi preliminari dello studio di usabilità degli operatori iconici del tool CoDIT per verificarne il funzionamento con schemi di grandi dimensioni.

### Fasi e macchine utilizzate

Fasi per la generazione di schemi utilizzabili con CoDIT:

- *Generazione schema iBench*, fase in cui vengono generati gli schemi logici (source e target) in formato xml e i dati relativi solo allo schema source, attraverso il file di configurazione suddetto (vedi paragrafo *Generazione dati per lo schema target*);
- *Traduzione xml to xmi con parser*, fase in cui viene tradotta la struttura dello schema logico generato da iBench in xml in schema concettuale compreso da CoDIT in xmi;
- *Creazione database e caricamento dati su MySQL*, fase in cui, basandoci sulla struttura dello schema generato da iBench, creiamo un database per ogni schema (sourceSchema e targetSchema). Ogni tabella relativa al sourceSchema verrà popolata in modo automatico utilizzando i file csv generati da iBench.

Per testare le tempistiche delle suddette fasi al fine di ottenere materiale atto allo studio di usabilità sono state utilizzate tre macchine con le seguenti specifiche:

PC	CPU	RAM	Tipo memoria	Sistema operativo
1	I7-5500U 2.40GHz	8GB	HDD	Windows 10 64bit
2	Rayzen 5 2500U 2GHz	8GB	SSD	Windows 10 64bit
3	I5 2.7GHz	8GB	SSD	Mac OS Mojave

Tabella 2: Specifiche dei pc.

### Generazione schemi con iBench

Uno degli obiettivi principali dello studio che ci è stato affidato era quello di poter generare schemi “abbastanza grandi” che poi sarebbero stati utilizzati per un successivo studio di usabilità di CoDIT.

### Risultati

Per comprendere le dimensioni degli schemi adatte per un buon utilizzo del tool CoDIT, abbiamo svolto le 3 fasi o parte di esse su schemi di varie dimensioni. Di seguito verranno mostrati i diversi casi valutati.

Inizialmente, con il tool iBench sono stati generati schemi di dimensioni relativamente grandi. Di seguito sono descritte le caratteristiche di ognuno di essi, considerando che sono gli schemi più grandi che siamo riusciti a generare con le macchine a disposizione.



ID schema	#Entità	#Attributi per entità <sup>1</sup>	#Tuple	Percentuale dipendenze	Percentuale chiavi esterne
SC0	200	5	100	100	50
SC1	400	7	200	50	50
SC2	470	9	150	80	50
SC3	490	10	140	50	50

Tabella 3: Schemi che, convertiti in xmi, sono risultati di dimensioni superiori al massimo consentito.

Nella tabella seguente vengono riportati i tempi impiegati per la generazione degli schemi:

ID schema	PC1	PC2	PC3
SC0	37s	24s	37s
SC1	378s	319s	308s
SC2	546s	468s	443s
SC3	700s	err <sup>2</sup>	490s

Tabella 4: Tempi relativi alla generazione di schemi su iBench.

I suddetti schemi, successivamente alla conversione in xmi, non sono risultati adatti alla visualizzazione su CoDIT, poiché le dimensioni dei file risultavano superiori alla dimensione massima (vedi paragrafo *Limitazione sulla dimensione massima dei file caricati*).

Analizzando il codice, siamo riusciti ad aumentare la dimensione massima dei possibili file caricati e abbiamo provato a caricare lo schema più piccolo, cioè SC0. Il risultato non è stato dei migliori, in quanto il tempo impiegato risulta essere eccessivo per l'attesa (circa 15 minuti per il PC2, 23 minuti per il PC1). Inoltre, la visualizzazione risultava essere poco comprensibile e non totale dello schema, poiché la sezione per la visualizzazione è limitata. Alla luce di quanto detto, riteniamo sia poco utile generare grandi schemi per l'utilizzo di CoDIT, sia per i tempi di attesa che per la visualizzazione finale (vedi paragrafo *Limitazione della sezione di visualizzazione degli schemi*).

Abbiamo, quindi, generato altri tre schemi di dimensioni inferiori, che abbiamo ritenuto più consoni per un utilizzo con il tool CoDIT. Per ottenere schemi strutturati in modo differente, abbiamo abilitato alcune proprietà nel file di configurazione per lo schema SC5 e SC6.

ID schema	#Entità	#Attributi per entità <sup>1</sup>	#Tuple	Percentuale dipendenze	Percentuale chiavi esterne	Proprietà attivate
SC4	50	5	50	100	50	-
SC5	100	7	100	85	50	Partizionamento verticale 10
SC6	70	10	200	50	50	AddDelAttribute 5

Tabella 5: Schemi utilizzati per lo studio di usabilità.

<sup>1</sup> Per introdurre variabilità nella costruzione delle relazioni, è stata introdotta la variabilità del numero di attributi. Durante i test è stata utilizzata una variabilità di 3 attributi.

<sup>2</sup> La generazione dello schema SC3 non è stata possibile per il PC2 poiché durante la generazione dei file csv veniva saturata la RAM.

Per ogni macchina utilizzata è riportata una tabella con i tempi di esecuzione per ogni fase. I tempi che sono stati indicati sono il risultato di una media su tre esecuzioni.

#### PC1

ID schema	Generazione schema iBench	Traduzione xml to xmi con parser	Visualizzazione schema sul tool CoDIT	
			SourceSchema	TargetSchema
SC4	10s	0.3s	39s	38s
SC5	39s	0.5s	274s	312s
SC6	20s	0.5s	1838s	1825s

Tabella 6: Tempi ottenuti con il PC1.

#### PC2

ID schema	Generazione schema iBench	Traduzione xml to xmi con parser	Visualizzazione schema sul tool CoDIT	
			SourceSchema	TargetSchema
SC4	7s	1s	60s	69s
SC5	18s	1s	327s	317s
SC6	24s	1s	320s	278s

Tabella 7: Tempi ottenuti con il PC2.

#### PC3

ID schema	Generazione schema iBench	Traduzione xml to xmi con parser	Visualizzazione schema sul tool CoDIT	
			SourceSchema	TargetSchema
SC4	5s	0.08s	48s	48s
SC5	15s	0.15s	113s	46s
SC6	106s	0.13s	340s	330s

Tabella 8: Tempi ottenuti con il PC3.

Anche questi schemi, però, non sono risultati adatti. I motivi legati all'inadeguatezza degli schemi sono principalmente dovuti alla struttura dell'interfaccia grafica di CoDIT. Dopo un primo utilizzo svolto da noi tre, per la riorganizzazione degli schemi da poter usufruire per effettuare lo studio di usabilità sugli operatori iconici, abbiamo ritenuto opportuno avere un riscontro anche da altri colleghi in modo da confermare la nostra teoria. Abbiamo, quindi, sottoposto ad un gruppo di colleghi della laurea di Informatica un test sulla visualizzazione degli schemi su descritti, con la possibilità di poterli riorganizzare in modo da renderli comprensibili. I risultati non sono stati molto diversi da ciò che avevamo ipotizzato.

Il gruppo di ragazzi può essere diviso in due categorie, quelli che si arrendevano dopo pochi minuti, ritenendo la situazione troppo stressante e quelli che cercavano di completare il task assegnatogli fino alla fine, nonostante lo stress dovuto alla poca usabilità del tool.

I riscontri finali ottenuti sull'utilizzo di CoDIT sull'usabilità relativa al task assegnato si possono riassumere nei seguenti punti:

- Sarebbe utile poter effettuare zoom in e zoom out sulle singole sezioni di visualizzazione degli schemi in modo da poter avere un quadro generale dello schema oppure poter visualizzare a schermo intero le due sezioni in modo da poterle utilizzare singolarmente;
- Data la possibilità di nascondere gli attributi relativi ad un'entità con il doppio click su di essa, sarebbe utile poter spostare l'entità facendo sì che alla ricomparsa degli attributi, questi siano riposizionati nelle vicinanze dell'entità stessa;
- Una migliore gestione sul rilascio della linea di collegamento tra elementi nel caso in cui la si clicchi per sbaglio;
- La possibilità di un pulsante che permetta di annullare le operazioni effettuate involontariamente;
- Consentire la creazione di linee di collegamento nel caso in cui vengano eliminate erroneamente.

In conclusione, riteniamo che data la poca usabilità dell'interfaccia grafica di CoDIT, non permetta l'utilizzo di schemi con eccessive entità e relazioni tra esse.

## Tempistiche di esecuzione delle fasi preliminari

Successivamente ai problemi riscontrati, abbiamo definito schemi di piccole dimensioni (dalle 6-10 entità, con circa 5 attributi a testa) che sono risultati adatti allo studio di usabilità degli operatori iconici di CoDIT sia per tempi di esecuzione che per comprensibilità.

ID schema	#Entità	#Attributi per entità <sup>3</sup>	#Tuple	Percentuale dipendenze	Percentuale chiavi esterne	Proprietà attivate
SC7	2	5	100	70	100	VertPartition = 4
SC8	4	5	100	70	100	Merging = 2
SC9	2	5	100	70	100	VertPartition = 2 and Merging = 2

Per ogni macchina a disposizione sono state svolte le tre fasi preliminari (vedi paragrafo *Fasi e macchine utilizzate*) con successiva visualizzazione sul tool. Per ogni fase sono stati riportati i tempi di esecuzione, determinati da una media su tre esecuzioni. Successivamente alla visualizzazione degli schemi, questi ultimi sono stati riorganizzati in modo da renderli comprensibili, salvando gli schemi con le relative posizioni. I file risultanti saranno allegati alla documentazione. La riorganizzazione di ogni singolo schema è stata effettuata all'incirca in cinque minuti.

### PC1

ID schema	Generazione schema iBench	Traduzione xml to xmi con parser	Creazione e popolamento del database su MySQL	Visualizzazione schema sul tool CoDIT	
				SourceSchema	TargetSchema
SC7	1.93s	0.56s	10.4s	1.14s	1.80s
SC8	2.20s	0.12s	9.71s	1.61s	1.19s
SC9	2.30s	0.13s	16.10s	1.66s	1.56s

Tabella 9: Tempi relativi alle tre fasi e successiva visualizzazione su CoDIT ottenuti con il PC1.

### PC2

ID schema	Generazione schema iBench	Traduzione xml to xmi con parser	Creazione e popolamento del database su MySQL	Visualizzazione schema sul tool CoDIT	
				SourceSchema	TargetSchema
SC7	2s	0.27s	1.67s	1.23s	1.53s
SC8	1.84s	0.14s	1.99s	1.52s	1.31s
SC9	1.9s	0.14s	2.49s	1.24s	1.39s

Tabella 10: Tempi relativi alle tre fasi e successiva visualizzazione su CoDIT ottenuti con il PC2.

<sup>3</sup> Per introdurre variabilità nella costruzione delle relazioni, è stata introdotta la variabilità del numero di attributi. Durante i test è stata utilizzata una variabilità di 2 attributi.

PC3

ID schema	Generazione schema iBench	Traduzione xml to xmi con parser	Creazione e popolamento del database su MySQL	Visualizzazione schema sul tool CoDIT	
				SourceSchema	TargetSchema
SC7	2s	0.08s	0.88s	0.01s	0.02s
SC8	2.18s	0.05s	0.7s	0.02s	0.02s
SC9	2.30s	0.05s	0.78s	0.02s	0.02s

Tabella 11: Tempi relativi alle tre fasi e successiva visualizzazione su CoDIT ottenuti con il PC3.

## Studio di usabilità

### Metriche utilizzate

Al fine di una maggiore comprensione ed un'individuazione delle problematiche relative all'usabilità di CoDIT sono state utilizzate le seguenti metriche:

- *Tempo d'azione*: è il tempo totale che serve all'utente per completare l'attività. Il cronometro viene fatto scattare quando l'utente ha compreso il compito da portare a termine e viene stoppato nel momento in cui tutte le azioni sono terminate;
- *Livello di soddisfazione circa il compito e il test*: si tratta di una metrica qualitativa che consiste nel chiedere all'utente di esprimere un parere circa la difficoltà incontrata nel portare a termine il compito assegnato e nell'effettuare il test. Tale metrica è stata ricavata attraverso il questionario post esperimento;
- *Errori*: durante lo svolgimento dei task sono state contate le azioni compiute erroneamente durante l'esecuzione dell'operazione, come selezione errata di un'icona o sezioni non cliccabili, click successivi nel tentativo di attivare un'azione, trascinamento di elementi in sezioni errate, etc.

Il singolo task è stato suddiviso in varie fasi, dipendenti l'una dall'altra, per rendere l'esecuzione più chiara possibile. Il tempo d'azione, quindi, è stato calcolato sommando il tempo di esecuzione delle varie fasi poiché non sempre i candidati capivano cosa stesse succedendo e quindi avevano bisogno di alcuni chiarimenti.

All'interno del questionario post esperimento sono state riportate domande riguardanti le fasi di ogni task, in modo da permettere al candidato di esprimere il proprio grado di soddisfazione circa l'esecuzione del task.

### Le fasi della procedura

Lo studio dell'usabilità è stato condotto utilizzando le macchine sopra descritte (paragrafo *Fasi e macchine utilizzate*). La procedura è composta dalle seguenti fasi:

1. Preparazione;
2. Esecuzione;
3. Analisi dei risultati.

#### Preparazione

Questa fase prevede i seguenti aspetti:

- analisi preliminare del tool;
- quali operatori iconici utilizzare in base agli schemi generati;
- quanti utenti selezionare e quali tipologie di utenti scegliere;
- quali e quanti task preparare;
- come preparare i moduli per la raccolta dati;
- cosa fare prima dell'osservazione: il test pilota.

#### Analisi preliminare del tool

Essendo un tool a noi sconosciuto per avere una conoscenza sufficiente per poter illustrare al meglio le funzionalità offerte, abbiamo svolto un utilizzo preliminare. Inoltre, essendo sperimentale, abbiamo testato quali fossero le operazioni realmente funzionanti.

### *Quali operatori iconici utilizzare in base agli schemi generati*

Valutando gli schemi generati con il tool *iBench*, abbiamo selezionato gli operatori iconici utilizzabili durante il test di usabilità.

Gli operatori iconici selezionati con i rispettivi inversi (che non sono però riportati) sono i seguenti:

#### Mapping di attributi



Da utilizzare nel caso in cui lo stesso concetto sia stato modellato da una parte con un attributo semplice e dall'altro con uno derivabile.



Da utilizzare nel caso in cui lo stesso concetto sia stato modellato da una parte con un attributo semplice e dall'altro da uno composto.



Da utilizzare nel caso in cui lo stesso concetto sia stato modellato da una parte con un attributo semplice e dall'altro con una relazione tra due entità.

#### Partizionamento



Da utilizzare nel caso in cui la relazione dello schema di destra rappresenti una decomposizione verticale dell'entità dello schema di sinistra.

### *Quanti utenti selezionare e quali tipologie di utenti scegliere*

Essendo un tool relativo ai database e all'integrazione degli stessi, abbiamo ritenuto opportuno selezionare utenti del Corso di Laurea di Informatica, sia della triennale che della magistrale. Il numero di utenti selezionati è stato 10 per avere un campione variegato in modo da ottenere abbastanza dati da poter analizzare.

#### *Quali e quanti task preparare*

I task da far svolgere durante il test sono stati preparati basandoci sugli operatori iconici scelti nella fase precedente (paragrafo *Quali operatori iconici utilizzare in base agli schemi generati*). Quindi, abbiamo preparato quattro task, uno per ogni operatore iconico.

#### Task 1

**Situazione:** ti trovi in una situazione in cui in uno schema hai un attributo semplice e nell'altro un attributo derivato. Quale operatore iconico utilizzeresti per mantenere una configurazione piuttosto che un'altra?

**Schema da utilizzare:** SC7.

**Informazioni preliminari:** l'entità coinvolta è cheese\_cp\_1\_nI0\_ce0.

*Elementi da selezionare:* Nello schema source, nell'entità cheese\_cp\_1\_nI0\_ce0 hai l'attributo ranch\_cp\_1\_nI0\_ae2 come attributo semplice. Nello schema target, medesima entità, hai l'attributo compare\_cp\_1\_nI0\_ae1 derivabile dall'attributo branch\_cp\_1\_nI0\_ae2.

*Task 1.1:* Mantenere la configurazione con l'attributo semplice.

*Task 1.2:* Mantenere la configurazione con l'attributo derivabile.

*Task 1.3:* Visualizza gli elementi selezionati.

*Task 1.4:* Annulla l'operazione appena eseguita.

*Task 1.5:* Elimina l'operatore iconico utilizzato.

## Task 2

*Situazione:* ti trovi in una situazione in cui in uno schema hai un attributo semplice e nell'altro un attributo composto. Quale operatore iconico utilizzeresti per mantenere una configurazione piuttosto che un'altra?

*Schema da utilizzare:* SC7.

*Informazioni preliminari:* l'entità coinvolta è test\_cp\_0\_nI0\_ce0.

*Elementi da selezionare:* Nello schema source, nell'entità test\_cp\_0\_nI0\_ce0 hai gli attributi slope\_cp\_0\_nI0\_ae2, touch\_cp\_0\_nI0\_ae4 e measure\_cp\_0\_nI0\_ae3 che rappresentano parti dell'attributo measure\_cp\_0\_nI0\_ae3 dello schema source. Nello schema target, medesima entità, hai l'attributo measure\_cp\_0\_nI0\_ae3 come attributo semplice.

*Task 2.1:* Mantenere la configurazione con l'attributo semplice.

*Task 2.2:* Mantenere la configurazione con l'attributo composto.

## Task 3

*Situazione:* ti trovi in una situazione in cui in uno schema hai un attributo semplice e nell'altro è rappresentato da due entità relazionate tra loro. Quale operatore iconico utilizzeresti per mantenere una delle due configurazioni?

*Schema da utilizzare:* SC7.

*Informazioni preliminari:* le entità coinvolte sono red\_vp\_1\_nI0\_ce0, tail\_vp\_0\_nI0\_ce1 e warn\_vp\_0\_nI0\_ce0.

*Elementi da selezionare:* Nello schema source, nell'entità red\_vp\_1\_nI0\_ce0 hai l'attributo describe\_vp\_1\_nI0\_ae5 come attributo semplice. Nello schema target invece, l'attributo describe\_vp\_1\_nI0\_ae5 suddetto è stato modellato sulle entità tail\_vp\_0\_nI0\_ce1 e warn\_vp\_0\_nI0\_ce0, relazionate attraverso tail\_vp\_0\_nI0\_ce1/warn\_vp\_0\_nI0\_ce0.

*Task 3.1:* Far sì che anche nel target sia mantenuto l'attributo semplice.

*Task 3.2:* Far sì che anche nel source l'attributo semplice sia scomposto in due entità come accade nel target.

## Task 4

*Situazione:* ti trovi in una situazione in cui in uno schema hai un'unica entità e nell'altro è rappresentata da due entità relazionate tra loro. Quale operatore iconico utilizzeresti per mantenere una delle due configurazioni?

*Schema da utilizzare:* SC9.



*Informazioni preliminari:* le entità coinvolte sono cheese\_cp\_1\_nI0\_ce0 e indicate\_vp\_0\_nI0\_ce1.

*Elementi da selezionare:* Nello schema target, l'entità cheese\_cp\_1\_nI0\_ce0 dello schema source, è descritta dalle due entità cheese\_cp\_1\_nI0\_ce0 e indicate\_vp\_0\_nI0\_ce1 relazionate da cheese\_cp\_1\_nI0\_ce0/indicate\_vp\_0\_nI0\_ce1.

*Task 3.1:* Far sì che anche nel target sia mantenuto l'attributo semplice.

*Task 3.2:* Far sì che anche nel source l'attributo semplice sia scomposto in due entità come accade nel target.

#### *Come preparare i moduli per la raccolta dati*

Ai candidati sono stati sottoposti due questionari: uno prima dell'esperimento ed uno dopo.

Il primo questionario ha il compito di raccogliere informazioni statistiche circa il candidato e le sue conoscenze riguardo l'utilizzo di database e delle varie operazioni possibili su attributi, entità e relazioni. Dato che CoDIT è un'applicazione web e presenta alcune operazioni descritte in lingua inglese, è stato chiesto il grado di conoscenza relativo a queste ultime competenze.

L'ultimo questionario raccoglie i pareri dei candidati riguardo l'usabilità del tool ed è diviso in tre sezioni:

- *Valutazione generica dell'usabilità* utilizzando il metodo SUS (System Usability Scale);
- *Valutazione delle singole sezioni del tool;*
- *Suggerimenti e motivazioni.*

Il metodo SUS è composto da dieci affermazioni di cui il candidato, tramite una scala *likert* da 1 a 5, esprime il grado di approvazione. In particolare, le affermazioni di posizione pari rispecchiano un aspetto negativo; positivo altrimenti.

Successivamente vengono valutati gli aspetti legati all'utilizzo degli operatori iconici e ad alcune funzionalità annesse.

L'ultima sezione permette al candidato di esprimere la propria opinione e le proprie perplessità riscontrate durante il test che non si evincono dalle sezioni precedenti.

#### *Cosa fare prima dell'osservazione: il test pilota*

Prima di avviare la fase di testing con tutti i candidati, è stata prevista una sessione pilota. Attraverso questa sessione abbiamo compreso le tempistiche e le modalità con cui avremmo dovuto somministrare ogni task.

Ogni task è stato spiegato in modo diverso per trovare quello più chiaro. Inoltre, sono state confermate le metriche precedentemente scelte.

Attraverso il test pilota abbiamo capito che bisognava prima mostrare tutti gli operatori iconici e successivamente spiegare il task in modo da far sentire il candidato a proprio agio nella scelta dell'operatore corretto. Inoltre, abbiamo compreso come richiedere l'esecuzione di un'operazione senza suggerire l'elemento dell'interfaccia corretto.

Una volta completato il test pilota, abbiamo proseguito con gli altri candidati.

#### *Esecuzione*

L'esecuzione del test si scompone nelle seguenti fasi:

1. Compilazione questionario pre-esperimento, per la raccolta dei dati statistici come le conoscenze preliminari;

2. Lettura e comprensione delle slide che racchiudono le conoscenze preliminari alla comprensione degli operatori;
3. Spiegazione dello svolgimento delle successive fasi al candidato;

Le successive fasi verranno svolte unicamente dal candidato, senza nessun supporto dove possibile.

4. Selezione di due elementi casuali in modo da visualizzare la lista degli operatori iconici;
5. Individuazione dell'operatore iconico appropriato in base alla situazione spiegata;
6. Selezione degli elementi coinvolti indicati dai facilitatori;
7. Selezione dell'operatore appropriato;
8. Selezione della configurazione giusta in base all'operazione indicata dai facilitatori;
9. Completamento procedura aggiuntiva, ove prevista;
10. Utilizzo dei tasti relativi all'eliminazione dell'operatore, visualizzazione degli elementi coinvolti e modifica della configurazione, dove richiesto.

## Analisi dei risultati

### Questionari

La prima sezione del questionario post-esperimento, creata seguendo il metodo System Usability Scale per valutare in generale l'usabilità di CoDIT, risulta prevalentemente negativa. La maggioranza degli utenti, infatti, ha ritenuto il sistema poco confortevole nell'utilizzo non trovandosi a proprio agio.

Nella seconda parte, relativa alle singole sezioni del tool, sono state riportate le sensazioni su singoli aspetti riscontrati durante l'utilizzo. In particolare, sui colori utilizzati, le icone relative all'eliminazione dell'operatore, la visualizzazione degli elementi selezionati e l'annullamento dell'operazione, la comprensione delle icone relative agli operatori iconici e i relativi pop-up per il completamento dell'operazione. I risultati sono stati positivi rispetto alle icone relative agli operatori iconici, meno positivi per quanto riguarda gli altri aspetti.

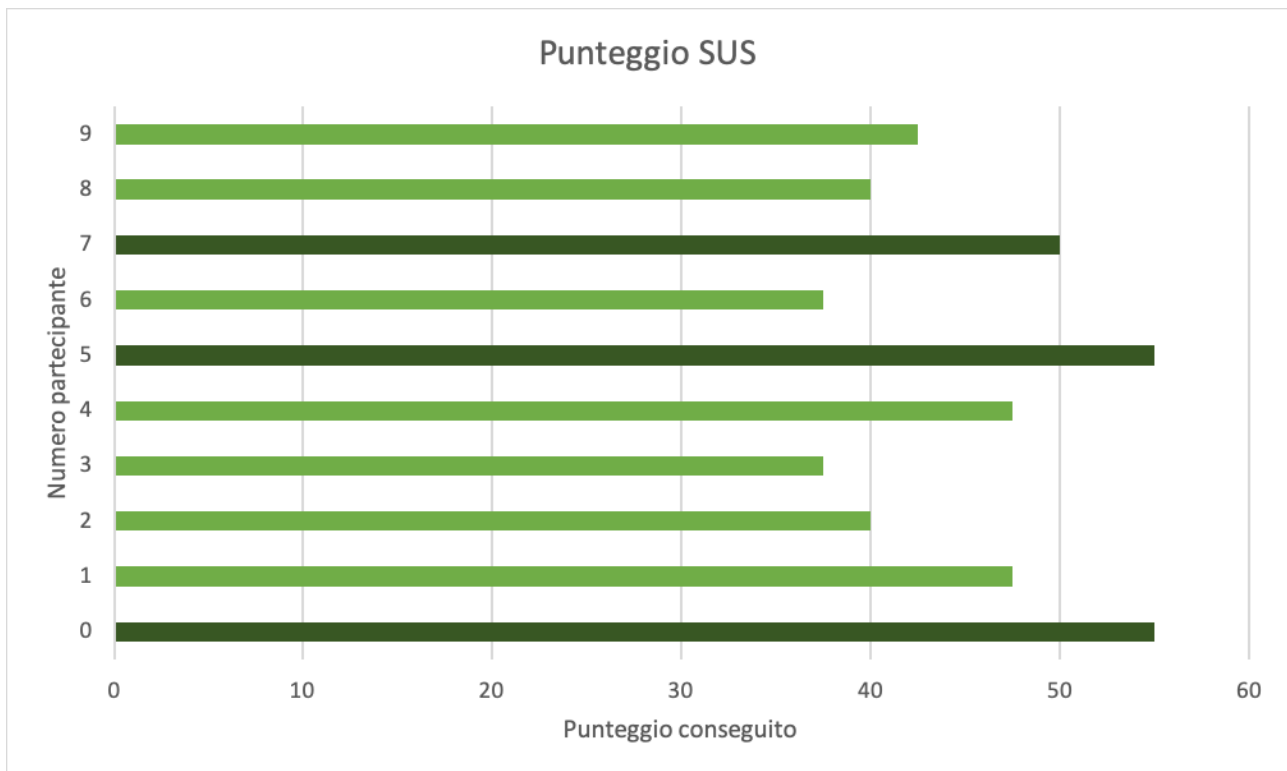
Nell'ultima sezione, infine, abbiamo richiesto agli utenti di esprimere le loro opinioni riguardanti l'utilizzo generale del tool e ipotetici consigli su come migliorare l'usabilità dell'interfaccia.

Per elaborare il risultato relativo alla prima sezione del questionario, essendo basato sul metodo SUS, abbiamo utilizzato la relativa procedura per ricavarne il punteggio complessivo:

- Sommare i punti di tutte le domande con posizione dispari ed in fine sottrarre 5;
- Sottrarre a 25 la somma di tutti i punteggi delle domande di posizione pari;
- Sommare i risultati dei primi due punti e moltiplicare per 2.5.

Il risultato ottenuto, pur non essendo una percentuale, è compreso tra 0 e 100. Si possono considerare casi positivi tutti i valori superiori a 49; negativi altrimenti.

Di seguito verrà mostrato un grafico che riassume il risultato ottenuto.

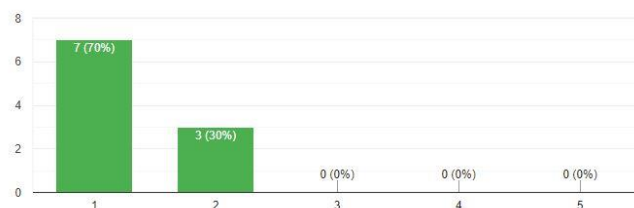


Le barre evidenziate indicano gli unici casi positivi, quindi nel complesso questo risultato indica una poca usabilità del tool CoDIT.

Per quanto riguarda la seconda sezione, riportiamo solo le risposte più significative relative agli aspetti che sono risultati negativi. In particolare, mostreremo i risultati riguardo l'utilizzo dei colori, l'icona dell'annullamento dell'operazione e i pop-up relativi al completamento delle operazioni. Per ognuno dei quesiti è stata richiesta una possibile motivazione di cui riporteremo un sunto complessivo oppure la più esplicativa.

L'utilizzo dei colori è stato indicativo per lo svolgimento delle operazioni da effettuare?

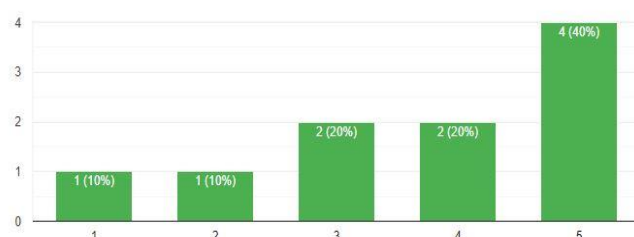
10 risposte



*“Il concetto alla base (colorare il riquadro dove si trova il cursore) è giusto ma l'utilizzo dei colori non è appropriato. Sarebbe più giusto usare altri colori che però non confondano l'utente, poiché verde e rosso vengono usati per confermare o annullare un'azione.”*

L'icona mostrata è risultata chiara per svolgere l'azione associata?

10 risposte



*“L'icona rappresenta un'operazione di cancellazione che, associata all'operazione di selezionamento delle scelte delle configurazioni, porta il candidato ad intuire in maniera corretta il suo funzionamento.”*

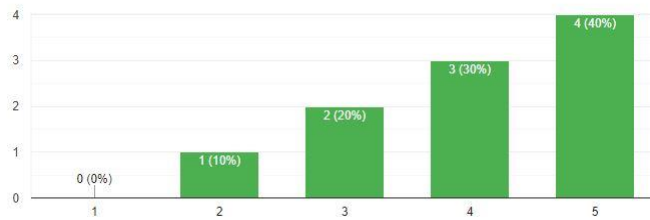
*“Credo che l'icona con il cestino non sia adeguata per l'operazione intesa: la cambiarei*

con un pulsante con testo che indica che cosa accade quando viene premuto.”

L'icona mostrata è risultata chiara per svolgere l'azione associata?



10 risposte

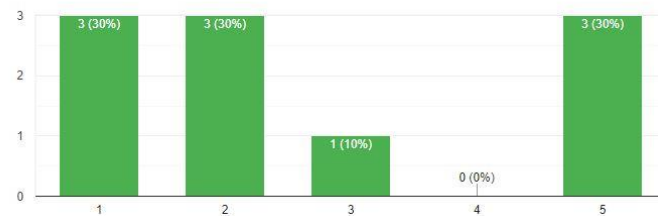


“Metterei questa icona non sotto gli operatori ma sopra i grafi.”

L'icona mostrata è risultata chiara per svolgere l'azione associata?



10 risposte

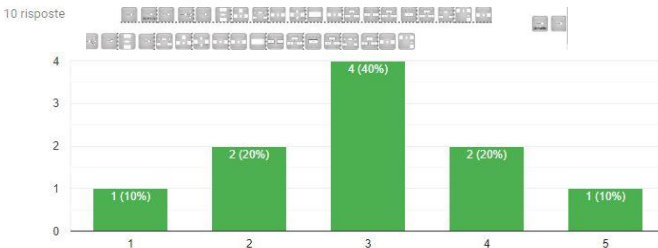


“Perché una matita dovrebbe indicare lo stato precedente?”

“Avrei preferito una gomma al posto della matita, perché io immaginavo che mi facesse modificare un'operazione appena effettuata.”

E' risultata chiara la disposizione degli operatori?

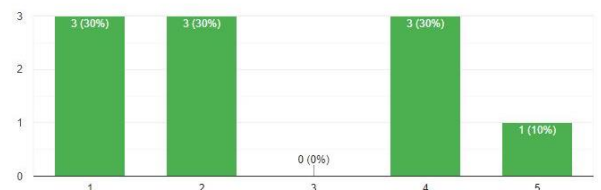
10 risposte



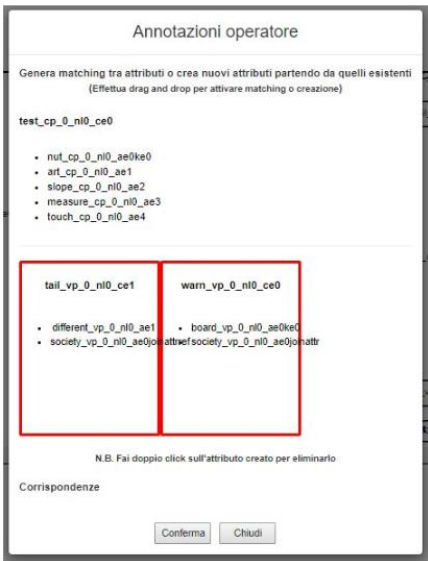
“Confusionaria, sarebbe meglio suddividerle in sezioni separate in base alla categoria. Inoltre, sarebbe meglio implementare la possibilità di invertire l'ordine delle configurazioni così da risparmiare icone e migliorare la gestione dello spazio.”

Dopo aver selezionato l'operatore di destra, le operazioni da effettuare in questa schermata ti sono risultate chiare ed intuitive? \*

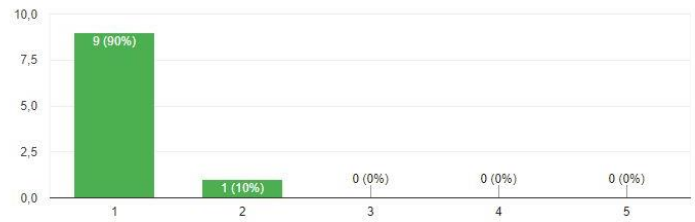
10 risposte



Dopo aver selezionato l'operatore di destra, le operazioni da effettuare in questa schermata ti sono risultate chiare ed intuitive? \*



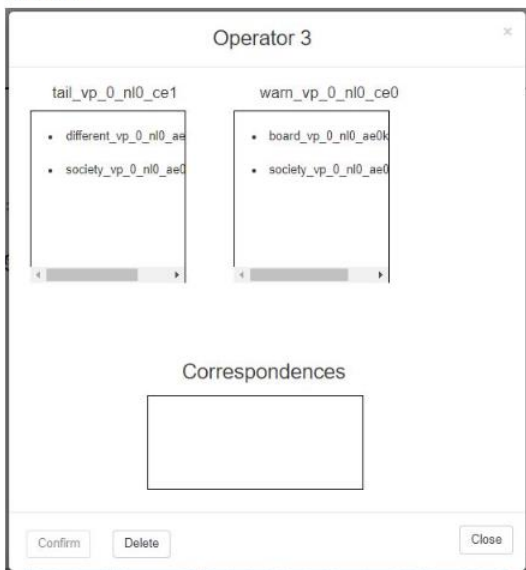
10 risposte



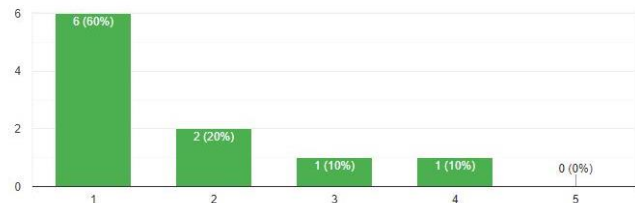
*“Guardando l'interfaccia, ho fatto fatica a capire come dovessi portare a termine l'operazione, portandomi a dimenticare non solo cosa dovessi fare, ma anche come farlo.”*

*“Non sono chiare le operazioni da eseguire, non vi sono descrizione che orientino l'utente nel corretto funzionamento dell'operazione.”*

Dopo aver selezionato l'operatore di destra, le operazioni da effettuare in questa schermata ti sono risultate chiare ed intuitive? \*



10 risposte



*“Nessuna indicazione o suggerimenti non ben comprensibili o all'apparenza non del tutto corretti. Del resto interfaccia molto mediocre. Utilizzo dei colori confusionale.”*

Nell'ultima sezione sono state richiesti suggerimenti o perplessità riguardo le funzionalità provate. Di seguito riportiamo le più significative.

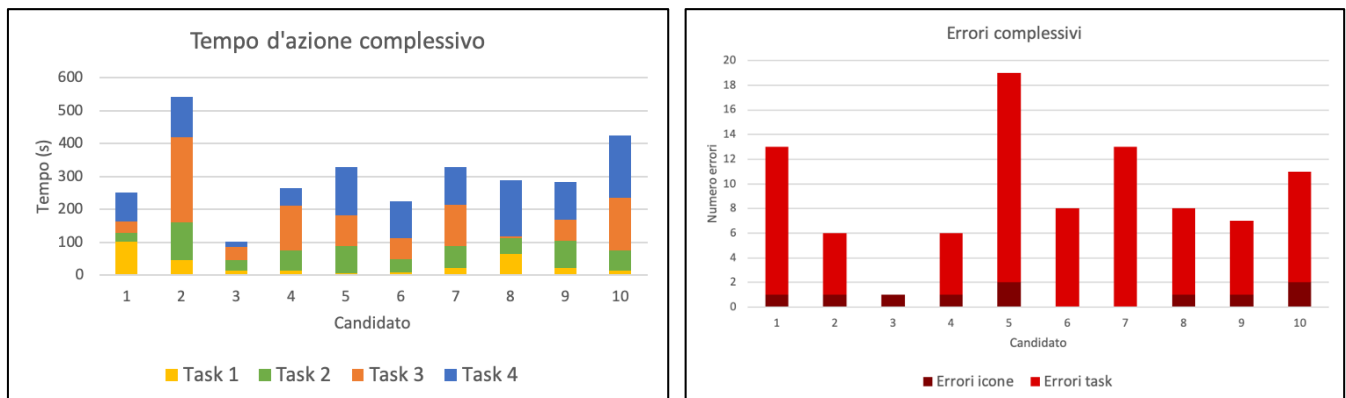
1. Suggerirei una revisione dell'usabilità generale del sistema
2. Inserirei messaggi di notifica per le operazioni più chiari
3. I colori risultano utilizzati male per le loro funzioni reali
4. Il "close" nelle finestre relative agli operativi non annulla le operazioni effettuate.
5. Fisserei l'interfaccia con un'unica lingua o la possibilità di alternarla come tutti i normali software.
6. Alcuni tasti ridondanti o confusionari.”

*“Funzionalità di ingrandire i diagrammi ER in modo da avere una visione globale del sistema, rendere i tool di modifica più chiari andando a inserire descrizioni d'uso, simboli e una migliore gestione del colore dato che, a volte, risultava disorientante. Rendere la disposizione delle componenti del diagramma più compatte e ordinate dopo una modifica effettuata, rendere le icone delle possibili modifiche ordinate per qualsiasi risoluzione.”*

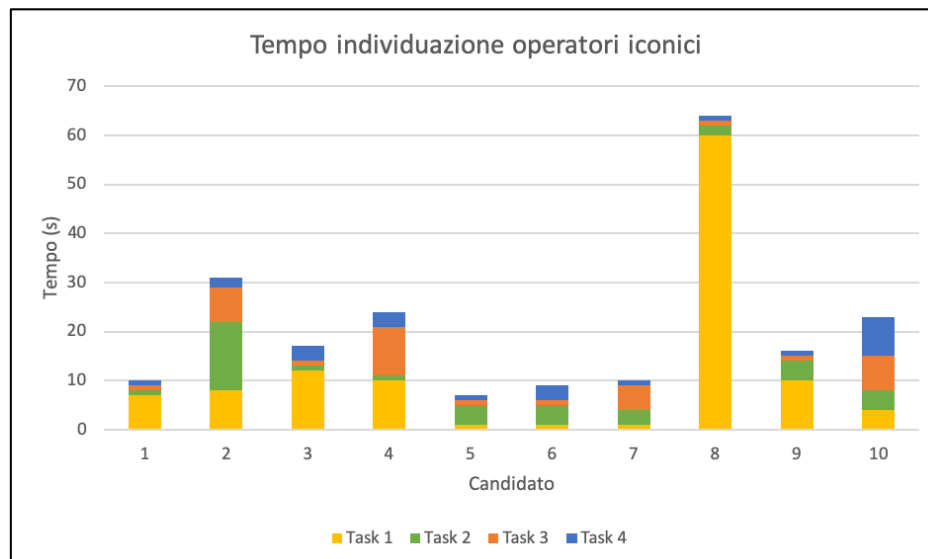
*“Inserire uno wizard che, almeno sui pop up, così da guidare l'utente durante l'operazione.”*

#### Tempi di esecuzione ed errori

Di seguito verranno mostrati i tempi totali di esecuzione relativi ai quattro task somministrati ai candidati. Inoltre, saranno mostrati anche il totale degli errori commessi.



In particolare, mostriamo i tempi di individuazione degli operatori iconici di ogni candidato suddivisi per task.



I risultati ottenuti mostrano che in media il tempo è molto basso, tranne in alcuni casi isolati.

Lo studio di usabilità svolto su CoDIT non è stato possibile su degli schemi di grandi dimensioni per via della mancata gestione di un numero elevato di entità, e dove presente non ben fatta. Utilizzando iBench sono stati generati schemi man mano più grandi per testare quale fosse il limite di CoDIT. Purtroppo, oltre al tempo eccessivo impiegato per il caricamento del file xmi sul tool, nemmeno la visualizzazione era possibile. Per effettuare il test di usabilità abbiamo quindi deciso di utilizzare degli schemi piccoli e sui quali era possibile applicare alcuni degli operatori offerti da CoDIT.

Relativamente al task 2, pensiamo che ove è possibile poter selezionare una sola opzione è inutile chiedere l'azione all'utente: dopo aver selezionato gli attributi da unificare non è necessario dover selezionare l'operazione di concatenazione/scissione, dato che è l'unica operazione possibile ed è richiesta per completare il task. Spesso, il candidato si soffermava sulla possibile azione da compiere non capendo il motivo della possibilità di selezione.

Riguardo il task 4, per i candidati era chiara l'azione dell'operatore ma non il procedimento richiesto dall'interfaccia, dato che l'operazione di trascinamento non era ben chiara: gli attributi trascinabili e non trascinabili erano visualizzati in modo identico; mentre la zona dove dovevano essere rilasciati non era indicata quindi i candidati tentavano in diverse zone sentendosi confusi.

39

L'oggetto principale dello studio di usabilità sono stati gli operatori iconici. Il risultato ottenuto è stato positivo. Le icone utilizzate sono risultate chiare e ben esplicative della situazione che rappresentavano. La maggior parte dei candidati non ha avuto problemi nell'identificazione, esclusi alcuni casi in cui sbagliavano la selezione, non per la rappresentazione dell'operatore, ma per la disposizione di quest'ultimo. Infatti, anche dai questionari somministrati è risultato che la disposizione degli operatori era confusionaria, portando ad impiegare maggior tempo per la ricerca (come mostrato dal grafico nel paragrafo *Tempi di esecuzione ed errori*). Inoltre, è risultata poco chiara la suddivisione utilizzata per separare un operatore iconico dall'altro.

In conclusione, riteniamo che le icone utilizzate per gli operatori iconici siano state ben realizzate, per il resto ci sono molte criticità a livello di interfaccia. Ciò è stato riscontrato sia dai questionari post-esperimento che dall'utilizzo preliminare che abbiamo dovuto svolgere non conoscendo il tool. Infatti, anche per noi facilitatori è stato difficile in alcune situazioni comprenderne l'utilizzo, che risultava chiaro solo dopo un minimo di tre tentativi, date le poche o caotiche indicazioni. Inoltre, avendo dovuto utilizzare CoDIT anche per la visualizzazione degli schemi, dai più grandi ai finali, ci siamo resi conto della poca praticità/usabilità del tool relativa alla visualizzazione e riorganizzazione degli schemi.



## Indice delle tabelle

<i>Tabella 1: Tabella relativa agli schemi nella fase iniziale dell'utilizzo di CoDIT.</i>	14
<i>Tabella 2: Specifiche dei pc.</i>	24
<i>Tabella 3: Schemi che, convertiti in xmi, sono risultati di dimensioni superiori al massimo consentito.</i>	25
<i>Tabella 4: Tempi relativi alla generazione di schemi su iBench.</i>	25
<i>Tabella 5: Schemi utilizzati per lo studio di usabilità.</i>	25
<i>Tabella 6: Tempi ottenuti con il PC1.</i>	26
<i>Tabella 7: Tempi ottenuti con il PC2.</i>	26
<i>Tabella 8: Tempi ottenuti con il PC3.</i>	26
<i>Tabella 9: Tempi relativi alle tre fasi e successiva visualizzazione su CoDIT ottenuti con il PC1.</i>	28
<i>Tabella 10: Tempi relativi alle tre fasi e successiva visualizzazione su CoDIT ottenuti con il PC2.</i>	28
<i>Tabella 11: Tempi relativi alle tre fasi e successiva visualizzazione su CoDIT ottenuti con il PC3.</i>	29