CCISTARML(v0.6):
A Java Package for handling iStarML files

Carlos Cares
ccares@lsi.upc.edu
Technical University of Catalonia (UPC).

Contents

1 Introduction

The *i** (istar) is a modelling framework proposed in [9] to deal with intentional organizational situations, focused on goal and actors. Therefore it has been recognized as a relevant modelling language for requirements engineering [8]. Moreover, it has been used such the modelling language of Tropos [2, 5], a relevant agent-oriented software development methodology. Both main applications of *i** have implied that the framework has become very popular on the respective scholar communities: requirements engineering and agent-oriented software engineering.

Like a natural progression, many scholars have proposed to go deep on specific issues around the *i** framework. Others have worked on to extend its semantic. Therefore, today, there are different variations of the *i** framework. The main variations and a reference model which include these variations is discussed in [3].

These different scholar teams have produced an interesting set of *i** tools [7]. However, a common problem on the *i** community have been the problem of sharing models. A proposal trying to find a method to share *i** models is the iStarML proposal [4] which is based on the XML syntax [6] and it aims is to generate a common mark up language.

This Java package allows handling iStarML expressions and files [4]. iStarML is a xml-based language to represent different variations of *i** diagrams.

2 The ccistarml v0.6 Functionality

The ccistarml v0.6 java package allows creating, importing and handling istarml files complaint with the iStarML version 1.0 proposal. The package was programmed using jdk1.5.0_11 and the NetBeans IDE 5.5.

This package allows checking the xml syntax and the specific istarml syntax. Moreover it allows handling and creating the istarml structure by using a reduced set of Java classes. The Java classes and their functionality appear in Table 1. Each one of these classes is explained and illustrated using simple examples.

Table 1. Main classes of the ccistarml v0.6 Package

Class	Functionality
ccistarmlFile	Encapsulates an abstract functionality for handling iStarML files. It allows XML parsing and iStarML parsing separately. Two different methods allow handling the iStarML data structure. On the other hand it provides the basic functionality to create an iStarML file.
ccfileError	An object of this class represents an error on the parsing of the iStarML file
ccistarmlContent	This class supports a dynamic data structure which represents the iStarML file structure. Each object of this class represents a tag in the iStarML structure. Nested structures represent nested tags.
ERelementList	This class handles a list which contains the iStarML elements in an entity-relationship representation. In this list there are not nested structures, all elements are represented by a unique identifier and relationships among iStarML elements are represented like nodes in the same list. The nodes of this list are objects of the ERelement class.
ERelement	An object of this class represents an element of the iStarML file or a relationship between two elements of the same file.

3 The ccistarmlFile Class

3.1 Constructing an ccistarmlFile object

The main class of the ccistarml Java package is the class ccistarmlFile. It allows handling the basic operations of parsing and importing an iStarML file. The construct has two options, to specify a string which contains the iStarML (XML) content; the other option is to create an empty file object and loading the content of a text file after that.

ccistarmlFile			
Method	Description	Parameters	Returns
ccistarmlFile (constructor)	Creates an object which represents a new (optionally empty) iStarML file	Without parameters 1: String XMLContent	ccistarmlFile
Example			
import ccistarml.*; ⋮ ccistarmlFile my_file = new ccistarmlFile("<?xml version='1.0'> ⋮ ");			
Observation			
The object is created by specifying the initial iStarML (XML) content of the file. If there are not specified parameters then the file starts empty.			

3.2 Loading an external iStarML File.

ccistarmlFile			
Method	Description	Parameters	Returns
loadFile	Takes the content of an external XML file and put it such the new content of the iStarML file	1: String fileName	ccistarmlFile
Example			

import ccistarml.*;
⋮
ccistarmlFile my_iFile = new ccistarmlFile();
my_iFile.loadFile("test_iStarMLfile.istarml");			

3.3 Verifying the XML content

The xml content is verified separately of the istarml content. These are separated process which can be invoked by using a ccistarmlFile object. The method xmlParser allows verifying the XML content. We have implemented a Finite Determinist Automata (FDA) in order to do this job. This process not only verifies the xml syntax but also build a Java data structure which lately allows handling the istarml content.

ccistarmlFile			
Method	Description	Parameters	Returns
xmlParser	Verifies the XML content	Without parameters	boolean

Example

```
import ccistarml.*;
  .
  .
  .
ccistarmlFile my_iFile = new ccistarmlFile();
my_iFile.loadFile("testFile.istarml");
boolean result = my_iFile.xmlParser();
```

Observation

The result of the xmlParser indicates the resulting of the parsing process. If the XML content is correct then the result is true, other case the result is false.

3.4 Verifying the iStarML Content

The iStarML content should be verified once a time the XML content has been checked. These are separated process which can be invoked by using a ccistarmlFile object. The method istarmlParser allows verifying the iStarML content. We have implemented a set of method which verify the syntax and the use of tags' references in the iStarML file.

ccistarmlFile			
Method	Description	Parameters	Returns
istarmlParser	Verifies the iStarML content	Without parameters	boolean

Example

```
import ccistarml.*;
  .
  .
  .
ccistarmlFile my_iFile = new ccistarmlFile();
my_iFile.loadFile("testFile.istarml");
boolean resultXML = my_iFile.xmlParser();
if (resultXML) if(my_iFile.istarmlParser())
        System.out.println("All OK");
```

Observation

The result of the istarmlParser is a boolean result which indicates the true value resulting of the parsing process. If the iStarML content is correct then the result is true, other case the result is false.

3.5 Displaying Errors

When the result of verification process (XML or iStarML) results on false, then it is possible to display the errors on the console by using the method displayErrors. The method errors reports the total mount of detected errors either the XML parsing or the iStarML parsing. The method do not stop the parsing when an error is found, therefore, in some cases some specific error drags additional errors. On the next table we show two cases; in the first one we have an iStarML file with two XML errors and one iStarML error. We run the example program one time and we have an XML errors' report. Then we correct the XML errors and we run the program again. The second time, using the same code lines, we have an iStarML errors' report.

ccistarmlFile

Method	Description	Parameters	Returns
displayErrors	Verifies the iStarML content	Without parameters	boolean

Example

```
ccistarmlFile my_iFile = new ccistarmlFile();
my_iFile.loadFile("testFile.istarml");
boolean allOk;
if (allOk = my_iFile.xmlParser())
        if(allOk = my_iFile.istarmlParser())
                System.out.println("All OK");
if(!allOk) {
    System.out.println(iStarMLfile.errors()+" errors");
    my_iFile.displayErrors(); }
```

The source file with two XML errors (dark marks)

```
1   <?xml version="1.0"?>
2   <istarml version="1.0">
3    <diagram name="sample 6.2">
4     <actor>
5      <boundary>
6       <ielement type="goal"
7                 name="Supervise students' career"/>
8       <ielement type="task"
9                 name="Send personal email"/>
10     </boundary>
11    </actors>
12    </diagram>
13  </istarml>
```

First Run: Reporting XML errors

```
run:
3 errors
Line 3: < Error:expected tag name
Line 11: </actors> Error:expected </actor>
Line 12: </diagram> Error:expected </istarml>
```

Second Run after correcting XML errors but reporting iStarML errors

```
run:
...xml parsing ok
1 errors
Line 4: <actor> Error:an actor must have
        id or name attribute
```

3.6 Getting a List of Errors

For handling the error's report in GUI environment results convenient to get independently each error. The method errorList returns a LinkedList (java.util. package) which contains a list of objects of the class ccfileError whose hold the information corresponding with the errors in the parsed file. In section 5 is showed how handle with the information of each error.

ccistarmlFile			
Method	Description	Parameters	Returns
errorList	Get a linked list which contains the iStarML errors (ccfileError objects)	Without parameters	LinkedList (java.util.LinkedList)
Example			
LinkedList errs = my_iFile.errorList(); System.out.println("There are "+errs.size()+" errors");			
Observation			
The operation on ccfileError objects is reviewed in section 5			

3.7 Getting and Handling the XML Content

The class ccistarmlFile allows getting the XML structure in a dynamic structure. This structure is constituted by a set of connected objects of the class ccistarmlContent, which is explained in the section 6. In the meanwhile we can say that this content is returned by the public method xmlStructure. The result is an object of the class ccistarmlContent.

ccistarmlFile			
Method	Description	Parameters	Returns
xmlStructure	Get a data structure which contains the complete XML structure of the file	Without parameters	ccistarmlContent
Example			
ccistarmlContent s = my_iFile.xmlStructure();			
Observation			
The operations on ccistarmlContent objects is reviewed in subsection 6.2			

3.8 Creating and Saving an iStarML file

To create a new iStarML file the class ccistarmlFile has a set of methods for handling this kind of documents. However it is necessary to combine the use with some methods of the class ccistarmlContent which adds and modifies specific XML elements. Therefore general file creation and saving options are functions of the ccistarmlFile class while that the creation and modification of the iStarML elements are functions of the ccistarmlContent class.

ccistarmlFile			
Method	Description	Parameters	Returns
saveFile	Save the iStarML content on a XML format in the specified text file	1:String fileName	boolean
Example			

ccistarmlFile f = new ccistarmlFile();
⋮ // creation and/or modification sentences
⋮
f.saveFile("test001.istarml");			

Observation			
The creation and modification of operations on the istarml Content are part of the operations reviewed in section 6.			

3.9 Getting and Displaying the XML content

An object of the ccistarmlFile class can produce a String containing the corresponding XML code by using the buildXML method. This method produces an indented and structured string containing the XML code corresponding with the internal structure of the file. Therefore in order to get a well-formed displaying of the file it is enough to print this String to the output

ccistarmlFile			
Method	Description	Parameters	Returns
buildXML	Build the text content corresponding to the internal iStarML structure	Without parameters	String
Example			
ccistarmlFile f = new ccistarmlFile(); ⋮ System.out.println(f.buildXML());			
Observation			
In the next table it is showed a comparative output between this method (buildXML) and displayXML.			

In order to get a console display we provide an additional method namely displayXML. This method displays a code similar to the gotten code from the buildXML method; however it uses different lines for each attribute tag.

ccistarmlFile			
Method	Description	Parameters	Returns
displayXML	Get a console output of the XML content of the iStarML file	Without parameters	void

Example

```
ccistarmlFile f = new ccistarmlFile();
f.loadFile("smallTest.istarml");
f.xmlParser();
System.out.println(f.buildXML());
System.out.println("----------");
f.displayXML();
```

The input iStarML file

```
1  <?xml version="1.0"?>
2  <istarml version="1.0">
3    <diagram name="My first i* diagram" author="Me">
4      <actor name="University"/>
5    </diagram>
6  </istarml>
```

Console output

```
run:
<?xml version="1.0"?>
<istarml version="1.0">
    <diagram name="My first i* diagram" author="Me">
        <actor name="University"/>
    </diagram>
</istarml>
----------
    <xml/>
    version=1.0
    <istarml>
    version=1.0
        <diagram>
        name=My first i* diagram
        author=Me
            <actor/>
            name=University
        </diagram>
    </istarml>
```

3.10 Removing Diagrams from the istarml File

To remove a specific diagram from the file the ccistarmlFile class has three methods: remove diagramByID, remove diagramByName and remove diagram. The two initial choices require one parameter to identify either the id or the name of the diagram.

ccistarmlFile			
Method	Description	Parameters	Returns
remove_diagramByName	Remove completely the diagram identified by the name passed to the method	String diagramName	Boolean. True if the diagram exists. False otherwise.
remove_diagramByID	Remove completely the diagram identified by the ID passed to the method	String diagramID	Boolean. True if the diagram exists. False otherwise.

Example

```
ccistarmlFile f = new ccistarmlFile();
ccistarmlContent diagram_1,diagram_2,diagram_3;
diagram_1 = f.add_diagram("My first i* diagram");
diagram_2 = f.add_diagram("My second i* diagram");
diagram_3 = f.add_diagram("My third i* diagram");
f.remove_diagramByName("My first i* diagram");
f.remove_diagramByID(diagram_3.id());
System.out.println(f.buildXML());
```

Console output

```
run:
<xml version="1.0"/>
<istarml>
    <diagram name="My second i* diagram">
    </diagram>
</istarml>
```

ccistarmlFile			
Method	Description	Parameters	Returns
remove_diagram	Remove completely the diagram object	ccistarmlContent theDiagram	Boolean. True if the diagram exists. False otherwise.

Example

```
ccistarmlFile f = new ccistarmlFile();
ccistarmlContent diagram_1,diagram_2,diagram_3;
diagram_1 = f.add_diagram("My first i* diagram");
diagram_2 = f.add_diagram("My second i* diagram");
diagram_3 = f.add_diagram("My third i* diagram");
f.remove_diagram(diagram_2);
System.out.println(f.buildXML());
```

Console output

```
run:
<xml version="1.0"/>
<istarml>
    <diagram name="My first i* diagram">
    </diagram>
    <diagram name="My third i* diagram">
    </diagram>
</istarml>
```

4 The ccfileError Class

The class ccfileError allows accessing the errors that the parsing process has found. Its access could be necessary if the programmers want to report the errors on a device different from the console, e.g. using a GUI using AWT or SWING. The class defines the public attributes which store the error detail.

ccfileError		
Public attributes	Description	Type
error	Contains the description of the error	String
line	Contains the error's line number	int
textLine	Contains the portion of the source line where the error was found	String
isXmlError	This attribute is true when the error was found by the method XmlParser. If the error comes from the method istarmlParser then the returning value is false	boolean

Example

```
LinkedList errs = my_iFile.errorList();
Iterator it = errs.iterator();
ccfileError myErr;
while (it.hasNext()) {
   myErr = (ccfileError)it.next();
   System.out.println(myErr.error+"** at line -->"+myErr.line);
}
```

Console output

```
id already defined** at line -->12
expected diagram tag** at line -->14
```

Observation

The method errorList is described in subsection 4.6

5 The class ccistarmlContent

5.1 General class's content

This class is a complex class which handles the kernel of both Xml Parsing and iStarmlParsing. Moreover it has some public methods which facilitate handling iStarML structures. Each object of this class represents a portion of the iStarML file. This portion can be a structured tag, a simple and self-contained tag and a text portion inside the file. If the content is a structured tag then there are nested ccistarmlContent objects.

There are some public attributes which allows knowing the type of content. The table 2 shows the relevant ones.

Table 2. Main attributes of the class ccistarmlContent

Attribute	Type	Represents	Example
tagName	String (java.lang)	If the object corresponds with a tag then this attribute contains the tag name. Other case the tagName attribute is empty (but not null)	actor
text	StringBuffer (java.lang)	If the object correspons with a text which is contained inside an external tag, then this attribute stores all the text.	Sample text
attribute	Hashtable (java.util)	Stores the attributes of a tag.	type=”goal”,
content	LinkedList (java.util) of ccistarmlContent objects	If the tag contains another tags inside it then all the contents (tags, text or both) are sorted in this list	
root	boolean	There is only one tag which set this attribute to true and corresponds to an abstract specification of the XML file. It has two contents, the xml tag (<?xml …) and the main tag of this file. In the case of iStarml it corresponds to the istarml tag.	

5.2 Structure of the xml’s content

Once a time the XML parsing is verified this structure is completed. Therefore, it structure can contains an XML content which does not complaint the iStarML format. The example 9 shows an XML portion which has its corresponding structure formed by ccistarmlContent objects in figure 1.

Example. A bad-formed iStarML structure but XML well-formed

```
1  <?xml version="1.0" encoding="UTF-8"?>
2  <actor type="agent" name="John">
3  Hello John
4  <actorLink type="is_a" aref="R34"/>
5    <boundary>
6      <ielement type="softgoal"/>
7    </boundary>
8  </actor>
```

In order to handle the structure, without altering the original one, it is recommended to copy the structure in your own ccistarmlContent structure by using the method xmlStructure of the class ccistarmlFile described in subsection 4.7.

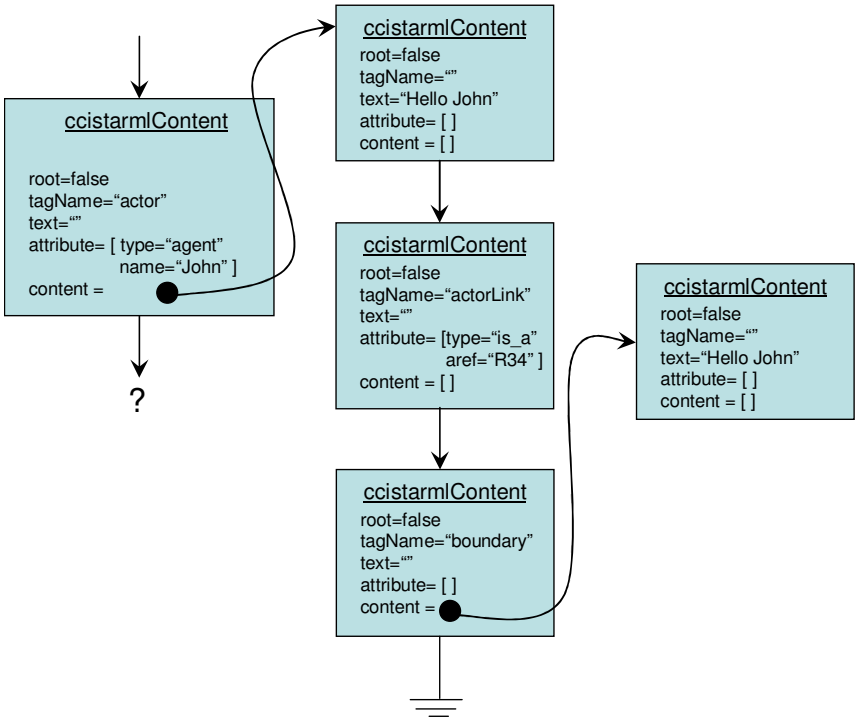

Figure 1. ccistarmlContent objects corresponding to the above XML code

5.3 Adding actors

The Class ccistarmlContent provides different methods for adding istaml elements to the existing diagrams. In this section we review the methods which allow adding actors. We can add generic actors by using the method add_actor or we can use specific methods add_role, add_position and add_agent to add specific types of actors.

ccistarmlContent			
Method	Description	Parameters	Returns
add_actor	Creates an actor inside the invoking element when it is possible. The actor is the returning data.	1: ccistarmlContent actor 1: String actorName	ccistarmlContent or null
Example			

ccistarmlFile f = new ccistarmlFile ();
ccistarmlContent d1,d2,a,b;
d1 = f.add_diagram("My new i* diagram");
a = d1.add_actor("My organizational unit");
d2 = f.add_diagram("Another diagram");
b = d2.add_actor(a);
System.out.println(f.buildXML());			

Console output			

```
run:
<xml version="1.0"/>
<istarml>
    <diagram name="My new i* diagram">
        <actor id="AA01" name="My organizational unit"/>
    </diagram>
    <diagram name="Another diagram">
        <actor aref="AA01"/>
    </diagram>
</istarml>
```

Observation

Note that if the same actor is added on a different diagram only its reference is added

ccistarmlContent

Method	Description	Parameters	Returns
add_role	Creates an actor of type role inside the invoking element when it is possible	1: String actorName	ccistarmlContent or null
add_agent	Creates an actor of type agent inside the invoking element when it is possible	1: String actorName	ccistarmlContent or null
add_position	Creates an actor of type position inside the invoking element when it is possible	1: String actorName	ccistarmlContent or null

Example

```
ccistarmlFile f = new ccistarmlFile ();
ccistarmlContent diag_1;
diag_1 = f.add_diagram("My new i* diagram");
diag_1.add_agent("Carlos");
diag_1.add_role("Finance Supervisor");
diag_1.add_position("General Manager");
System.out.println(f.buildXML());
```

Console output

```
run:
<xml version="1.0"/>
<istarml>
    <diagram name="My new i* diagram">
        <actor name="Carlos" type="agent"/>
        <actor name="Finance Supervisor" type="role"/>
        <actor name="General Manager" type="position"/>
    </diagram>
</istarml>
```

5.4 Adding actors' relationships

In order to create actors' relationships the class ccistarmlContent provides the method add_actorLink to add a generic actor's relationships. Also it is provided the methods add_is_a, add_is_part_of, add_instance_of, add_plays, add_occupies and add_covers in order to add the classical *i** actors' relationships.

ccistarmlContent			
Method	Description	Parameters	Returns
add_actorLink	Create an actor's relationship using the invoking element such the source of the relationship and the parameter like the target	1: ccistarmlContent actor 2: String relationshipType	ccistarmlContent or null

Example

```
ccistarmlFile f = new ccistarmlFile ();
ccistarmlContent diagram,org,cus;
diagram = f.add_diagram("My new i* diagram");
org = diagram.add_actor("My organization");
cus = diagram.add_actor("Customers' Association");
cus.add_actorLink(org,"supervises");
System.out.println(f.buildXML());
```

Console output

```
run:
<xml version="1.0"/>
<istarml>
    <diagram name="My new i* diagram">
        <actor id="AA01" name="My organization"/>
        <actor name="Customers' Association">
            <actorLink type="supervises" aref="AA01"/>
        </actor>
    </diagram>
</istarml>
```

ccistarmlContent			
Method	Description	Parameters	Returns
add_is_a	Add an actor's relationship of the type is_a from the invoking element to the parameter actor	1: ccistarmlContent actor	ccistarmlContent or null
add_is_part_of	Add an actor's relationship of the type is_part_of from the invoking element to the parameter actor	1: ccistarmlContent actor	ccistarmlContent or null
add_instance_of	Add an actor's relationship of the type instance_of from the invoking element to the parameter actor	1: ccistarmlContent actor	ccistarmlContent or null
add_plays	Add an actor's relationship of the type plays from the invoking agent to the parameter role	1: ccistarmlContent role	ccistarmlContent or null

add_occupies	Add an actor's relationship of the type occupies from the invoking agent to the parameter position	1: ccistarmlContent position	ccistarmlContent or null
add_covers	Add an actor's relationship of the type covers from the invoking position to the parameter role	1: ccistarmlContent role	ccistarmlContent or null

Example

```
ccistarmlFile f = new ccistarmlFile ();
ccistarmlContent diag_1,carlos,supervisor,manager;
diag_1 = f.add_diagram("My new i* diagram");
carlos = diag_1.add_agent("Carlos");
supervisor = diag_1.add_role("Finance Supervisor");
manager = diag_1.add_position("General Manager");
manager.add_covers(supervisor);
carlos.add_occupies(manager);
carlos.add_covers(supervisor);
System.out.println(f.buildXML());
```

Console output

```
run:
<xml version="1.0"/>
<istarml>
    <diagram name="My new i* diagram">
        <actor name="Carlos" type="agent">
            <actorLink type="occupies" aref="AA02"/>
        </actor>
        <actor id="AA01" name="Finance Supervisor" type="role"/>
        <actor id="AA02" name="General Manager" type="position">
            <actorLink type="covers" aref="AA01"/>
        </actor>
    </diagram>
</istarml>
```

Observation

Note that the last add_covers did not take effect because "carlos" is not a role

5.5 Adding intentional elements

To create intentional elements such as goals, softgoals, and resources, among others, the class ccistarmlContent provides the method add_ielement to add a generic intentional element. Also it is provided the methods add_goal, add_softgoal, add_resource, add_task and add_belief in order to add the more used types of intentional elements.

ccistarmlContent			
Method	Description	Parameters	Returns
Add_ielement	Add an intentional element using the invoking element such the container element. Also it is possible to add a previously created intentional element	1: String ielementName 2: String ielementType	ccistarmlContent or null
		1: ccistarmlContent ielement	

Example

```
ccistarmlFile f = new ccistarmlFile ();
ccistarmlContent dgr1,dgr2,supervisor,manager,goal;
dgr1 = f.add_diagram("Diagram 1");
dgr2 = f.add_diagram("Finance Department");
supervisor = dgr1.add_actor("Finance Department");
goal = dgr2.add_ielement("Design finance politic","pgoal");
dgr1.add_ielement(goal);
System.out.println(f.buildXML());
```

Console output

```
run:
<xml version="1.0"/>
<istarml>
    <diagram name="Diagram 1">
        <actor name="Finance Department"/>
        <ielement iref="AA01"/>
    </diagram>
    <diagram name="Finance Department">
        <ielement id="AA01" name="Design finance politic" type="pgoal"/>
    </diagram>
</istarml>
```

Observation

Note that if the ielement is added on a different diagram then only its reference is added

ccistarmlContent			
Method	Description	Parameters	Returns
Add_goal	Creates an ielement of type goal either inside an actor boundary or a diagram	1: String goalName	ccistarmlContent or null
Add_softgoal	Creates an ielement of type softgoal either inside an actor boundary or a diagram	1: String softgoalName	ccistarmlContent or null

Add_resource	Creates an ielement of type resource either inside an actor boundary or a diagram	1: String resourceName	ccistarmlContent or null
Add_task	Creates an ielement of type task either inside an actor boundary or a diagram	1: String taskName	ccistarmlContent or null
Add_belief	Creates an ielement of type belief either inside an actor boundary or a diagram	1: String beliefName	ccistarmlContent or null

Example

```
ccistarmlFile f = new ccistarmlFile ();
ccistarmlContent d,itsGoal,carlos;
d = f.add_diagram("Diagram 1");
carlos = d.add_agent("carlos");
itsGoal = d.add_goal("To provide software support");
carlos.add_resource("Viruses' behaviour database");
carlos.add_belief("A support job allows knowing organizations");
System.out.println(f.buildXML());
```

Console output

```
run:
<?xml version="1.0"?>
<istarml>
    <diagram name="Diagram 1">
        <actor name="carlos" type="agent">
            <boundary>
                <ielement name="Viruses' behaviour database"
                                                type="resource"/>
                <ielement name="A support job allows knowing
                                organizations" type="belief"/>
            </boundary>
        </actor>
        <ielement name="To provide software support"
                                            type="goal"/>
    </diagram>
</istarml>
```

5.6 Adding Intentional Relationships

To create intentional relationships such as contribution, mean-end, and decomposition among others, the class ccistarmlContent provides the method add_ielementLink to add a generic intentional relationship. Also it is provided the methods add_decomposition, add_means_end, add_why and add_contribution in order to add the more used types of intentional elements.

ccistarmlContent			
Method	Description	Parameters	Returns
Add_ielementLink	Add an intentional relationship using the current ccistarmlContent like the source of the relationship.	1: ccistarmlContent targetElement 2: String relationshipType 3: String value --- 1: ccistarmlContent targetElement 2: String relationshipType	ccistarmlContent or null

Example

```
ccistarmlFile f = new ccistarmlFile ();
ccistarmlContent d,goal_1,goal_2,hospital;
d = f.add_diagram("Diagram 1");
hospital = d.add_actor("Hospital");
goal_1 = hospital.add_softgoal("To provide high quality medical services");
goal_2 = hospital.add_goal("To implement a quality control system");
goal_2.add_ielementLink(goal_1,"contribution","some+");
System.out.println(f.buildXML());
```

Console output

```
run:
<?xml version="1.0"?>
<istarml>
    <diagram name="Diagram 1">
        <actor id="AA02" name="Hospital">
            <boundary>
              <ielement id="AA01" name="To provide high
               quality medical services" type="softgoal"/>
              <ielement name="To implement a quality
               control system" type="goal">
                    <ielementLink type="contribution"
                     iref="AA01" aref="AA02" value="some+"/>
              </ielement>
            </boundary>
        </actor>
    </diagram>
</istarml>
```

Observation

Note that identification attributes and id references are automatically added

ccistarmlContent			
Method	Description	Parameters	Returns
Add_means_end	Creates a *means_end* relationship from the current ccistarmlContent to another	1: ccistarmlContent targetElement	ccistarmlContent or null

Add_contribution	Creates a *contribution* relationship from the current ccistarmlContent to another	1: ccistarmlContent targetElement 2: String contributionValue	ccistarmlContent or null
Add_decomposition	Creates a *decomposition* relationship from the current ccistarmlContent to another	1: ccistarmlContent targetElement	ccistarmlContent or null
		1: ccistarmlContent targetElement 2: String decompositionValue	
Add_why	Creates a *why* relationship from the current ccistarmlContent to another	1: ccistarmlContent targetElement	ccistarmlContent or null

Example

```
ccistarmlFile f = new ccistarmlFile ();
ccistarmlContent d,g,t,st1,st2,r;
d = f.add_diagram("Diagram 1");
g = d.add_goal("To implement a quality control system");
r = d.add_resource("BSC software system");
t = d.add_task("To implement a metric based quality");
st1 = d.add_task("To elicitate quality metrics");
st2 = d.add_task("To configure BSC system");
st1.add_decomposition(t,"and");
st2.add_decomposition(t,"and");
r.add_means_end(st2);
t.add_means_end(g);
f.saveFile("iLink01.istarml");
```

The resulting file

```
1  <?xml version="1.0"?>
2  <istarml>
3    <diagram name="Diagram 1">
4      <ielement id="AA03" name="To implement a quality control
   system" type="goal"/>
5      <ielement name="BSC software system" type="resource">
6        <ielementLink type="means-end" iref="AA02"/>
7      </ielement>
8      <ielement id="AA01" name="To implement a metric based
   quality" type="task">
9        <ielementLink type="means-end" iref="AA03"/>
10     </ielement>
11     <ielement name="To elicitate quality metrics" type="task">
12       <ielementLink type="decomposition" iref="AA01" value="and"/>
13     </ielement>
14     <ielement id="AA02" name="To configure BSC system" type="
   task">
15       <ielementLink type="decomposition" iref="AA01" value="and"/>
16     </ielement>
17   </diagram>
18 </istarml>
```

5.7 Adding dependencies

The concept of dependency is a key concept on i* modelling. A dependency always has an intentional element which gives characterize the dependency. The ccistarml v0.6 package allows handling and creating dependencies, however the dependency is set throw the participating actor. That are the actor who depends (depender) from another one (dependee). The methods add_depender and add_dependee creates the dependency.

ccistarmlContent

Method	Description	Parameters	Returns
Add_depender	Creates a depender taking the current object like the intentional element which describes the dependency	1: ccistarmlContent dependerActor	ccistarmlContent or null
		1: ccistarmlContent dependerActor 2: String dependencyType	
Add_dependee	Creates a dependee taking the current object like the intentional element which describes the dependency	1: ccistarmlContent targetElement 2: String contributionValue	ccistarmlContent or null

Example

```
ccistarmlFile f = new ccistarmlFile ();
ccistarmlContent d,hospital,patient,g;
d = f.add_diagram("Diagram 1");
g = d.add_goal("To receive medical attention");
patient = d.add_actor("Patient");
hospital = d.add_actor("Hospital");
g.add_depender(patient,"open");
g.add_dependee(hospital);
f.saveFile("dependency01.istarml");
```

The resulting file

```
1  <?xml version="1.0"?>
2  <istarml>
3    <diagram name="Diagram 1">
4      <ielement name="To receive medical attention" type="goal">
5        <dependency>
6          <depender aref="AA01" value="open"/>
7          <dependee aref="AA02"/>
8        </dependency>
9      </ielement>
10     <actor id="AA01" name="Patient"/>
11     <actor id="AA02" name="Hospital"/>
12   </diagram>
13 </istarml>
```

5.8 Setting graphic properties

The iStarML v1.0 specification [4] considers two ways of specifying graphic properties: by using basic attributes as part of the iStarML specification and, by using embedded SVG tags [1]. The ccistarml Java package includes a set of methods for specifying the basic graphic options and, for the SVG choice, the method set_graphic_SVG allows specifying the XML content of a SVG specification. These methods are reviewed in the next tables.

ccistarmlContent			
Method	Description	Parameters	Returns
Set_graphic_bgcolor	Set the *bgcolor* graphic property (e.g. 44ff3E). If the tag graphic does not exist then this method create it	1: String hexRGBColor	boolean
Set_graphic_fontcolor	Set the *fontcolor* graphic property. (e.g. 44ff3E). If the tag graphic does not exist then this method create it	1: String hexRGBColor	boolean
Set_graphic_fontfamily	Set the *fontfamily* graphic property.(e.g. "Arial") If the tag graphic does not exist then this method create it	1: String fontName	boolean
Set_graphic_fontsize	Set the *fontsize* graphic property. If the tag graphic does not exist then this method create it	1: int fontSizeValue	boolean
Set_graphic_height	Set the *height* graphic property. If the tag graphic does not exist then this method create it	1: int heightValue	boolean
Set_graphic_shape	Set the *shape* graphic property ("ellipse", "rect", "spline", or "polyline"). If the tag graphic does not exist then this method create it	1: String shapeName	boolean
Set_graphic_unit	Set the *unit* graphic property.("cm", "in", or "pt"). If the tag graphic does not exist then this method create it	1: String unitSymbol	boolean
Set_graphic_width	Set the *width* graphic property. If the tag graphic does not exist then this method create it	1: int widthValue	boolean
Set_graphic_xpos	Set the *xpos* graphic property. If the tag graphic does not exist then this method create it	1: int xposValue	boolean
Set_graphic_ypos	Set the *ypos* graphic property. If the tag graphic does not exist then this method create it	1: int yposValue	boolean

Example

```
ccistarmlFile f = new ccistarmlFile ();
ccistarmlContent d,hospital,patient,g,dep;
d = f.add_diagram("Diagram 1");
d.set_graphic_width(350);
d.set_graphic_height(450);
d.set_graphic_unit("pt");
g = d.add_goal("To receive medical attention");
g.set_graphic_bgcolor("ff33ee");
g.set_graphic_xpos(20);
g.set_graphic_ypos(55);
g.set_graphic_width(80);
g.set_graphic_height(60);
f.saveFile("graphicSpec01.istarml");
```

The resulting file

```
1   <?xml version="1.0"?>
2   <istarml>
3     <diagram name="Diagram 1">
4       <graphic content="basic" width="350" height="450" unit="pt"/>
5       <ielement name="To receive medical attention" type="goal">
6         <graphic content="basic" xpos="20" ypos="55" width="80"
7                  height="60" bgcolor="ff33ee"/>
8       </ielement>
9     </diagram>
10  </istarml>
```

ccistarmlContent

Method	Description	Parameters	Returns
Set_graphic_SVG	Set the SVG XML content of a iStarML tag. If the passed XML is not well-formed then no graphic tag is added	1: String SVGContent	boolean

Example

```
ccistarmlFile f = new ccistarmlFile ();
ccistarmlContent d,hospital,patient,g,dep;
d = f.add_diagram("Diagram 1");
String mySVG = "\n<svg width=\"14cm\" height=\"4cm\"" +
            " viewBox=\"0 0 1200 500\">\n" +
            "<text x=\"20\" y=\"30\" font-family=\"Verdana\" "+
            " font-size=\"22\" fill=\"blue\" >" +
            " My i* diagram </text></svg>";
d.set_graphic_SVG(mySVG);
f.saveFile("graphicSpec02.istarml");
```

The resulting file

```
1   <?xml version="1.0"?>
2   <istarml>
3     <diagram name="Diagram 1">
4       <graphic content="SVG">
5         <svg width="14cm" height="4cm" viewBox="0 0 1200 500">
6           <text font-family="Verdana" font-size="22" y="30" fill="blue" x="20">
7             My i* diagram
8           </text>
9         </svg>
10      </graphic>
11    </diagram>
12  </istarml>
```

Observation

The SVG syntax is not checked

5.9 Setting additional attributes

One of the designing principles of iStarML was to get an extendible language without lost of the main concept of i*. It principle was implemented by giving the option of creating additional attributes on the existing tags. The method add_attribute implements this operation.

ccistarmlContent

Method	Description	Parameters	Returns
Set_attribute	Creates an additional attribute on an existing iStarML element.	1: String attributeName 2: String attributeValue	void

Example

```
ccistarmlFile f = new ccistarmlFile ();
ccistarmlContent d,hospital,patient,g,dep;
d = f.add_diagram("Diagram 1");
d.set_attribute("represents","future");
g = d.add_goal("To receive medical attention");
patient = d.add_actor("Patient");
patient.set_attribute("population","145000");
hospital = d.add_actor("Hospital");
dep = g.add_depender(patient,"open");
dep.set_attribute("value","commited");
g.add_dependee(hospital);
f.saveFile("dependency02.istarml");
```

The resulting file

```
1  <?xml version="1.0"?>
2  <istarml>
3    <diagram name="Diagram 1" represents="future">
4      <ielement name="To receive medical attention" type="goal">
5        <dependency>
6          <depender aref="AA01" value="commited"/>
7          <dependee aref="AA02"/>
8        </dependency>
9      </ielement>
10     <actor id="AA01" name="Patient" population="145000"/>
11     <actor id="AA02" name="Hospital"/>
12   </diagram>
13 </istarml>
```

Observation 1

Note that setting an exiting attribute means rewriting it.

Observation 2

Note that only some of the iStarML elements allow additional attributes. Therefore, if you do not control this option it is necessary running the parser at the end of the creation or update operation.

6 The class ERelementList

This class contains a list which allows handling the content of the iStarML file from an entity-relationship (ER) point of view. The constructor needs a ccistarmlContent object like the parameter which is the base for generating the ER structure. Moreover, this class has the method display which allows viewing the ER structure. Each node of the list represents either a tag element of the iStarML file or a relationship between two of these elements.

Each element is constituted by an identifier, a name and a set of attributes with their values. The class assigns a unique identifier to each individual element that does not have the ID attribute. Then to each element identifier is concatenated with the identifier of its corresponding diagram. In the case of diagram the concatenation is done with the identifier of the istarml tag. This concatenation constitutes the unique identifier of the iStarML elements. The name corresponds to the tag name or, in the case of relationships; it corresponds to the name "is_in". The set of attributes corresponds to the attributed of the tag in the iStarML file. In the case of the relationships the attributes allows identifying the elements (entities) involved in the relationship.

ERelementList

Method	Description	Parameters	Returns
ERelementList (constructor)	Creates an Entity Relationship representation for a iStarML nested structure	1: ccistarmlContent istarmlStructure	ERelement List
list	Get a list which contains the components of the iStarml structure in a Entity Relationships fashion	No parameters	LinkedList of ERelement objects

Example

```
ERelementList erL;
ccistarmlFile f = new ccistarmlFile ();
f.loadFile("ExampleER.istarml");
f.xmlParser();
erL = new ERelementList(f.xmlStructure());
erL.display();
System.out.println("Number of ER elements: "+erL.list().size());
```

Input file

```
1  <?xml version="1.0" encoding="UTF-8"?>
2  <istarml version="1.0">
3    <diagram name="Example ER">
4      <actor id="w124">
5        <boundary>
6          <ielement type="goal" name="Supervise students' career"/>
7          <ielement type="task" name="Send personal email"/>
8        </boundary>
9      </actor>
10   </diagram>
11 </istarml>
```

Console output
run: AA01-AA01:istarml-->{version=1.0} AA01-AA02:diagram-->{name=Example ER} AA02-wl24:actor-->{id=wl24} AA02-AA03:boundary-->{} AA02-AA04:ielement-->{type=goal} AA02-AA05:is_in-->{parent=AA03, child=AA04} AA02-AA06:ielement-->{name=Send personal email, type=task} AA02-AA07:is_in-->{parent=AA03, child=AA06} AA02-AA08:is_in-->{parent=wl24, child=AA03} AA02-AA09:is_in-->{parent=AA02, child=wl24} AA02-AA10:is_in-->{parent=AA01, child=AA02} Number of ER elements: 11
Observation
ERelement objects are described in the next section.

7 The class ERelement

This class allows containing the nodes of the ERelementList. An object of this class has some public attributes which identify the element and their own attributes.

ERelement		
Public attributes	Description	Type
diagram	Corresponds to the ID of the diagram which contains the current element	String
name	Corresponds to the xml tag name	String
text	If some tag has text between its open and end mark, then a separated element is created for this type of content. In this case the name has the value “string” and this attribute (text) has the content. Also it is used when the content corresponds to an SVG graphic content	String
ID	It contains the unique identifier of the iStarML element. If some element in the file does not have an ID the ER element creation algorithm assigns it.	String
attribute	This is a associative array which contains all the attributes (in the way of attribute-value pair) of the corresponding iStarML element.	Hashtable (java.util)

Example

```
ERelementList erL;
ERelement erle;
ccistarmlFile f = new ccistarmlFile ();
f.loadFile("ExampleER.istarml");
f.xmlParser();
erL = new ERelementList(f.xmlStructure());
Iterator it = erL.list().iterator();
System.out.println("Intentional Elements");
while (it.hasNext()){
   erle = (ERelement)it.next();
   if (erle.name.equals("ielement"))
      System.out.println("Code:"+erle.ID+" "+
                         erle.attribute +
                        " is on diagram "+erle.diagram);
}
```

Console output

```
run:
Intentional Elements
Code:AA04 {type=goal} is on diagram AA02
Code:AA06 {name=Send personal email, type=task} is on diagram AA02
```

8 Final Comments

The services provided by the ccistarml V0.6 package tackle the problem of importing, handling and exporting the XML content of an iStarML file.

We have used a one step process for parsing and creating the XML structure, therefore we have not used existing XML classes and we have implemented our own (but basic) XML recognizer. Besides some general features of XML file have not been included in this version, specifically we have not included the XML possibility of embedding different tags coming from different spacenames. Also we have not included the checking of the different possible set of chars.

We hope to reach a robust version 1.0 of the ccistarml Java package by including the error reports and some suggested extensions coming from the set of users who are trying to include iStarML as a medium representational language of i* diagrams. Therefore, any suggestion about how to improve this Java package will be very well received.

Acknowledgments

Special thanks to Gemma Grau who has suggested relevant functionality which has been added. Also I want to thanks to the University of La Frontera and its project FRO0105 which has supported my stay in the GESSI research group at Technical University of Catalonia, Barcelona, Spain.

References

1. *Scalable Vectors Graphics, SVG* http://www.w3.org/TR/SVG11/ Last accessed July 1, 2007 (2003)
2. Bresciani P., Perini A., Giorgini P., Giunchiglia F. and Mylopoulos J.:Tropos: An agent-oriented software development methodology. *Autonomous Agents And Multi-Agent Systems* **8**:(3) (2004) 203-236
3. Cares C., Franch X., Mayol E. and Quer C. A Reference Model for i*. In *Social Modelling for Requirements Engineering*, ed. E Yu, J Mylopoulos, N Maiden, P Giorgini, Boston: MIT Press, To be published (2007)
4. Cares C., Franch X., Perini A. and Susi A.: *iStarML Reference's Guide. Technical Report. LSI-07-46-R*, Technical University of Catalonia, Barcelona (2007)
5. Castro J., Kolp M. and Mylopoulos J.:Towards requirements-driven information systems engineering: the Tropos project. *Information Systems* **27** (2002) 365-389
6. Clark J. and DeRose S.: *XML Path Language (XPath) Version 1.0.World Wide Web Consortium (W3C)* http://www.w3.org/TR/xpath Last accessed April 2007 (1999)
7. Grau G.: *A Comparative of i* Modelling Tools* http://istar.rwth-aachen.de/tiki-index.php?page=i%2A+Tools Last accessed (2006)
8. Maiden N.:What has requirements research ever done for us? *IEEE Software* **22**:(4) (2005) 104-105
9. Yu E.: *Modelling Strategic Relationships for Process Reengineering*. PhD.Computer Science University of Toronto, Toronto. (1995)