DREAM

REGRESSION TESTING

Principles & User Guide v1.0 03.09.2014

http://www.testtooldream.com

support@drost.name

author: Marcus Drost

proofreader: Philip van Wijk

Table of Contents

Introduction

Software development is becoming more and more agile. In order to adapt agility many organizations switch from waterfall to agile frameworks like scrum or kanban. I myself have participated in a process of migration from waterfall to scrum. The team, I worked for, was a data intensive BI/DWH environment of a bank. There, I found that the process of testing is quite challenging in an agile organized team. In waterfall processes there is often much time reserved in order to test much functionality in a reserved period of time. In contrast, agile processes require many small pieces of functionality to be tested frequently. And this makes a big difference as test processes are not very flexible and need to be prepared intensively. Speaking from experience, testing data intensive applications in agile environments just as done in the previous waterfall situation, is very risky as many gaps in the test coverage will remain. Therefore, more sophisticated methods and tools are necessary in order to guarantee that customers' expectations regarding quality will be met.

There are many ways to automate test processes of agile development teams. One popular approach which is called TDD, is described in detail on the site of Scott W. Ambler: Test Driven Development. However, in this Principles & User Guide I would like to focus on a more special subject of data testing by acquainting you with an approach that focuses on finding and analyzing differences in data quickly on an ad hoc basis and without programming. Even though testing is quite a creative activity as every situation and test object is different, there are controlling tasks that occur often and these should be automated. I know from experience, that the task of comparing data in databases, excel sheets and XML documents is necessary in order to control regression sets and predicted output sets. Though, controlling them by random examples and check sums can lead to acceptable quality, some disadvantages do remain. First of all, both methods leave gaps of uncontrolled data where risky flaws still could exist. Secondly, those methods do not tell what causes the differences in data sets.

The DREAM approach is the answer to disadvantages of other test compare methods (see DREAM's competitors). The DREAM approach makes it possible to find 100% of differences in data sets and to analyze the root causes of these differences. DREAM is able to aggregate differences to a higher level but it is also able to drill down to see details. DREAM is based on XML and is thereby compatible to all types of databases, like RMBMS, NOSQL, Excel etc. DREAM is an excellent means for testers to reduce time and costs of comparing data and testing, especially regression testing.

DREAM's principles

Basic principles of comparing

In order to understand DREAM, one should understand the principles of comparing and how DREAM copes with these principles. Unlike many other applications DREAM does not compare two file line by line, but it recognizes the structure of the data (see Illustration 2: principles of data structures). As DREAM uses XML as source file format, DREAM has the possibility and flexibility to compare different formatted data as long as those data can be represented in XML structures . Thereby, DREAM can compare almost all practicable formats like Excel files, CSV files, JSON, relational tables, NoSQL data, Star Models, Data Warehouses and Data Vaults.

In every day life, comparing things occur often. Accountants, for example, are continuously comparing balance sheets. This is made easy by using check sums. But a clerk, comparing two different versions of multiple column Excel sheets with product data will need quite more time if rows are double or information has been changed somewhere arbitrarily in the Excel sheet's cells. Here, check sums might indicate differences, but finding and analyzing root causes of differences is still quite time consuming. Probably, testers and analysts in software development processes spend most of their working time on comparing the impact of changes by comparing actual results with expected results. Next to verifying the correctness of changes it is important to control the regression, i.e. assuring that what is out of scope of the change remains unchanged.

Illustration 1: basic principles of comparing describes how DREAM handles differences and equality. In this example, the form of an entity identifies it (see Identification of entities). Equal entities like the green rectangles are neglected as DREAM focuses on differences. The triangle, by contrast, differs, as it is on the left side blue and on the right side orange. The brown diamond only exists on the left side and the yellow circle only exists on the right side.

Conclusion: There are 4 situations of basic comparing:

- equality (which will be neglected)
- differences shown in both entities
- existances only on the left side
- existances only on the right side

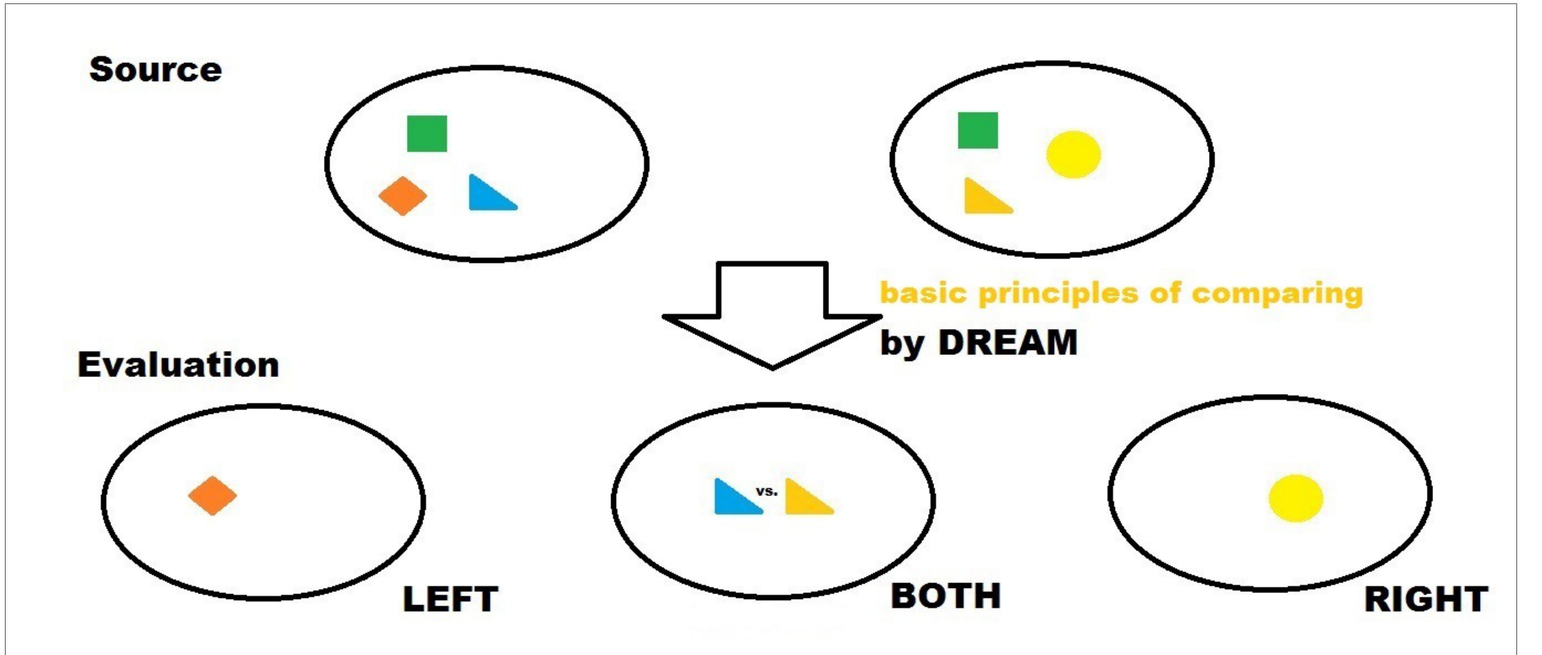
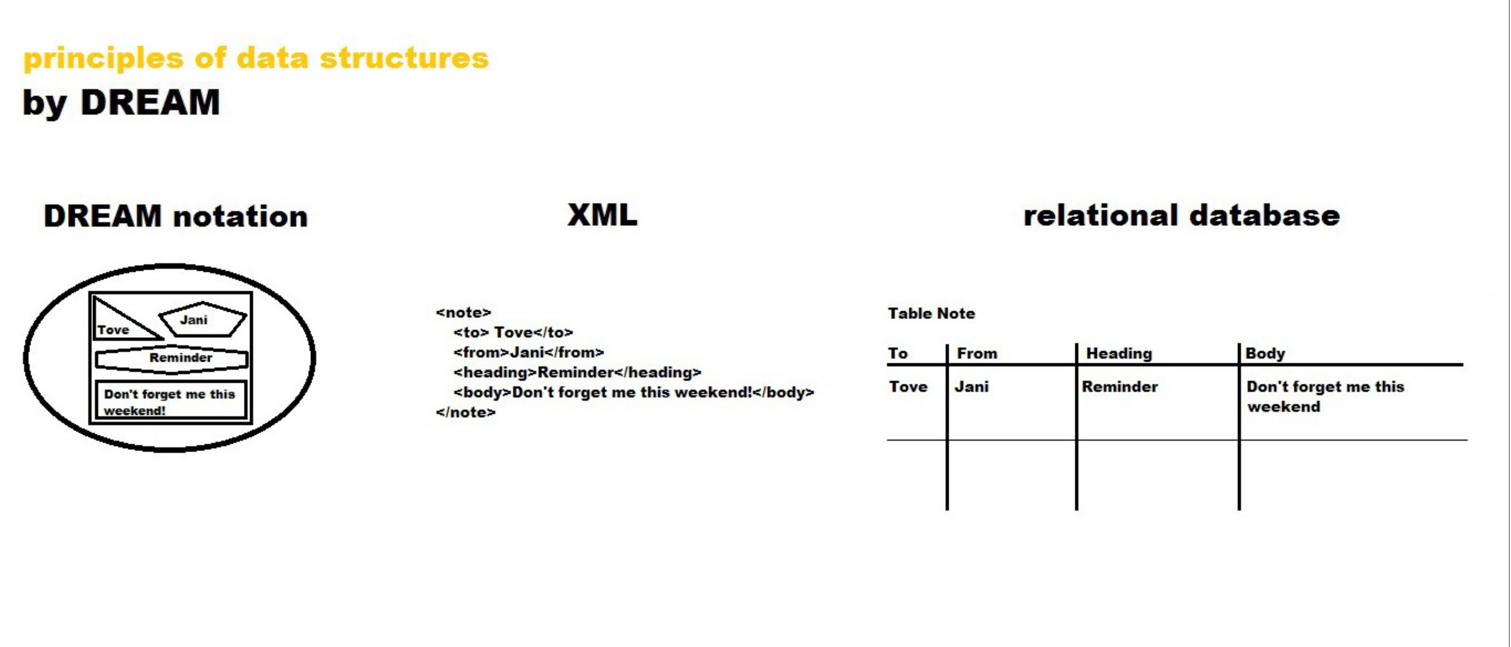


Illustration 1: basic principles of comparing

Principles of data structures

Data can be represented in different data structures like XML, relational database, NoSQL, CSV etc. Every data structure has advantages and disadvantages. Some data structures are very flexible, others are very suitable for high performance or for securing integrity of data. DREAM is based on XML as it is a universal web-standard of W3C. Almost all data from different data structures can be easily transformed to XML.

In Illustration 2: principles of data structures the same data is represented in three different data structures. In a world of multiple data structure standards, the fact that XML is an open standard was the reason to use XML as the basis for DREAM. For reasons of abstraction, many examples in this document are represented in the graphical DREAM notation as shown on the left side.



```
<note>
   <to> Tove</to>
   <from>Jani</from>
   <heading>Reminder</heading>
   <body>Don't forget me this weekend!</body>
</note>
```

Table Note

To	From	Heading	Body
Tove	Jani	Reminder	Don't forget me this weekend

Illustration 2: principles of data structures

Comparing tree structures

Comparing one dimensional entities or attributes like digits or names is still quite simple. But most data that we currently do use, has a more complex underlying structure. Those structures can be static like those of relational databases or CSV files, or they can be flexible like XML or JSON. In general , comparing data with a static structure is more understandable, than e.g. XML files that have variable depth tree structures. Next to the flexibility of structures, the concept of identification (see Identification of entities) is a decisive factor for comparison. For example, relational databases with unique keys are easier to compare than arbitrary filled XML files.

In Illustration 3: comparing tree strutures you can see how DREAM handles complex structures. DREAM can compare tree structures that are represented in XML. Thereby, it can handle all practicable data structures as almost all data can be represented in XML (see Converting Relational Database Into Xml Document, Bibliography). As stated above, their form identifies entities. For more complex data structures like trees the combination of successive forms and their depth are used for identifying those colored entities. These entities are leaves of the tree structures, and only leaves are compared by DREAM as in good XML design other nodes do not contain relevant data except for identification (see XML Tutorial W3C, XML Syntax Rules, Bibliography).

Every colored entity in the illustration below can be identified e.g. the green circle on left in the source by (rectangle-depth1->rectangle-depth2->circle-depth3) and the triangle on the left and right side of the source by (rectangle-depth1->rectangle-depth2->triangle-depth3). After comparing those colored entities, in this example differences are again represented in the context of their structure that identifies them uniquely.

Conclusion: In complex structures entities are compared within their position in the structure i.e. tree structure (see Tree data structure, Bibliography).

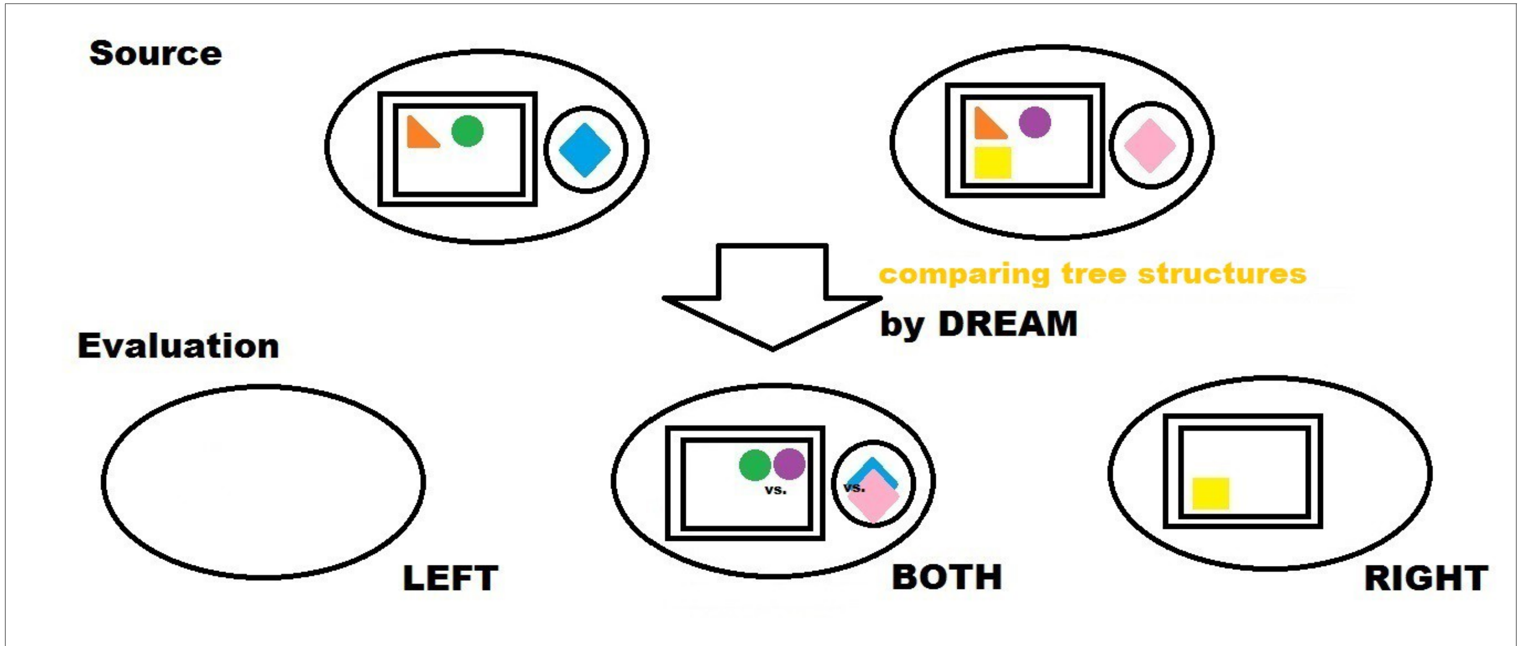


Illustration 3: comparing tree strutures

Identification of entities

Above the term of identification has already been introduced. In this section the impact of identification is shown in Illustration 4: principles of identification. In the example one can see that in the left and right source set, circles and triangles can be uniquely identified if the form of the entity is used for identification. But in case of rectangles entities cannot be identified uniquely. Comparing rectangles of both sides and using form as identification would lead to $2^2=4$ compares i.e. and an incomprehensible result. Instead, here in the case of rectangles a set has been identified which deserves a special way of comparing. Comparing sets is explained in the next section (see Comparing sets).

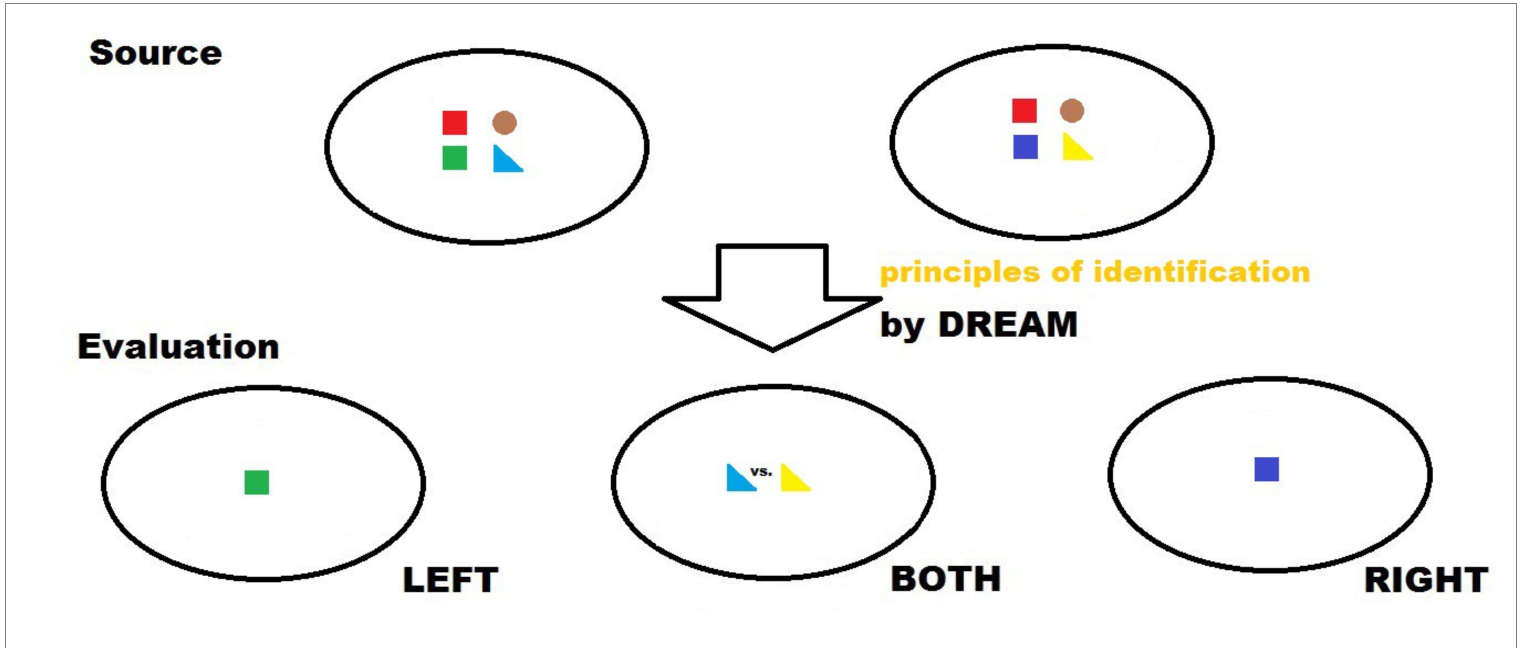


Illustration 4: principles of identification

Comparing sets

As long as every entity can be defined uniquely, comparing can be done in a relatively straightforward way. This is often the case in classical relational databases where entities are identified by unique keys. But even in relational databases there might be tables that contain double records. In Illustration 5: comparing sets this situation is shown. Using the method of identification by form as agreed above, we find on the left and right source side 5 entities with the same identification (rectangle-depth1->rectangle-depth2). In practice, this could happen e.g. if a system stores telephone numbers of customers with the option to store more than one mobile number per customer. Those entities are recognized by DREAM as sets. In case of sets DREAM uses the color i.e. the data that is stored in leaves in order, to divide entities into three types of sets. This conforms for the most part to the set theory (see Set theory, Bibliography), except that DREAM treats duplicates in an informal manner as in elementary school texts (see Can elements in a set be duplicated?, Bibliography):

- entities that appear on the left and right side i.e. the intersection (these entities are neglected by DREAM)
- remaining entities that appear only on the left side
- remaining entities that appear only on the right side

Due to the lack of means of identification, sets never evaluate to results in the set for both. The illustration shows that DREAM neglects the beige rectangles that appear on both sides as well as 2 brown rectangles, and that DREAM assigns arbitrarily 2 brown rectangles to the evaluation set on the left and 2 green rectangles to the evaluation set on the right side.

Conclusion: A set consists of 2 or more entities that can no longer be identified uniquely. For sets DREAM is largely based on the set theory (see Set theory Bibliography) but allows duplicates (see Can elements in a set be duplicated?, Bibliography).

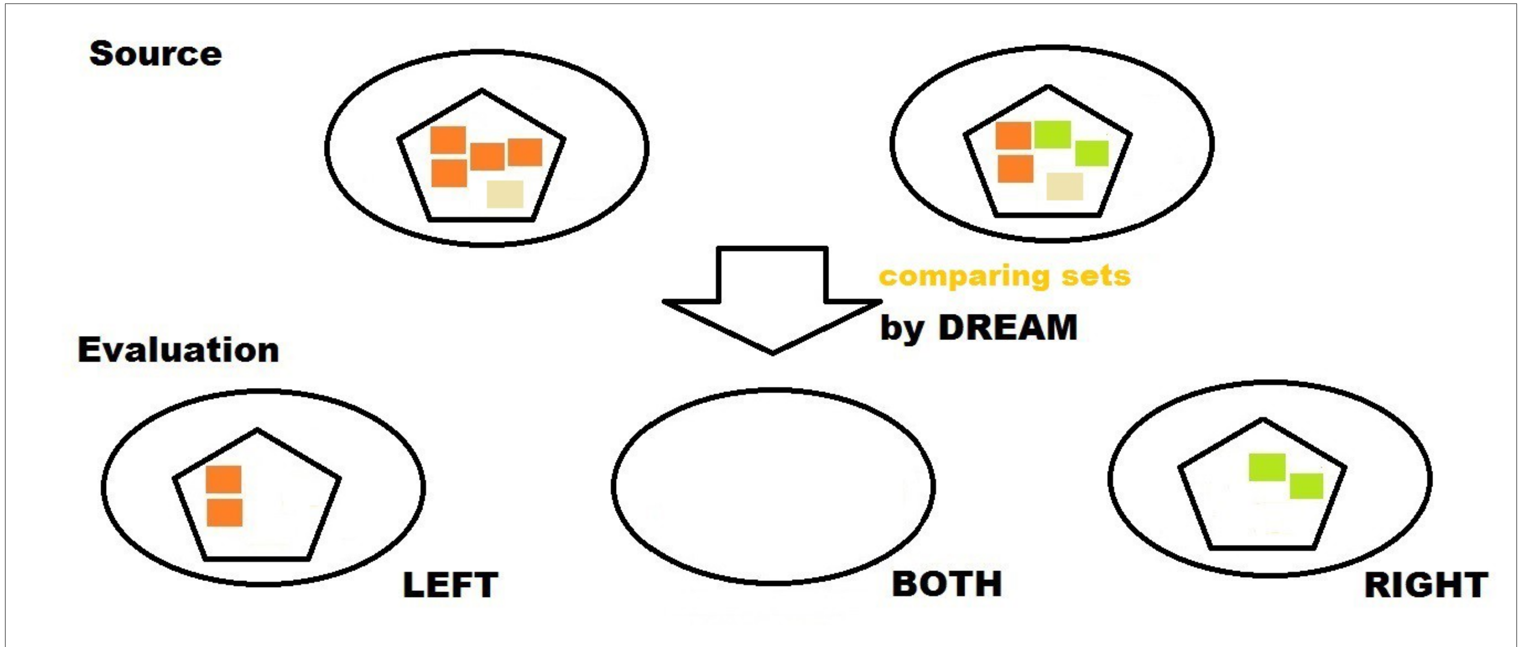


Illustration 5: comparing sets

Context analysis

The process of finding differences between different data sets often takes much time and effort. Working under pressure to meet that next deadline, it is very difficult to stay focused on pursuing a high level of quality. Furthermore, finding differences is not enough since problems can only be solved when a root cause is known.

In practice, differences between data sets are often the result of special situations that can only be found when a pattern is found somewhere in the data sets. If two versions of a payment database differ, those differences might be related to the payment type, special customers, their postal code, the date or time or some other data that is related to payments. As long as differences are the result of causes that are represented as special patterns in the data, DREAM can analyze them in order to find the root cause of differences.

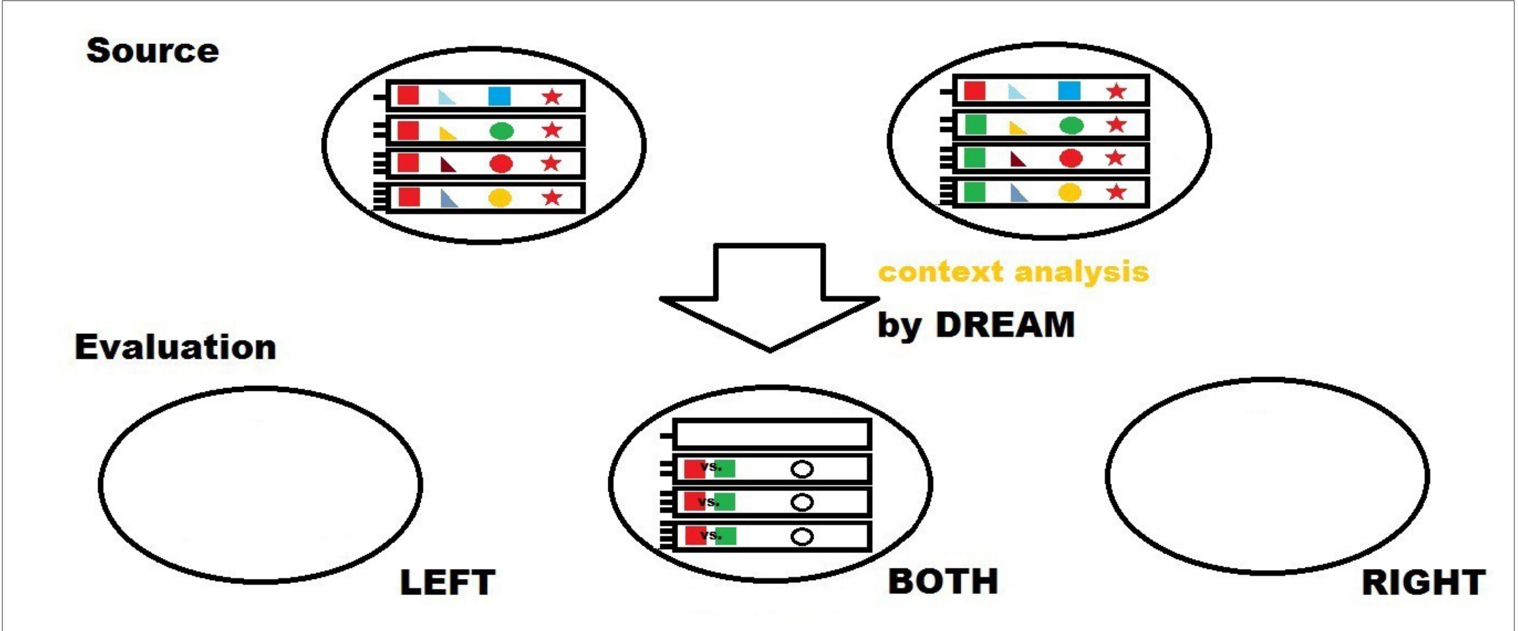


Illustration 6: context analysis

This is shown in Illustration 6: context analysis where the differences seem to occur in the case appearances of circles. Once automatically detected by DREAM, testers and analysts can verify the proposition of DREAM by checking data and source code. With DREAM's context analysis, finding root causes of differences will be much faster than in the past because additional programming is no longer necessary.

Context creation

After finding differences follows the step of finding root causes within the data set's context. The cause of differences might be located close by found differences or somewhere in far away related data (see Illustration 7: context creation). For example, let us assume that the amount of all foreign country payments differ and the amount as well as country indication exist in the same record or fact. In this situation it is sufficient to deliver the payment table in XML format to DREAM for analysis. If the root cause is supposed to be in an entity that can be joined to within several steps, the whole context i.e. the completely joined entities from difference to cause should be input in XML format to DREAM.

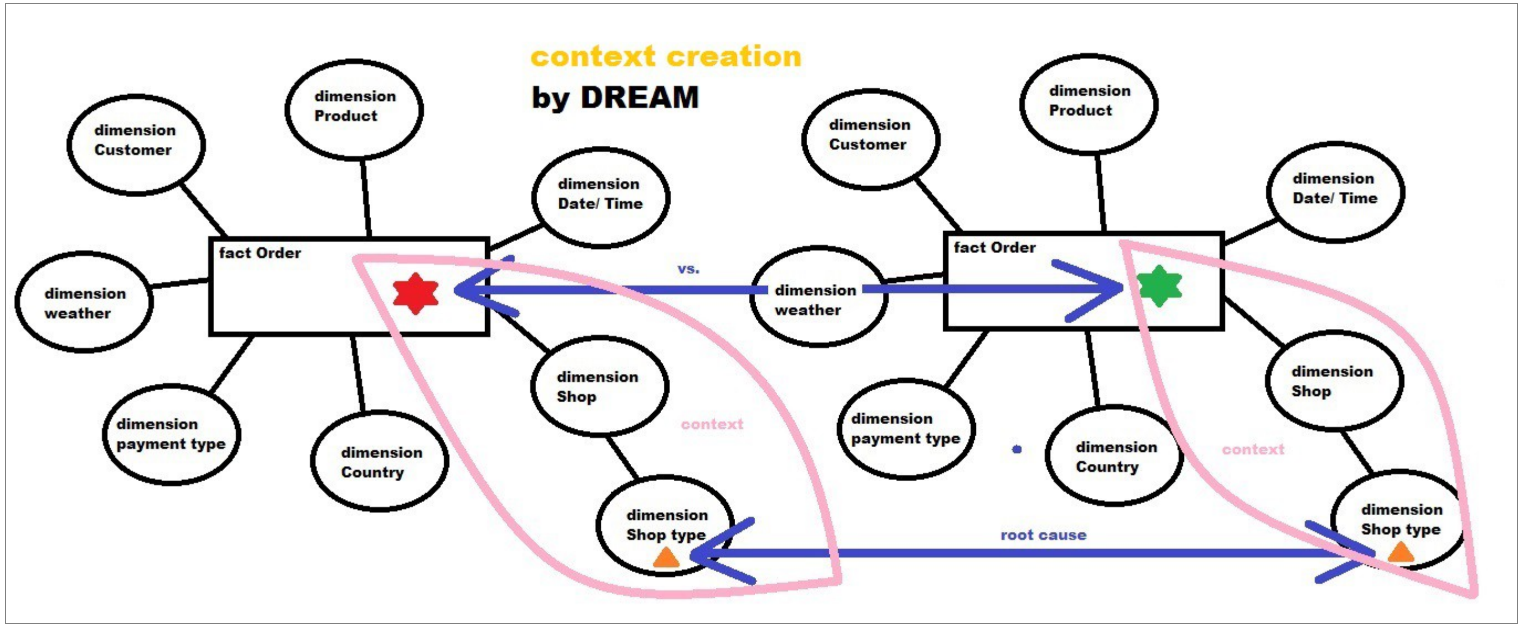


Illustration 7: context creation

Aggregation of results

Sometimes, the number of differences is such a high amount that it is too large to handle. Assume, two payment tables with 100.000 records have 10.000 differences in one column and 10.000 differences in another column. Looking at all detailed differences would be confusing. Therefore, DREAM allows the aggregation of those differences by comparing them with totals. Undoubtedly, aggregations help to estimate the impact of differences. With DREAM it is possible to aggregate to all levels within the tree structure. Thereby, analysts and testers can incrementally drill down from the top to leaves in order to refine the result of differences.

How aggregations works has been shown by the example in Illustration 8: aggregation of results. Instead of presenting every detailed difference, they are summed up as totals. The same is applicable for the totals of context analysis.

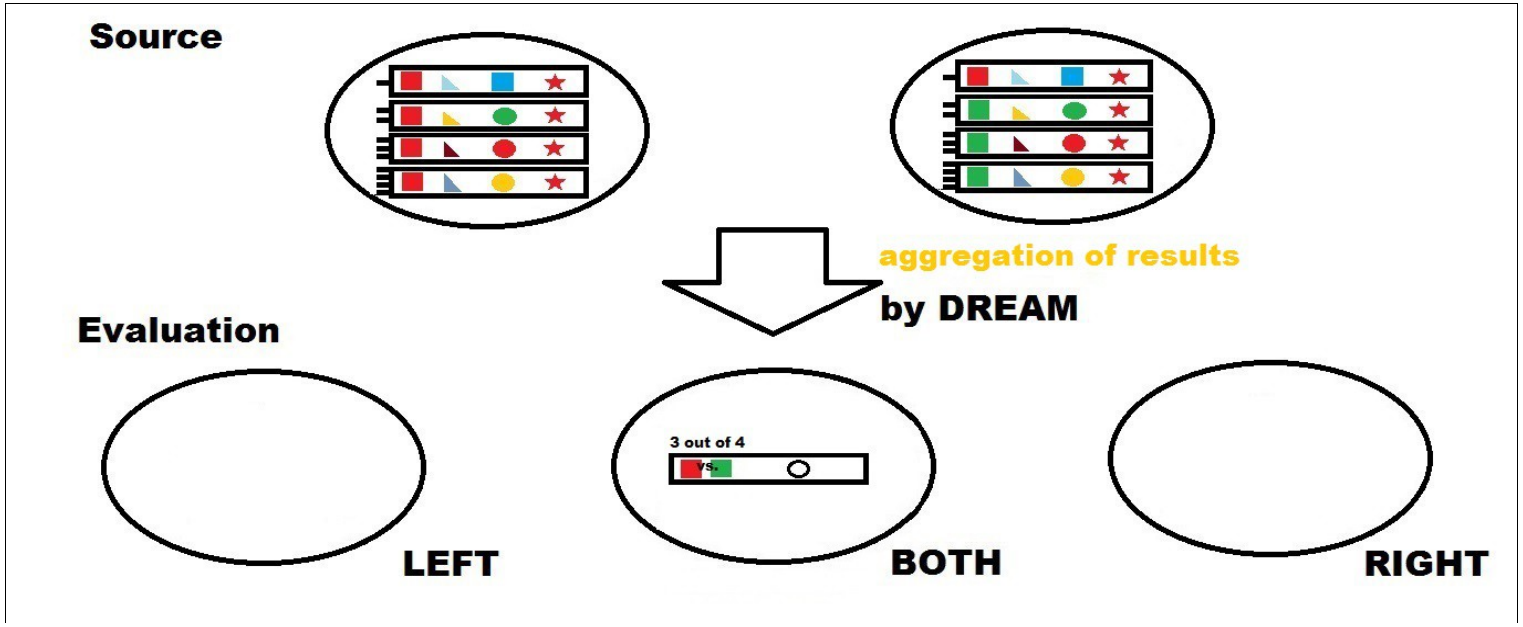


Illustration 8: aggregation of results

DREAM tool

Those who have read DREAM's principles should agree that comparing i.e. finding differences and investigating root causes is not a trivial matter. All that has been described in the principles chapter is needed in practice in order to succeed in situations that involve much data analysis, testing and especially regression testing activities. Fortunately, the tool DREAM exists and helps to automatize those complex activities. DREAM is available for everybody's PC and processes data in the universal XML format for all database types.

Licence

DREAM is proprietary software licenced by MD IT Consultancy. MD IT Consultancy has the exclusive legal right of being copyright holder. Licensee is given the right to use the software only under certain conditions as agreed with MD IT Consultancy.

Other uses are not allowed to do the following:

1. make modifications in the software
2. redistributing the software
3. reverse engineering

THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.

System requirements

Currently DREAM is available as Beta Release 1.

In order to run DREAM you will need:

- A 64-bit Windows operating system: Windows 7 or Windows 8
- A 64-bit JVM (Java Virtual Machine)

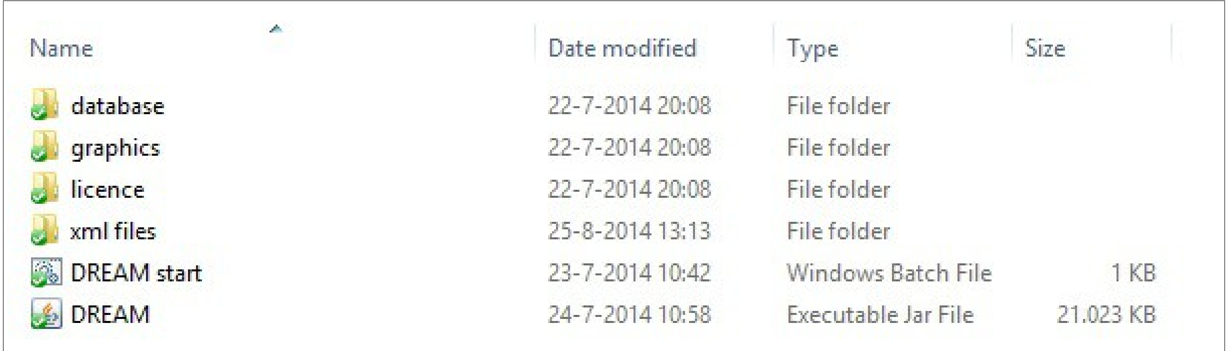
You can check the version of your JVM in the following way:

```
Start the command line in Windows. Then type: java -version. This reveals information about JVM bitness.

In case of 64 bit JVM it prints :
C:\>java -version
java version "1.6.0_25"
Java(TM) SE Runtime Environment (build 1.6.0_25-b06)
Java HotSpot(TM) 64-Bit Server VM (build 20.0-b11, mixed mode)

while in case of 32 bit JVM it will print
C:\> java -version
java version "1.6.0_26"
Java(TM) SE Runtime Environment (build 1.6.0_26-b03)
Java HotSpot(TM) Client VM (build 20.1-b02, mixed mode, sharing)
```

Downloading and running DREAM

You will find the most up to date download link on: www.testtooldream.com. The download directory has a size of approximately 30 MB.

After downloading you will find a structure as shown in Illustration 9: DREAM standard directory structure. Most Windows Systems will recognize DREAM as java application, so you can double click the file DREAM. If not double click the file 'DREAM start' in order to start DREAM.



Name	Date modified	Type	Size
database	22-7-2014 20:08	File folder	
graphics	22-7-2014 20:08	File folder	
licence	22-7-2014 20:08	File folder	
xml files	25-8-2014 13:13	File folder	
DREAM start	23-7-2014 10:42	Windows Batch File	1 KB
DREAM	24-7-2014 10:58	Executable Jar File	21.023 KB

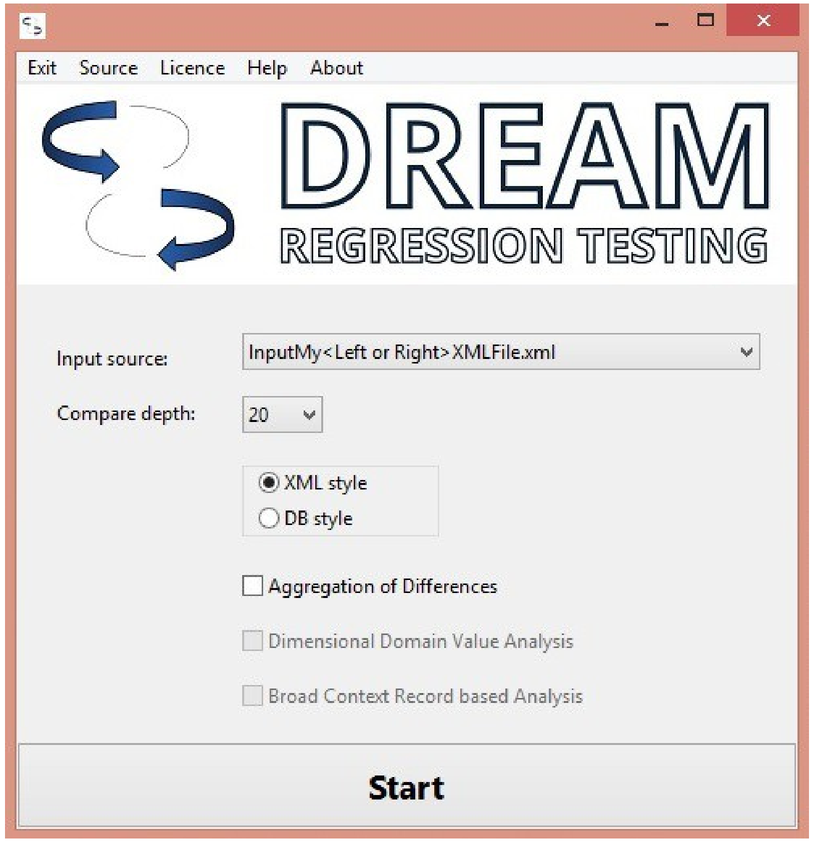
Illustration 9: DREAM standard directory structure

DREAM is proprietary software or closed source software released by MD IT Consultancy. If your licence has expired please contact: support@drost.name

After receiving a new licence number from MD IT Consultancy, you can use the licence dialogue in order to enter the licence key into DREAM (see Licence dialogue).

Center dialogue

The center dialogue (see Illustration 10: center dialogue) is what users see first when starting the application. From here you can navigate to all other dialogues. On the top of this dialogue there are some menu choices to other dialogues like the source dialogue, the licence dialogue and the about dialogue. As this is a beta version, the help dialogue is not ready yet. On the panel of the dialogue there are some options that let you decide how to compare files.



Input source: InputMy<Left or Right>XMLFile.xml

Compare depth: 20

[✓ XML style] DB style

☐ Aggregation of Differences

☐ Dimensional Domain Value Analysis

☐ Broad Context Record based Analysis

Illustration 10: center dialogue

On the panel of the dialogue there are some options that let you decide how to compare files.

Required choices in the Center dialogue:

- First of all, choose the pair of files to compare: the input sources.
- Secondly, by choosing the compare depth you decide how detailed your results will be. By choosing “1” you will compare just the roots of the trees. This can be increased node by node to the most far away leaves of the tree. When comparing table data from relational tables you will find the leaves already on level 3. In XML web-development structures trees can be much deeper.
- Thirdly, choose to compare the pair of files just as XML files, or as DB files. By comparing the files as DB files, DREAM assumes that the files have a 3 level symmetric structure as default generated by the ODBC dialogue (see Illustration 19: ODBC dialogue).

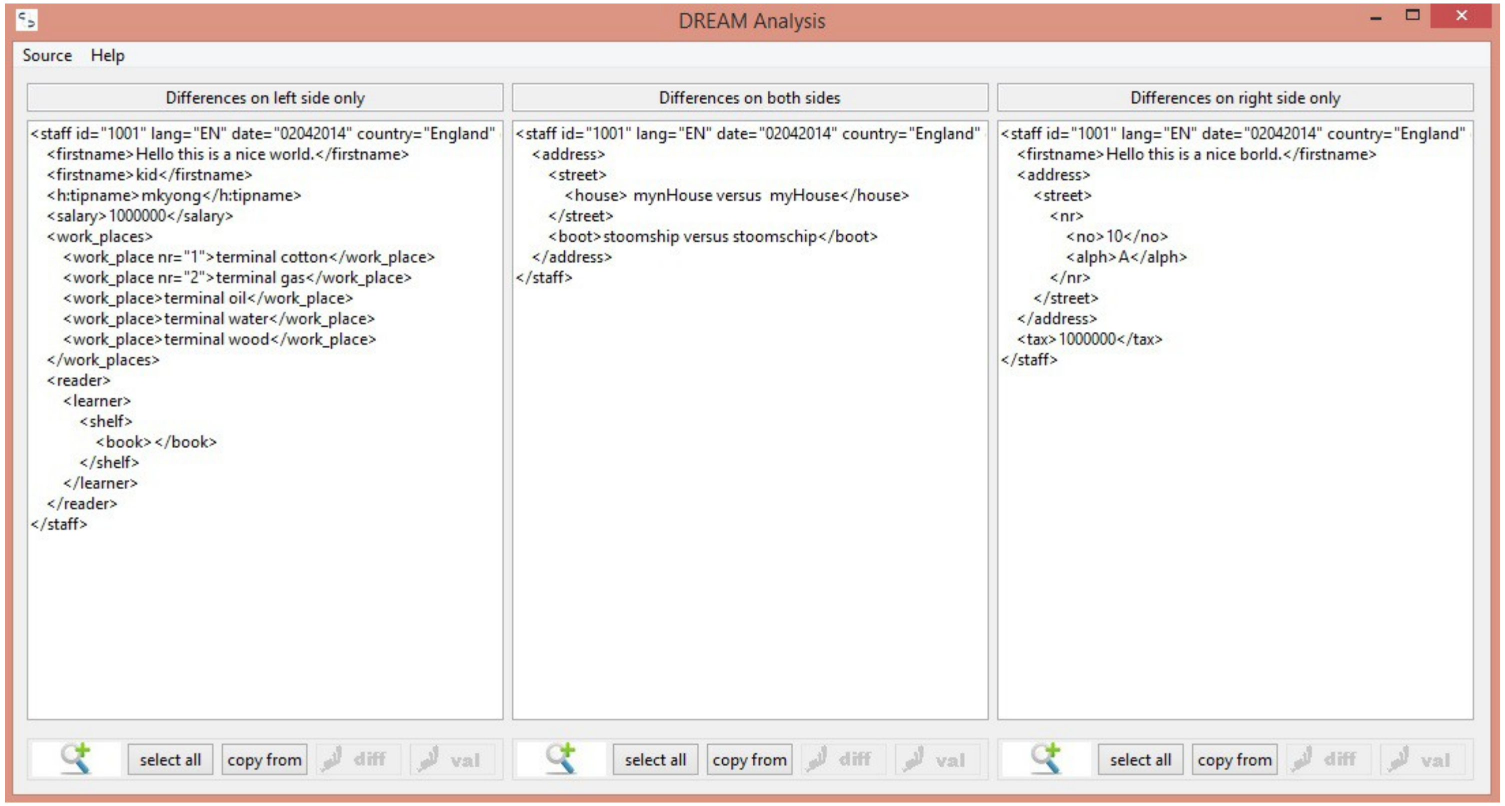
Optional choices in the Center dialogue:

- Fourthly, the differences can be aggregated to higher levels and compared to totals (see Illustration 8: aggregation of results).
- Fifthly, it is possible to do an analysis of domain values of differences. Assume, for example, a situation that in a payment database the currency column has differences and that in all differences the currency Dollar is involved. This may lead to special conclusions regarding the root cause.
- Lastly, the domain context analysis allows you to analyze the environment of differences. For example, that the currency differences of the payment database are due to special credit card types somewhere in the context i.e. the data about credit cards is available in the pair of input files that has been chosen (see Illustration 7: context creation)

After choosing all the options the process of finding differences and analyzing can be started by pressing the large start button at the bottom.

Analysis dialogue

The analysis dialogue splits differences in left, both and right (see Illustration 1: basic principles of comparing). By clicking the reading-glass button the panel with data can be enlarged (see Illustration 11: detailed analysis). There are two buttons for selection and copying data from the panels. The diff button is used for analyzing pairs of files by using DB style option. The val button is used for domain analysis and if chosen domain analysis of the context.



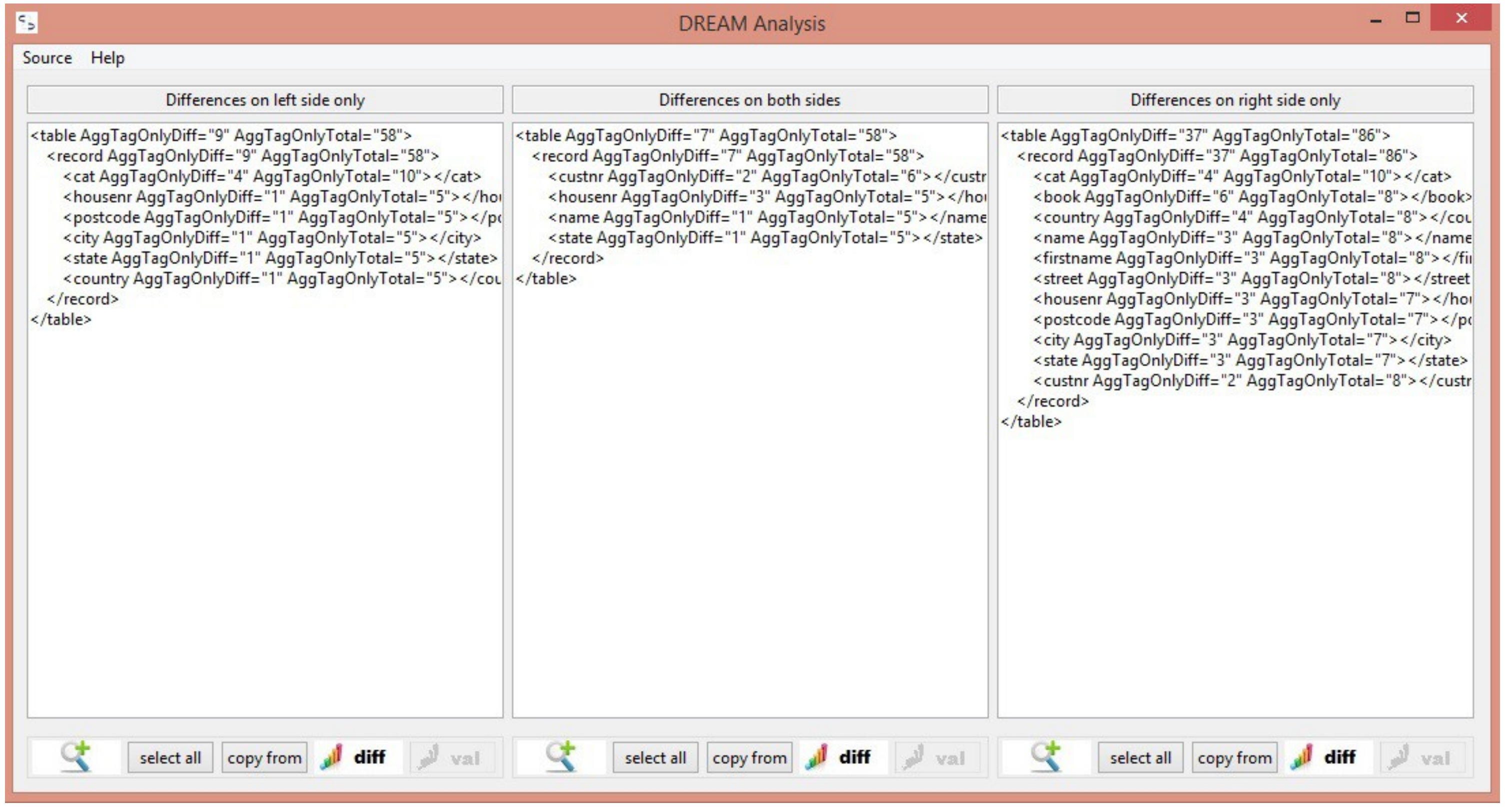
```
<staff id="1001" lang="EN" date="02042014" country="England"
  <firstname>Hello this is a nice world.</firstname>
  <firstname>kid</firstname>
  <h:tipname>mkyong</h:tipname>
  <salary>1000000</salary>
  <work_places>
    <work_place nr="1">terminal cotton</work_place>
    <work_place nr="2">terminal gas</work_place>
    <work_place>terminal oil</work_place>
    <work_place>terminal water</work_place>
    <work_place>terminal wood</work_place>
  </work_places>
  <reader>
    <learner>
      <shelf>
        <book></book>
      </shelf>
    </learner>
  </reader>
</staff>
```

```
<staff id="1001" lang="EN" date="02042014" country="England"
  <address>
    <street>
      <house> mynHouse versus  myHouse</house>
    </street>
    <boot>stoomship versus stoomschip</boot>
  </address>
</staff>
```

```
<staff id="1001" lang="EN" date="02042014" country="England"
  <firstname>Hello this is a nice borld.</firstname>
  <address>
    <street>
      <nr>
        <no>10</no>
        <alph>A</alph>
      </nr>
    </street>
  </address>
  <tax>1000000</tax>
</staff>
```

Illustration 11: detailed analysis

In Illustration 12: aggregated analysis an analysis is shown for which aggregation has been used in order to come to know the totals of differences. The aggregation option is available on the center dialogue (see Illustration 10: center dialogue).



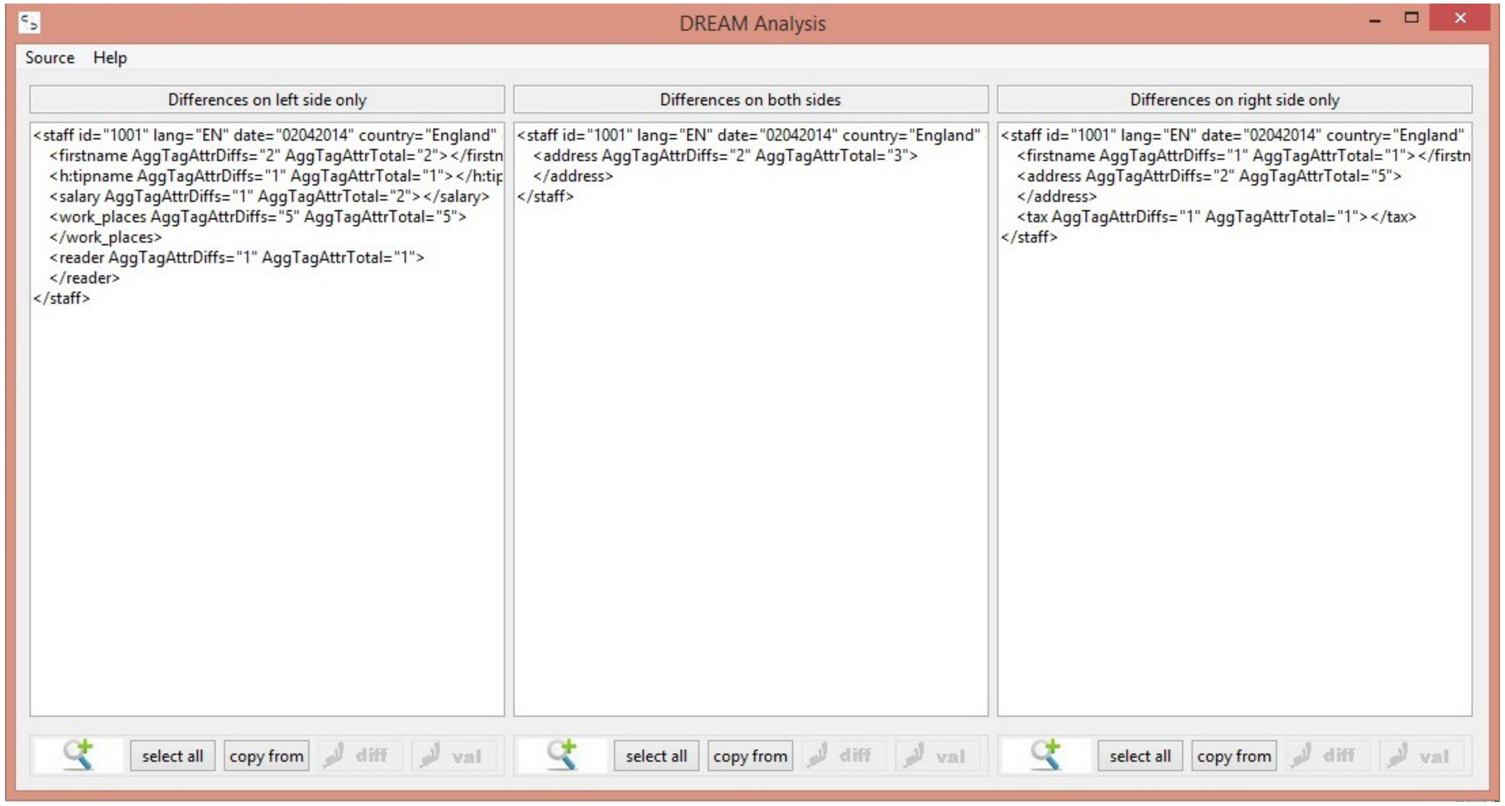
```
<table AggTagOnlyDiff="9" AggTagOnlyTotal="58">
  <record AggTagOnlyDiff="9" AggTagOnlyTotal="58">
    <cat AggTagOnlyDiff="4" AggTagOnlyTotal="10"></cat>
    <housenr AggTagOnlyDiff="1" AggTagOnlyTotal="5"></hou
    <postcode AggTagOnlyDiff="1" AggTagOnlyTotal="5"></pc
    <city AggTagOnlyDiff="1" AggTagOnlyTotal="5"></city>
    <state AggTagOnlyDiff="1" AggTagOnlyTotal="5"></state>
    <country AggTagOnlyDiff="1" AggTagOnlyTotal="5"></cou
  </record>
</table>
```

```
<table AggTagOnlyDiff="7" AggTagOnlyTotal="58">
  <record AggTagOnlyDiff="7" AggTagOnlyTotal="58">
    <custnr AggTagOnlyDiff="2" AggTagOnlyTotal="6"></custr
    <housenr AggTagOnlyDiff="3" AggTagOnlyTotal="5"></hou
    <name AggTagOnlyDiff="1" AggTagOnlyTotal="5"></name
    <state AggTagOnlyDiff="1" AggTagOnlyTotal="5"></state>
  </record>
</table>
```

```
<table AggTagOnlyDiff="37" AggTagOnlyTotal="86">
  <record AggTagOnlyDiff="37" AggTagOnlyTotal="86">
    <cat AggTagOnlyDiff="4" AggTagOnlyTotal="10"></cat>
    <book AggTagOnlyDiff="6" AggTagOnlyTotal="8"></book>
    <country AggTagOnlyDiff="4" AggTagOnlyTotal="8"></cou
    <name AggTagOnlyDiff="3" AggTagOnlyTotal="8"></name
    <firstname AggTagOnlyDiff="3" AggTagOnlyTotal="8"></fi
    <street AggTagOnlyDiff="3" AggTagOnlyTotal="8"></street
    <housenr AggTagOnlyDiff="3" AggTagOnlyTotal="7"></hou
    <postcode AggTagOnlyDiff="3" AggTagOnlyTotal="7"></pc
    <city AggTagOnlyDiff="3" AggTagOnlyTotal="7"></city>
    <state AggTagOnlyDiff="3" AggTagOnlyTotal="7"></state>
    <custnr AggTagOnlyDiff="2" AggTagOnlyTotal="8"></custr
  </record>
</table>
```

Illustration 12: aggregated analysis

In Illustration 13: aggregated analysis of compare depth 2 details of the analysis have been omitted by limiting the maximum compare depth to level 2. On the left side e.g. one cannot see any nodes hierarchically under the work_places tag, even though there are 5 differences. Those details have been omitted.


DREAM Analysis
Source Help
Differences on left side only
Differences on both sides
Differences on right side only
select all
copy from
diff
val

```
<staff id="1001" lang="EN" date="02042014" country="England"
   <firstname AggTagAttrDiffs="2" AggTagAttrTotal="2"></firstn
   <h:tipname AggTagAttrDiffs="1" AggTagAttrTotal="1"></h:tip
   <salary AggTagAttrDiffs="1" AggTagAttrTotal="2"></salary>
   <work_places AggTagAttrDiffs="5" AggTagAttrTotal="5">
   </work_places>
   <reader AggTagAttrDiffs="1" AggTagAttrTotal="1">
   </reader>
</staff>
```

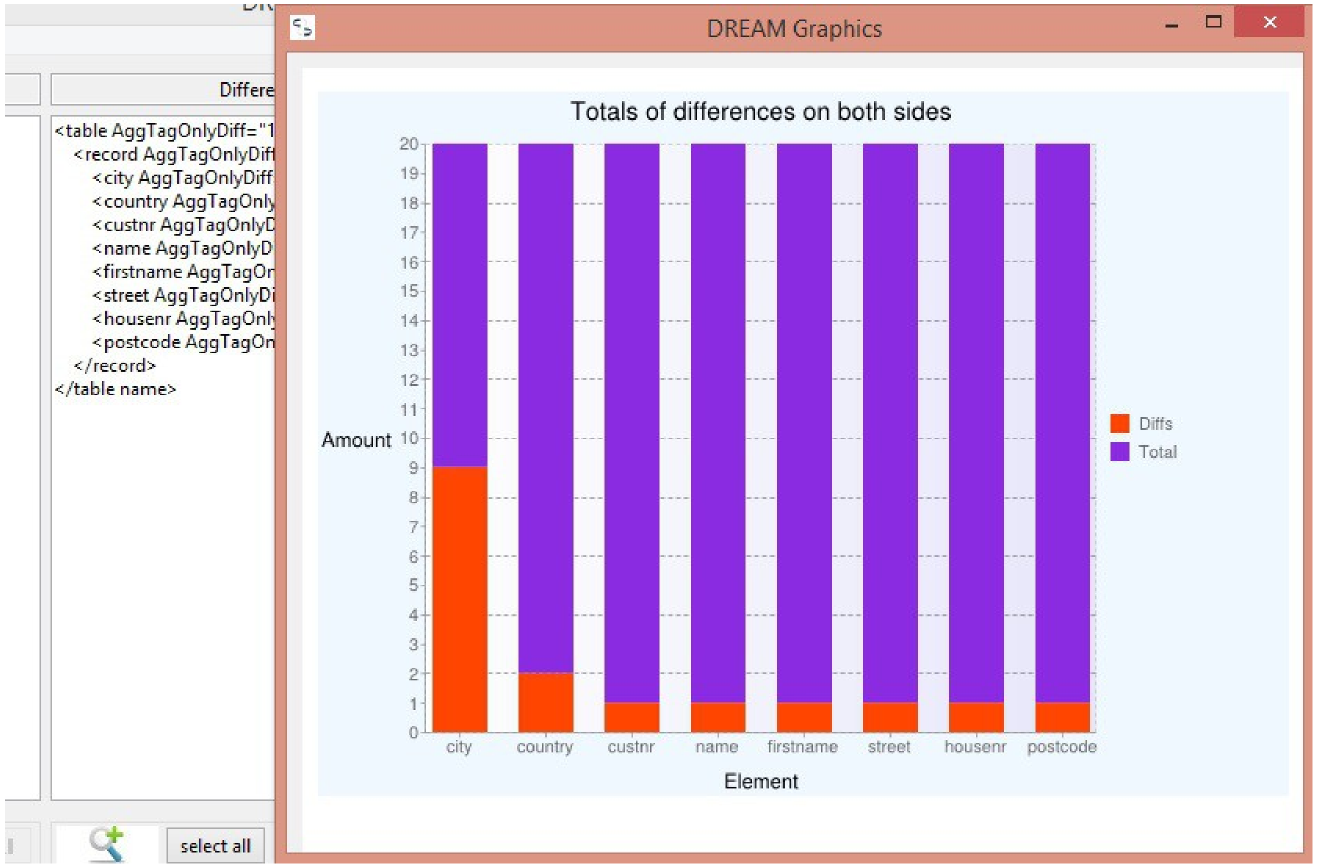
```
<staff id="1001" lang="EN" date="02042014" country="England"
   <address AggTagAttrDiffs="2" AggTagAttrTotal="3">
   </address>
</staff>
```

```
<staff id="1001" lang="EN" date="02042014" country="England"
   <firstname AggTagAttrDiffs="1" AggTagAttrTotal="1"></firstn
   <address AggTagAttrDiffs="2" AggTagAttrTotal="5">
   </address>
   <tax AggTagAttrDiffs="1" AggTagAttrTotal="1"></tax>
</staff>
```

Illustration 13: aggregated analysis of compare depth 2

Graphics dialogue

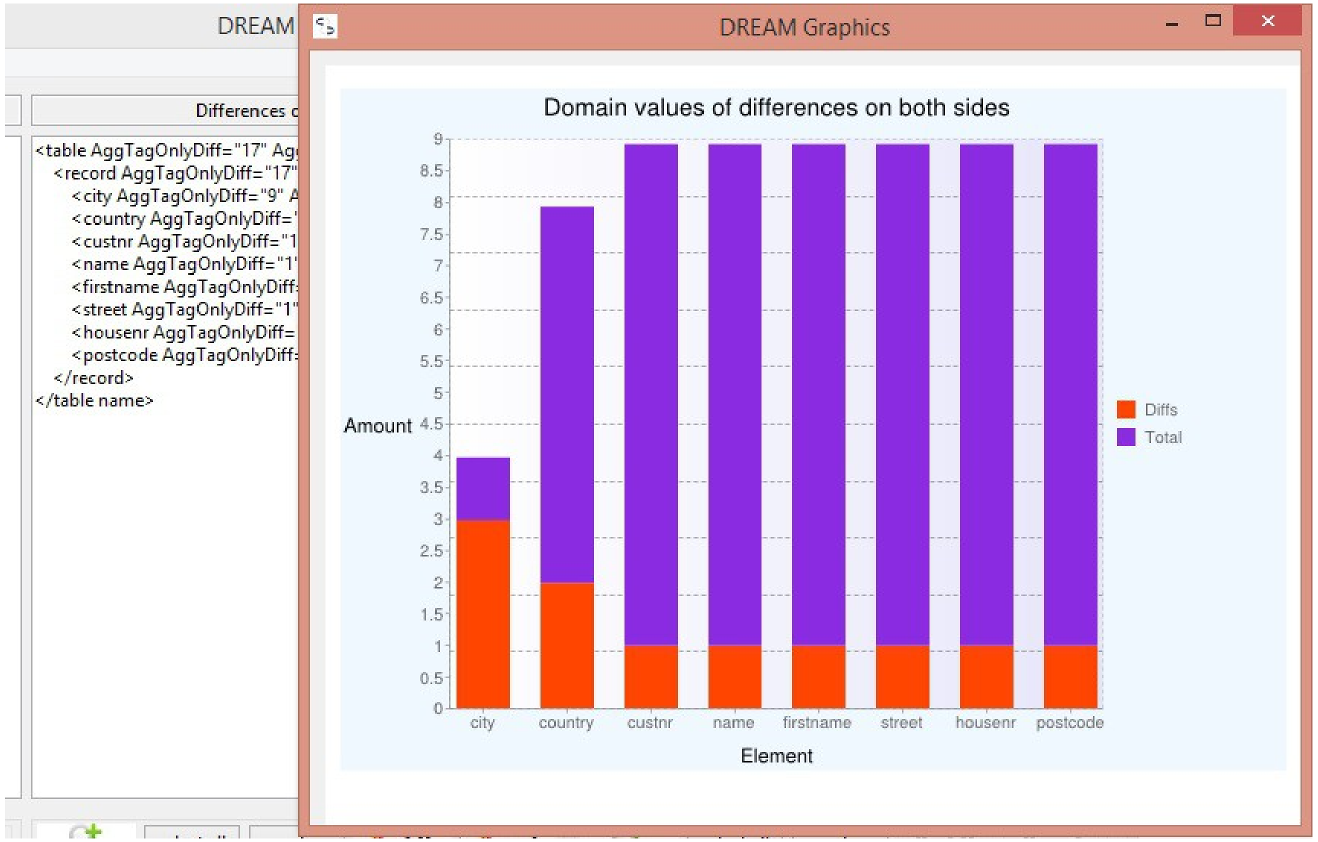
The graphics dialogue is a great means to get a visual impression of the quantity of differences. From Illustration 14: graphics of totals it can be concluded that the entity city suffers most from differences. Further investigation of differences of city has the highest priority.



```
<table AggTagOnlyDiff="1
  <record AggTagOnlyDiff
    <city AggTagOnlyDiff
    <country AggTagOnly
    <custnr AggTagOnlyD
    <name AggTagOnlyD
    <firstname AggTagOr
    <street AggTagOnlyDi
    <housenr AggTagOnly
    <postcode AggTagOn
  </record>
</table name>
```

Illustration 14: graphics of totals

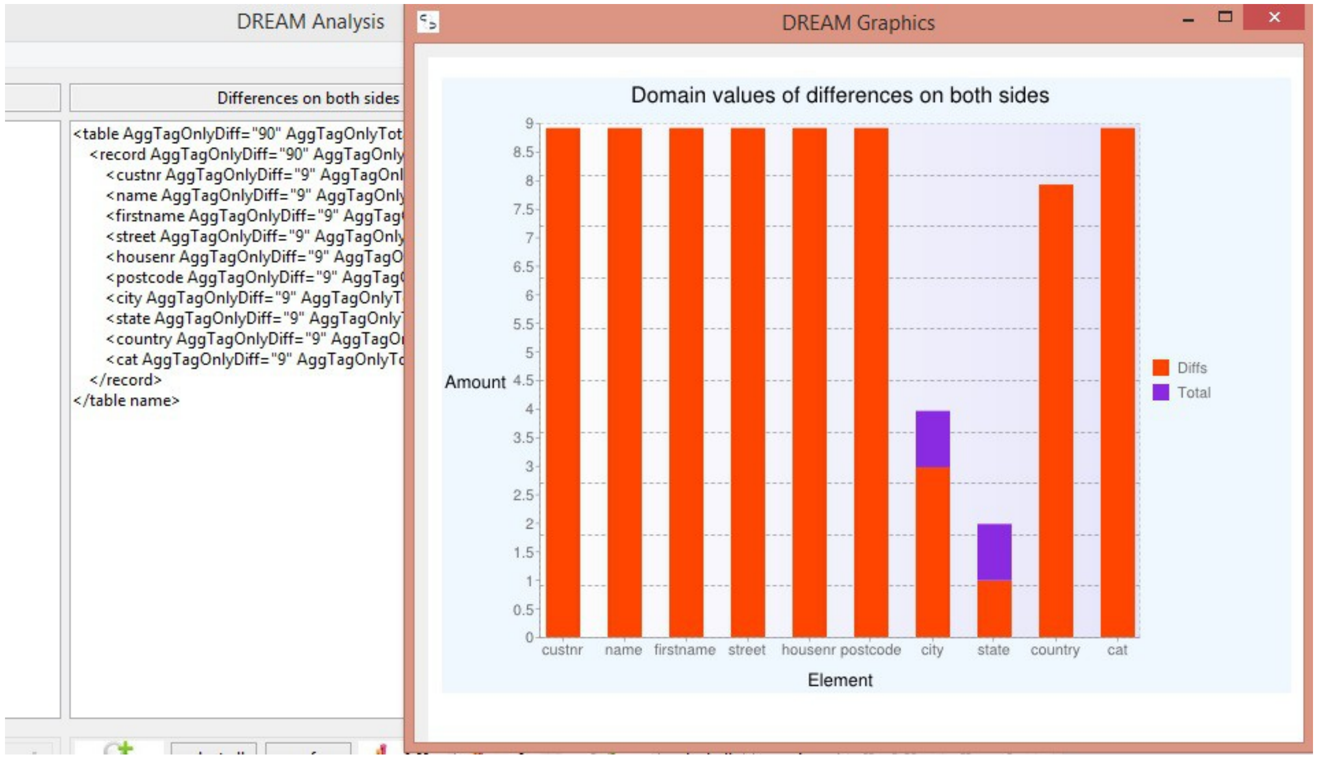
Another interesting conclusion can be derived from the domain analysis graphics below (see Illustration 15: graphics of domain analysis). Here we find that 75% (3 out of 4) of the cities have differences. Differences of other entities seems to occur less broad and more specific for special domain values.



```
<table AggTagOnlyDiff="17" Ag
  <record AggTagOnlyDiff="17"
    <city AggTagOnlyDiff="9" A
    <country AggTagOnlyDiff="
    <custnr AggTagOnlyDiff="1
    <name AggTagOnlyDiff="1"
    <firstname AggTagOnlyDiff
    <street AggTagOnlyDiff="1"
    <housenr AggTagOnlyDiff=
    <postcode AggTagOnlyDiff
  </record>
</table name>
```

Illustration 15: graphics of domain analysis

Now, continuing the investigation of the cause of differences (see Illustration 16: graphics of context analysis), one can see that when taking the entity state in the context (Broad Context Record based Analysis, see Illustration 10: center dialogue) the differences of city occur just for one of the two states. And indeed when looking up the demo source files, it seems that city differs always for South-Holland but never for North-Holland.

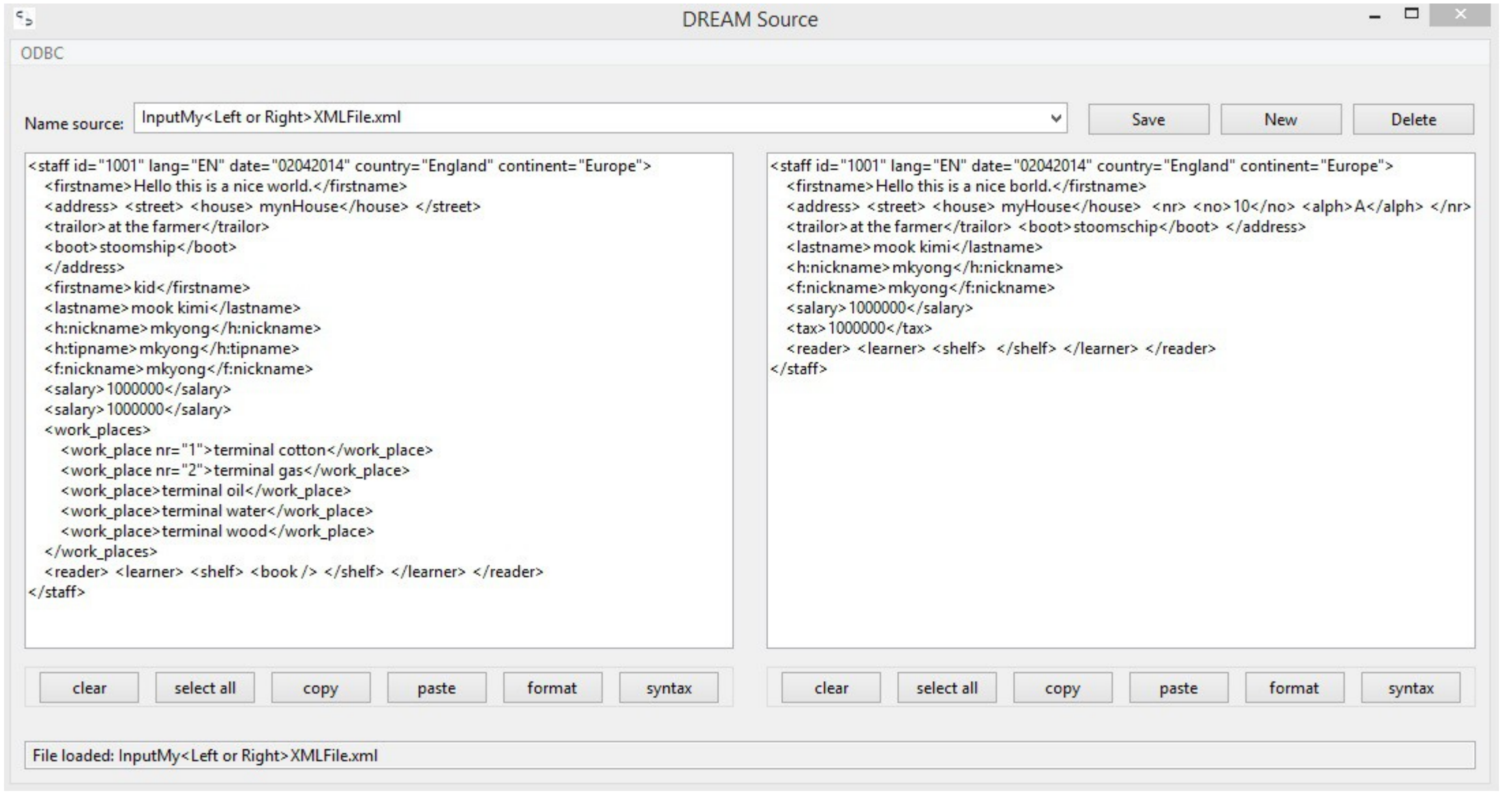


```
<table AggTagOnlyDiff="90" AggTagOnlyTot
  <record AggTagOnlyDiff="90" AggTagOnly
    <custnr AggTagOnlyDiff="9" AggTagOnl
    <name AggTagOnlyDiff="9" AggTagOnly
    <firstname AggTagOnlyDiff="9" AggTag
    <street AggTagOnlyDiff="9" AggTagOnly
    <housenr AggTagOnlyDiff="9" AggTagO
    <postcode AggTagOnlyDiff="9" AggTag
    <city AggTagOnlyDiff="9" AggTagOnlyT
    <state AggTagOnlyDiff="9" AggTagOnly
    <country AggTagOnlyDiff="9" AggTagO
    <cat AggTagOnlyDiff="9" AggTagOnlyT
  </record>
</table name>
```

Illustration 16: graphics of context analysis

Source dialogue

The source dialogue (see Illustration 17: source file in XML format) is used to make, edit, save, format and check pairs of source files. Press the new button to make a pair of files from scratch. The text can be copied from other editors and pasted into the source dialogue by using the paste button. The clear button can be used to remove the content, select all and copy can be used to get text out of the source dialogue. With the format button, the XML code will be formatted and finally one can check the syntactical correctness by pressing the syntax button.



Name source: InputMy<Left or Right>XMLFile.xml

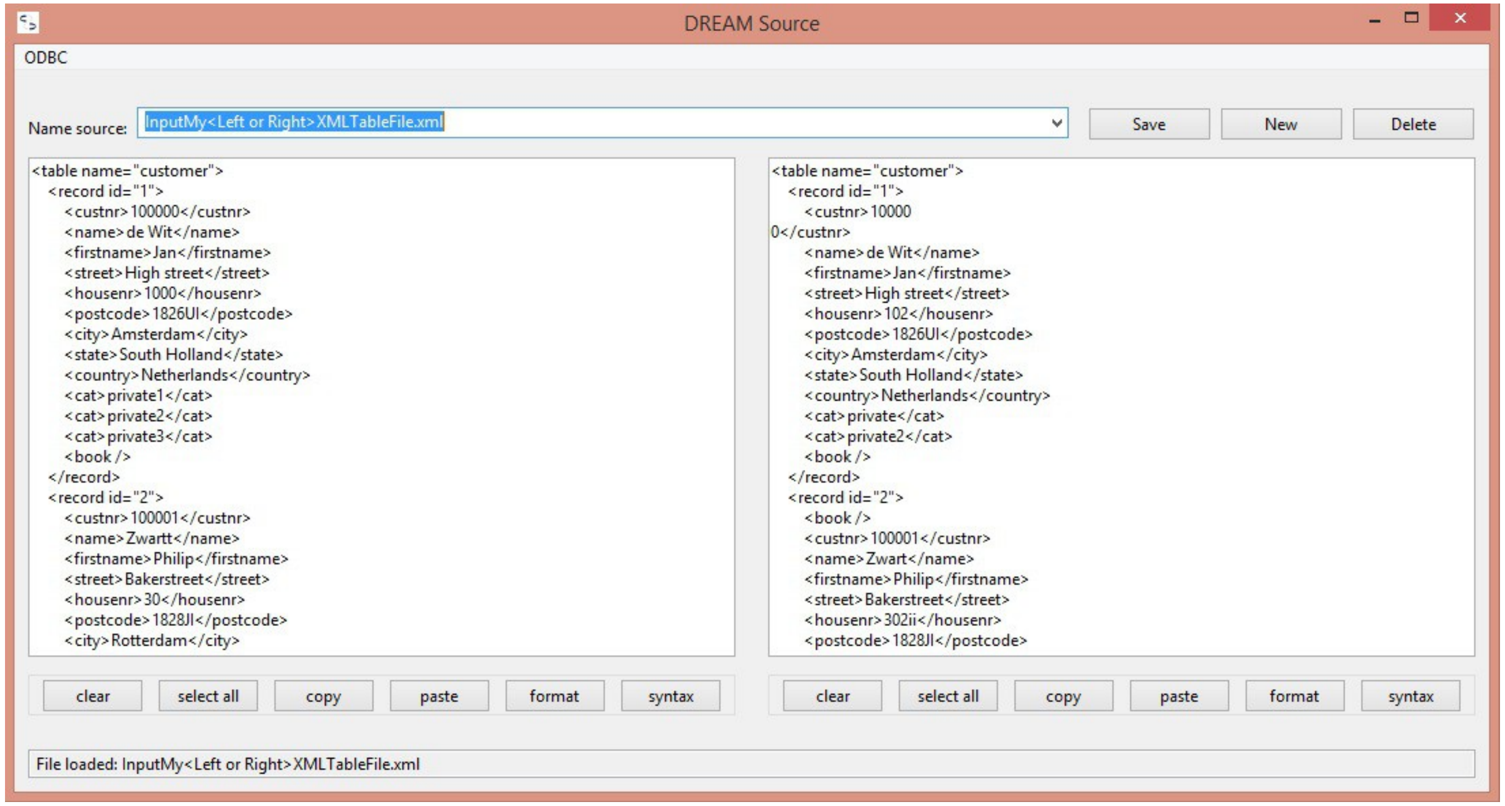
```xml
<staff id="1001" lang="EN" date="02042014" country="England" continent="Europe">
  <firstname>Hello this is a nice world.</firstname>
  <address> <street> <house> mynHouse</house> </street>
  <trailor>at the farmer</trailor>
  <boot>stoomship</boot>
  </address>
  <firstname>kid</firstname>
  <lastname>mook kimi</lastname>
  <h:nickname>mkyong</h:nickname>
  <h:tipname>mkyong</h:tipname>
  <f:nickname>mkyong</f:nickname>
  <salary>1000000</salary>
  <salary>1000000</salary>
  <work_places>
    <work_place nr="1">terminal cotton</work_place>
    <work_place nr="2">terminal gas</work_place>
    <work_place>terminal oil</work_place>
    <work_place>terminal water</work_place>
    <work_place>terminal wood</work_place>
  </work_places>
  <reader> <learner> <shelf> <book /> </shelf> </learner> </reader>
</staff>
```

```xml
<staff id="1001" lang="EN" date="02042014" country="England" continent="Europe">
  <firstname>Hello this is a nice borld.</firstname>
  <address> <street> <house> myHouse</house>  <nr> <no>10</no> <alph>A</alph> </nr>
  <trailor>at the farmer</trailor> <boot>stoomschip</boot> </address>
  <lastname>mook kimi</lastname>
  <h:nickname>mkyong</h:nickname>
  <f:nickname>mkyong</f:nickname>
  <salary>1000000</salary>
  <tax>1000000</tax>
  <reader> <learner> <shelf>  </shelf> </learner> </reader>
</staff>
```

File loaded: InputMy<Left or Right>XMLFile.xml

Illustration 17: source file in XML format

In Illustration 18: source file in DB format a pair of XML files is shown. The file has the standard structure for relational databases which is as DB format recognized by DREAM (see DREAM DB format syntax). Those files do not differ from other XML files except for static format. It is up to the DREAM user, to know that they are structured as DB files which makes them more useful for root cause and context analysis.



Name source: InputMy<Left or Right>XMLTableFile.xml

```
<table name="customer">
  <record id="1">
    <custnr>100000</custnr>
    <name>de Wit</name>
    <firstname>Jan</firstname>
    <street>High street</street>
    <housenr>1000</housenr>
    <postcode>1826Ul</postcode>
    <city>Amsterdam</city>
    <state>South Holland</state>
    <country>Netherlands</country>
    <cat>private1</cat>
    <cat>private2</cat>
    <cat>private3</cat>
    <book />
  </record>
  <record id="2">
    <custnr>100001</custnr>
    <name>Zwartt</name>
    <firstname>Philip</firstname>
    <street>Bakerstreet</street>
    <housenr>30</housenr>
    <postcode>1828Jl</postcode>
    <city>Rotterdam</city>
```

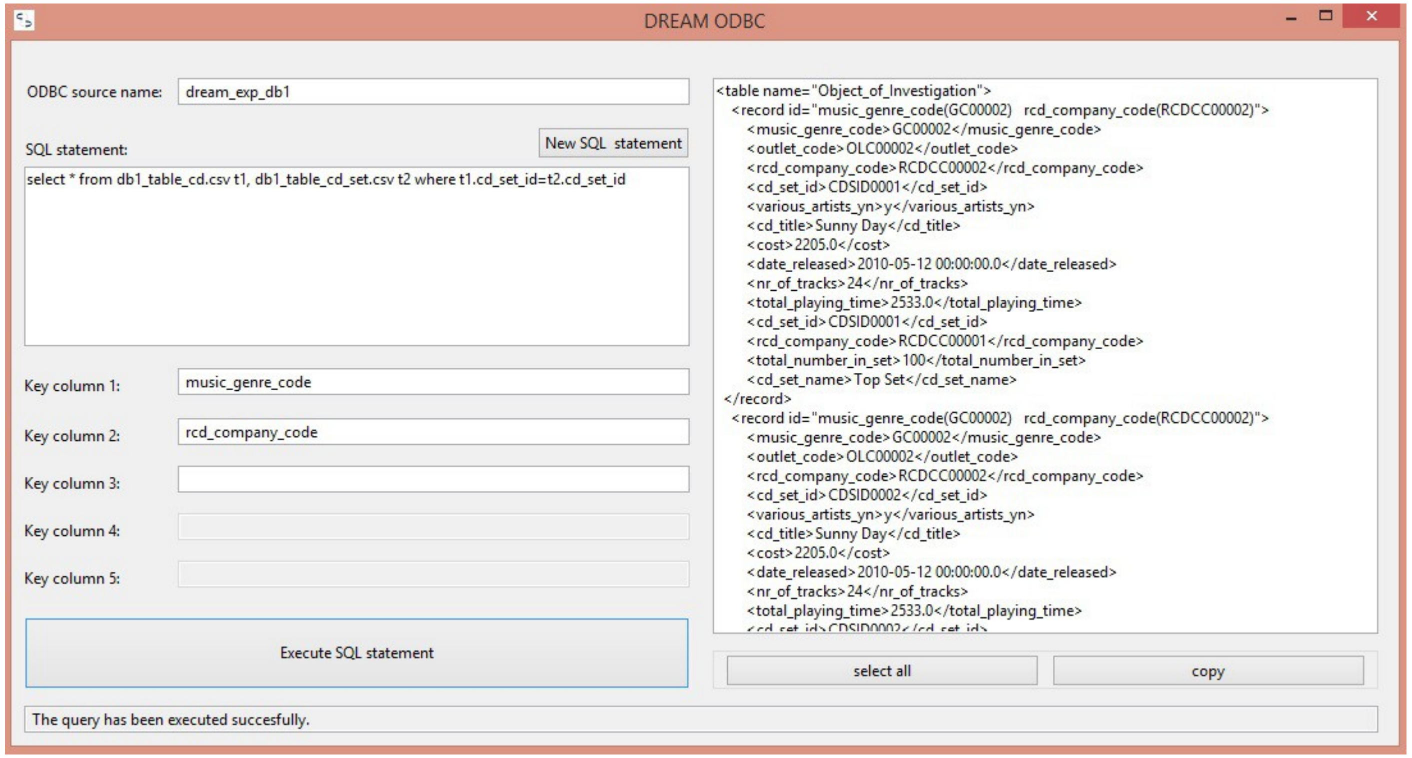
```
<table name="customer">
  <record id="1">
    <custnr>10000
0</custnr>
    <name>de Wit</name>
    <firstname>Jan</firstname>
    <street>High street</street>
    <housenr>102</housenr>
    <postcode>1826Ul</postcode>
    <city>Amsterdam</city>
    <state>South Holland</state>
    <country>Netherlands</country>
    <cat>private</cat>
    <cat>private2</cat>
    <book />
  </record>
  <record id="2">
    <book />
    <custnr>100001</custnr>
    <name>Zwart</name>
    <firstname>Philip</firstname>
    <street>Bakerstreet</street>
    <housenr>302ii</housenr>
    <postcode>1828Jl</postcode>
```

File loaded: InputMy<Left or Right>XMLTableFile.xml

Illustration 18: source file in DB format

ODBC dialogue

The ODBC dialogue can be reached from the source dialogue. This dialogue (see Illustration 19: ODBC dialogue) offers the functionality to extract data from data sources like relational databases (RDBMS), NoSQL databases, CSV files, etc. First, an ODBC data source has to be created in Windows. Then specify the name of the ODBC source, an SQL query and a composed key for record identification. After pressing the execution button the result appears on the right side. The data can be selected and copied to the source dialogue.



ODBC source name: dream_exp_db1

SQL statement:

```
select * from db1_table_cd.csv t1, db1_table_cd_set.csv t2 where t1.cd_set_id=t2.cd_set_id
```

Key column 1: music_genre_code

Key column 2: rcd_company_code

Key column 3:

Key column 4:

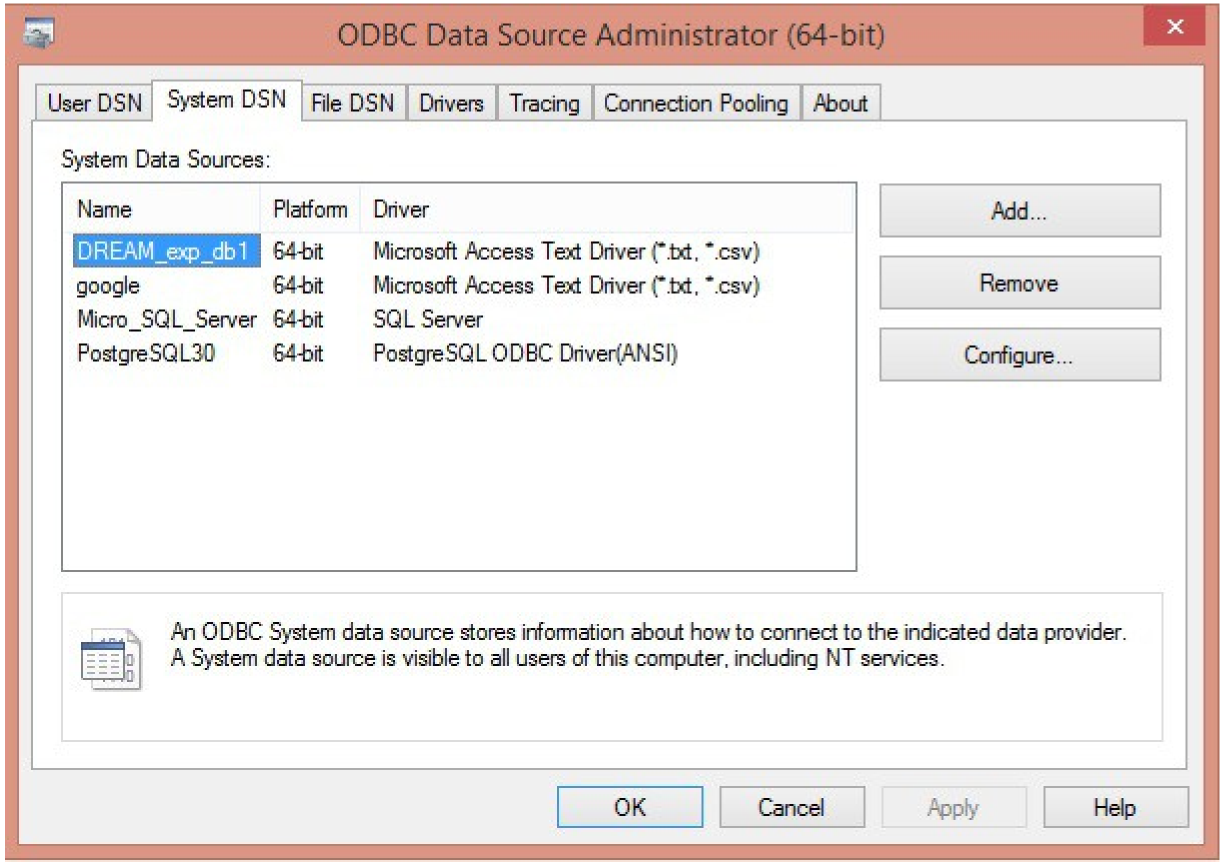
Key column 5:

```
<table name="Object_of_Investigation">
  <record id="music_genre_code(GC00002)   rcd_company_code(RCDCC00002)">
    <music_genre_code>GC00002</music_genre_code>
    <outlet_code>OLC00002</outlet_code>
    <rcd_company_code>RCDCC00002</rcd_company_code>
    <cd_set_id>CDSID0001</cd_set_id>
    <various_artists_yn>y</various_artists_yn>
    <cd_title>Sunny Day</cd_title>
    <cost>2205.0</cost>
    <date_released>2010-05-12 00:00:00.0</date_released>
    <nr_of_tracks>24</nr_of_tracks>
    <total_playing_time>2533.0</total_playing_time>
    <cd_set_id>CDSID0001</cd_set_id>
    <rcd_company_code>RCDCC00001</rcd_company_code>
    <total_number_in_set>100</total_number_in_set>
    <cd_set_name>Top Set</cd_set_name>
 </record>
  <record id="music_genre_code(GC00002)   rcd_company_code(RCDCC00002)">
    <music_genre_code>GC00002</music_genre_code>
    <outlet_code>OLC00002</outlet_code>
    <rcd_company_code>RCDCC00002</rcd_company_code>
    <cd_set_id>CDSID0002</cd_set_id>
    <various_artists_yn>y</various_artists_yn>
    <cd_title>Sunny Day</cd_title>
    <cost>2205.0</cost>
    <date_released>2010-05-12 00:00:00.0</date_released>
    <nr_of_tracks>24</nr_of_tracks>
    <total_playing_time>2533.0</total_playing_time>
```

The query has been executed succesfully.

Illustration 19: ODBC dialogue

In the Windows control panel one can make ODBC sources that can be used by DREAM. In Illustration 20: Windows ODBC source there is a ODBC source to a CSV file.


ODBC Data Source Administrator (64-bit)

User DSN | System DSN | File DSN | Drivers | Tracing | Connection Pooling | About

System Data Sources:

Name	Platform	Driver
DREAM_exp_db1	64-bit	Microsoft Access Text Driver (*.txt, *.csv)
google	64-bit	Microsoft Access Text Driver (*.txt, *.csv)
Micro_SQL_Server	64-bit	SQL Server
PostgreSQL30	64-bit	PostgreSQL ODBC Driver(ANSI)

Add... Remove Configure...

An ODBC System data source stores information about how to connect to the indicated data provider. A System data source is visible to all users of this computer, including NT services.

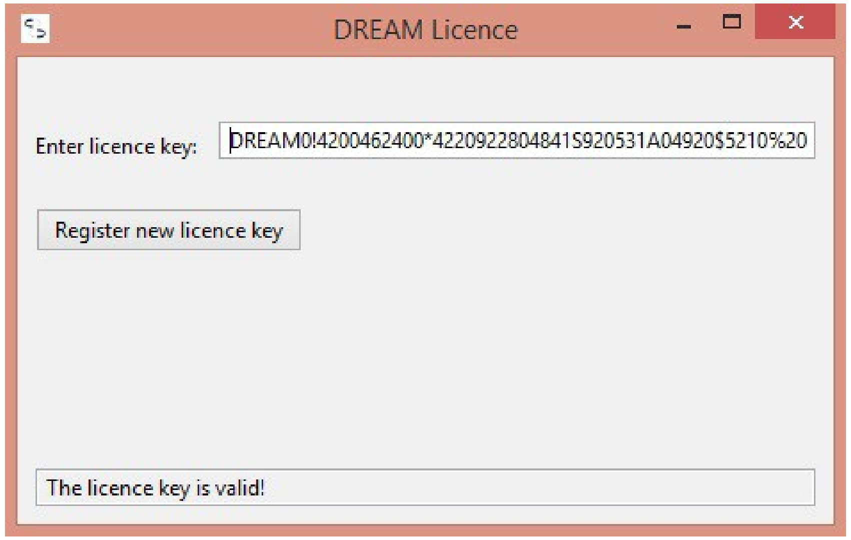
OK Cancel Apply Help

Illustration 20: Windows ODBC source

Licence dialogue

As DREAM is proprietary software or closed source software which can be used with a licence. The licence key can be obtained from support@drost.name

When the licence key has expired the user will get a message after clicking the start button on the center dialogue (Illustration 10: center dialogue). The licence dialogue gives a message if the licence is valid or not. After receiving a licence key, it can be entered into the licence dialogue (see Illustration 21: licence dialogue). Then press the button 'register new licence key' and check if the licence is valid.



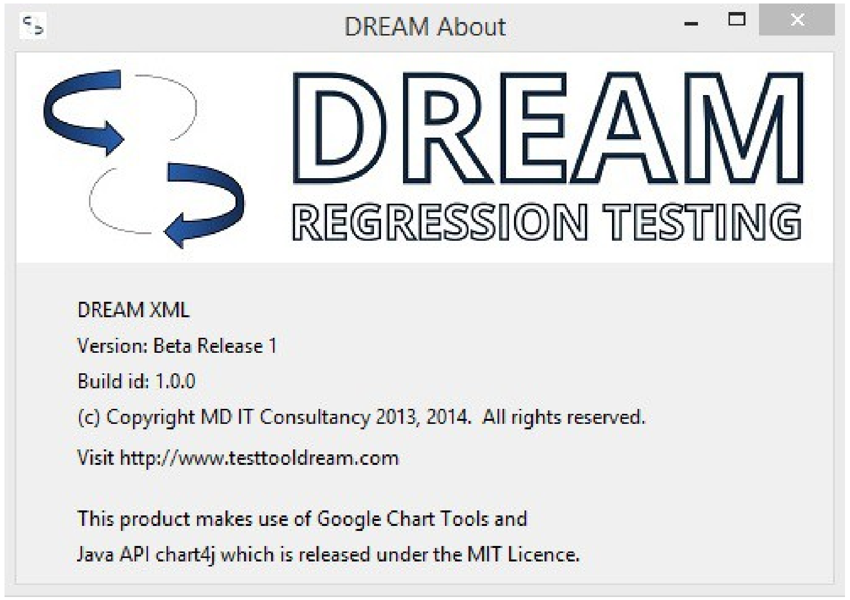
Enter licence key: DREAM0!4200462400*4220922804841S920531A04920$5210%20

The licence key is valid!

Illustration 21: licence dialogue

About dialogue

The about dialogue (see Illustration 22: about box) informs you about licensing and revision information of the software product.



DREAM XML

Version: Beta Release 1

Build id: 1.0.0

(c) Copyright MD IT Consultancy 2013, 2014. All rights reserved.

Visit http://www.testtooldream.com

This product makes use of Google Chart Tools and
Java API chart4j which is released under the MIT Licence.

Illustration 22: about box

Glossary

[DREAM] Name of the software product DREAM. DREAM XML can be used as synonym.
[Identification] Identification is the capability to find, retrieve, report, change, or delete specific data without ambiguity.
[Intersection] The intersection $A \cap B$ of two sets A and B is the set that contains all elements of A that also belong to B, but no other elements.
[Leaf (tree)] A node with no children
[Level (tree)] The level of a node is defined by 1 + the number of connections between the node and the root.
[NoSQL] Mechanism for storage and retrieval of data that is modeled in means other than the tabular relations
[Path (tree)] a sequence of nodes and edges connecting a node with a descendant.
[Relational database] A relational database is a database that stores information about both the data and how it is related.
[Root (tree)] The top most node in a tree.
[Set Theory] Set theory is the branch of mathematical logic that studies sets, which are collections of objects.
[Tag] In information systems, a tag is a non-hierarchical keyword or term assigned to a piece of information.
[Tree] A tree is a non-linear data structure that consists of a root node and potentially many levels of additional nodes that form a hierarchy.
[XML] Markup language that defines a set of rules for encoding documents in a format that is both human-readable and machine-readable.

Bibliography

- Kanagaraj.Sand Dr.Sunitha Abburu, Converting Relational Database Into Xml Document, IJCSI , Vol.9,Issue 2, No 1, March 2012
- Set theory (From Wikipedia, the free encyclopedia)
- Can elements in a set be duplicated?[1], Mathematics Stack Exchange
- Intersection (set theory) (From Wikipedia, the free encyclopedia)
- Identification (information) (From Wikipedia, the free encyclopedia)
- Relational Database (From Wikipedia, the free encyclopedia)
- Tree data structure (From Wikipedia, the free encyclopedia)
- Tag meta data (From Wikipedia, the free encyclopedia)
- XML Turorial W3C, XML Syntax Rules

1In formal set theory, each element counts as one even though they may appear more than once, so that {1}={1,1}. This is because if we let A={1} and B={1,1}, then x∈A@x∈B, so that the two sets are equal by the very definition of equality of sets.
However, in elementary school texts treating sets in a more informal manner, the elements of a set are often taken to mean something else. For example, for the word contrast, its letters may be taken to be c,o,n,t,r,a,s,t where there is the implicit supposition that one is dealing with c,o,n,t1,r,a,s,t2, so that the set actually has 8 elements instead of 7.

Appendix

DREAM DB format syntax

```
<table name="specify table name">
  <record id="specify unique identifier">
    <attribute 1>   </attribute 1>
    ...
    <attribute N>   </attribute N>
  </record id>
  ...
  <record id="specify unique identifier">
    <attribute 1>   </attribute 1>
    ...
    <attribute N>   </attribute N>
  </record id>
</table name>
```

DREAM's competitors

	Line based compare	*XML based compare*	*Set recognition functionality*	*Relational DB compare*	*Aggregation functionality*	*Domain analysis*	*Context cause analysis*	*Graphical presentation*
DREAM	No	Yes	Yes	Yes	Yes	Yes	Yes	Yes
Unix diff	Yes	No	No	No	No	No	No	No
Notepad ++	No	Yes	No	No	No	No	No	No
Git	Yes	No	No	No	No	No	No	No
Altova	No	Yes	No	No	No	No	No	No
diffxml	Yes	Yes	No	No	No	No	No	No