Horizon 2020 Program (2014-2020)

Big data PPP

Research addressing main technology challenges of the data economy

Industrial-Driven Big Data as a Self-Service Solution

D2.2: The Data Fabrication Platform (DFP, Interim version)[†]

Abstract: This report contains details about the modeling automation methodology for fabrication of synthetic and realistic data sets. This work described in this document is the first step towards complete automation. It focuses on data and metadata logic extraction from real data and on the way this data is used to create fabrication rules that the platform provides.

Contractual Date of Delivery	31/12/2018
Actual Date of Delivery	31/12/2018
Deliverable Security Class	Public
Editor	*Omer Boehm (IBM)*
Contributors	*IBM*
Quality Assurance	*Dr. Gerald Ristow (SAG)* *Ramon Martin de Pozuelo Genis (CAIXA)* *Dr. Kostas Lampropoulos (FORTH)*

[†] *The research leading to these results has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 780787.*

The *I-BiDaaS* Consortium

Foundation for Research and Technology – Hellas (FORTH)	Coordinator	Greece
Barcelona Supercomputing Center (BSC)	Principal Contractor	Spain
IBM Israel – Science and Technology LTD (IBM)	Principal Contractor	Israel
Centro Ricerche FIAT (FCA/CRF)	Principal Contractor	Italy
Software AG (SAG)	Principal Contractor	Germany
Caixabank S.A. (CAIXA)	Principal Contractor	Spain
University of Manchester (UNIMAN)	Principal Contractor	United Kingdom
Ecole Nationale des Ponts et Chaussees (ENPC)	Principal Contractor	France
ATOS Spain S.A. (ATOS)	Principal Contractor	Spain
Aegis IT Research LTD (AEGIS)	Principal Contractor	United Kingdom
Information Technology for Market Leadership (ITML)	Principal Contractor	Greece
University of Novi Sad Faculty of Sciences (UNSPMF)	Principal Contractor	Serbia
Telefonica Investigation y Desarrollo S.A. (TID)	Principal Contractor	Spain

Document Revisions & Quality Assurance

Internal Reviewers

1. *Dr. Gerald Ristow,* SAG
2. *Ramon Martin de Pozuelo Genis,* CAIXA
3. *Dr. Kostas Lampropoulos, FORTH*

Revisions

Version	Date	By	Overview
0.0.8	21/12/2018	Quality Assurance, Internal reviewers	Final review and approval
0.0.7	20/12/2018	Editor	Incorporation of the final comments
0.0.6	17/12/2018	Ramon Martin De Pozuelo Genis	Completed review by CAIXA
0.0.5	14/12/2018	Dr. Gerald Ristow	Completed review by SAG
0.0.4	11/12/2018	Quality Assurance, Internal reviewers	Comments on the first draft
0.0.3	04/12/2018	Editor	First draft
0.0.2	02/11/2018	Quality Assurance, Internal reviewers	Comments on the ToC first draft
0.0.1	22/10/2018	Editor	ToC First Draft

Table of Contents

List of Abbreviations

TDF – Test Data Fabrication

JSON – JavaScript Object Notation

MQTT – Message Queuing Telemetry Transport

MVP – Minimum Viable Product

UM – Universal Messaging

CSP – Constraint satisfaction problem

XLS – Microsoft excel file

CSV – Comma separated values

XML - Extensible Mark-up Language

UML – Unified Modelling Language

PK – Primary Key

FK – Foreign Key

List of Figures

Executive Summary

The deliverable provided an automated methodology for fabrication of synthetic and realistic data sets using TDF (Test data fabrication platform) by IBM.

The generated synthetic data solves the problem of security and leakage of sensitive data to unauthorized third parties and provides data for early exploration and development phases of the applications in cases where real data does not exist, cannot be shared or not available yet.

This document describes the design and approach of TDF to fabricate synthetic data that is as realistic and similar as possible to the underlying provided real data.

For that end, the TDF platform uses various types of variables and constraint rules that model the data and the relationships in it. An important feature that was developed to ease the modeling task is using both the metadata logic and the underlying real data to learn the data characteristics and to automatically create the required data model.

Fabricated data can be sent to various types of files (Positional Flat, XLS, CSV, XML, JSON), data bases (DB2, Oracle, PostgreSQL, SQLite, Apache Cassandra), or directly streamed (JSON via MQTT protocol) to messaging brokers like SAG's Universal Messaging.

1 Introduction

This document is the second deliverable of WP2, which addresses data curation ingestion and pre-processing. It supports the user requirements and MVP needs and interactions expected between the I-BiDaaS actors and the offered environment. The work, which is presented in this deliverable, relates to the work performed in WP3 about the evaluation and validation of the platform through real-life industrial experiments.

The purpose of this deliverable is to provide an automated methodology for the fabrication of synthetic and realistic data sets using IBM's technology.

Constructing a data model is done via constraint rules writing. This is a difficult and time-consuming task. The modeler must be highly familiarized with the solver language and the available constraints, but more importantly, to deeply understand what the generated data must look like, in such a way that tested applications will be able to use them seamlessly.

The deliverable includes an introduction to TDF and a description of how it uses a CSP solver to generate data, a description of the work done on automating the modelling process and description what are the TDF role and interactions within I-BiDaaS.

The position of the TDF module with respect to the overall I-BiDaaS platform is described next (Figure 1). The module interacts directly with the Universal Messaging bus, and indirectly with APAMA streaming analytics, and the batch processing layer which contains the COMPSs model and the advanced ML model, including the Hecuba database which is the main database system of the I-BiDaaS platform.

Analytics results also flow through the Universal Messaging and are delivered to the data fabrication platform. These results, as well as additional metadata, are used for assisting the automation process of the data modeling.

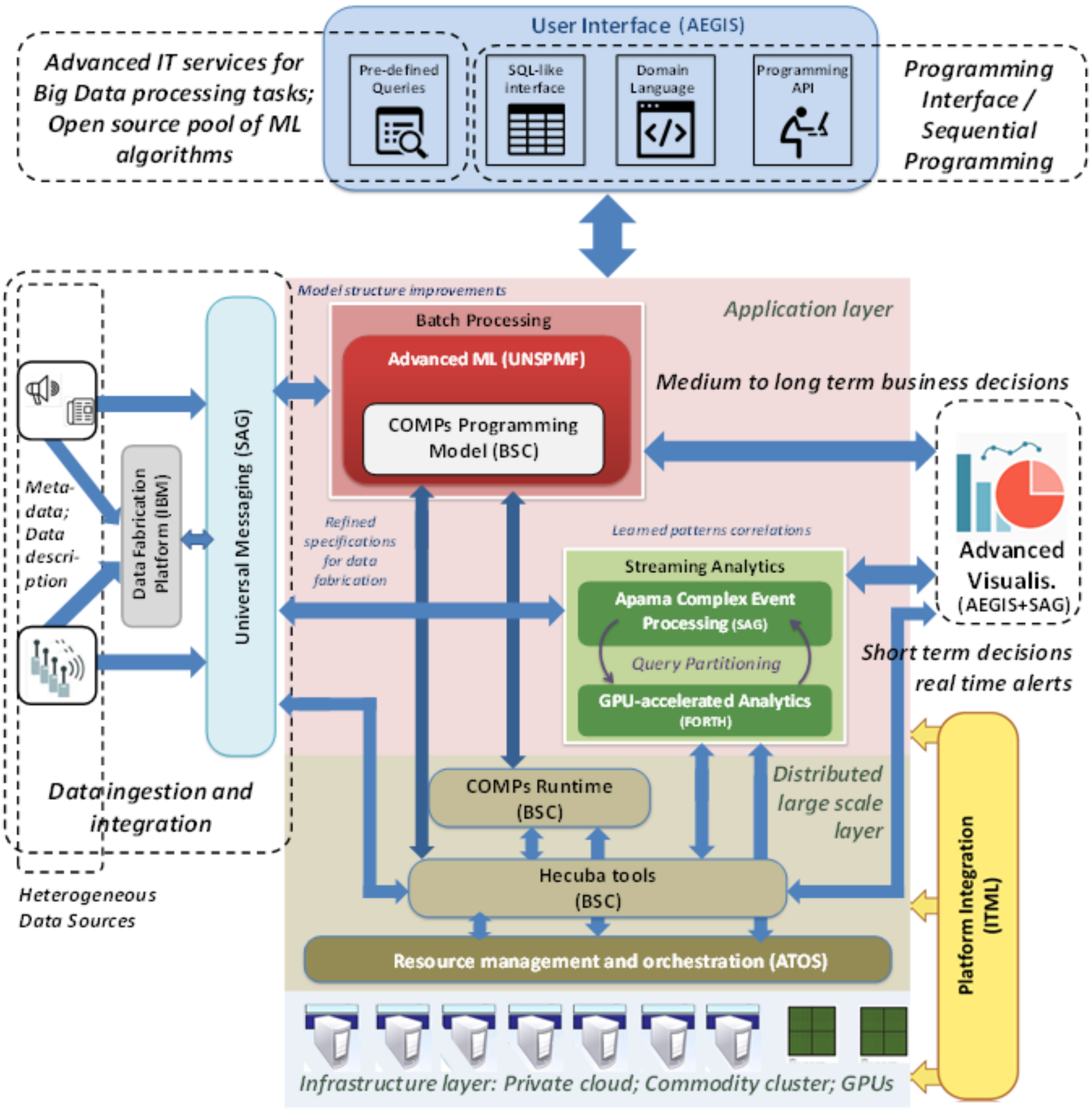
User Interface (AEGIS)
Advanced IT services for Big Data processing tasks; Open source pool of ML algorithms
Pre-defined Queries
SQL-like interface
Domain Language
Programming API
Programming Interface / Sequential Programming
Model structure improvements
Batch Processing
Advanced ML (UNSPMF)
COMPs Programming Model (BSC)
Application layer
Medium to long term business decisions
Meta-data; Data descri-ption
Data Fabrication Platform (IBM)
Universal Messaging (SAG)
Refined specifications for data fabrication
Learned patterns correlations
Streaming Analytics
Apama Complex Event Processing (SAG)
Query Partitioning
GPU-accelerated Analytics (FORTH)
Advanced Visualis. (AEGIS+SAG)
Short term decisions real time alerts
Data ingestion and integration
Heterogeneous Data Sources
COMPs Runtime (BSC)
Distributed large scale layer
Hecuba tools (BSC)
Resource management and orchestration (ATOS)
Platform Integration (ITML)
Infrastructure layer: Private cloud; Commodity cluster; GPUs

Figure 1: The I-BiDaaS platform

2 Introduction to TDF

The IBM's Test Data Fabrication platform [1][2] is a web based central platform for generating high-quality data for testing, development, and training. The platform provides a consistent and organizational wide methodology for creating test data. The methodology used is termed "rule guided fabrication".

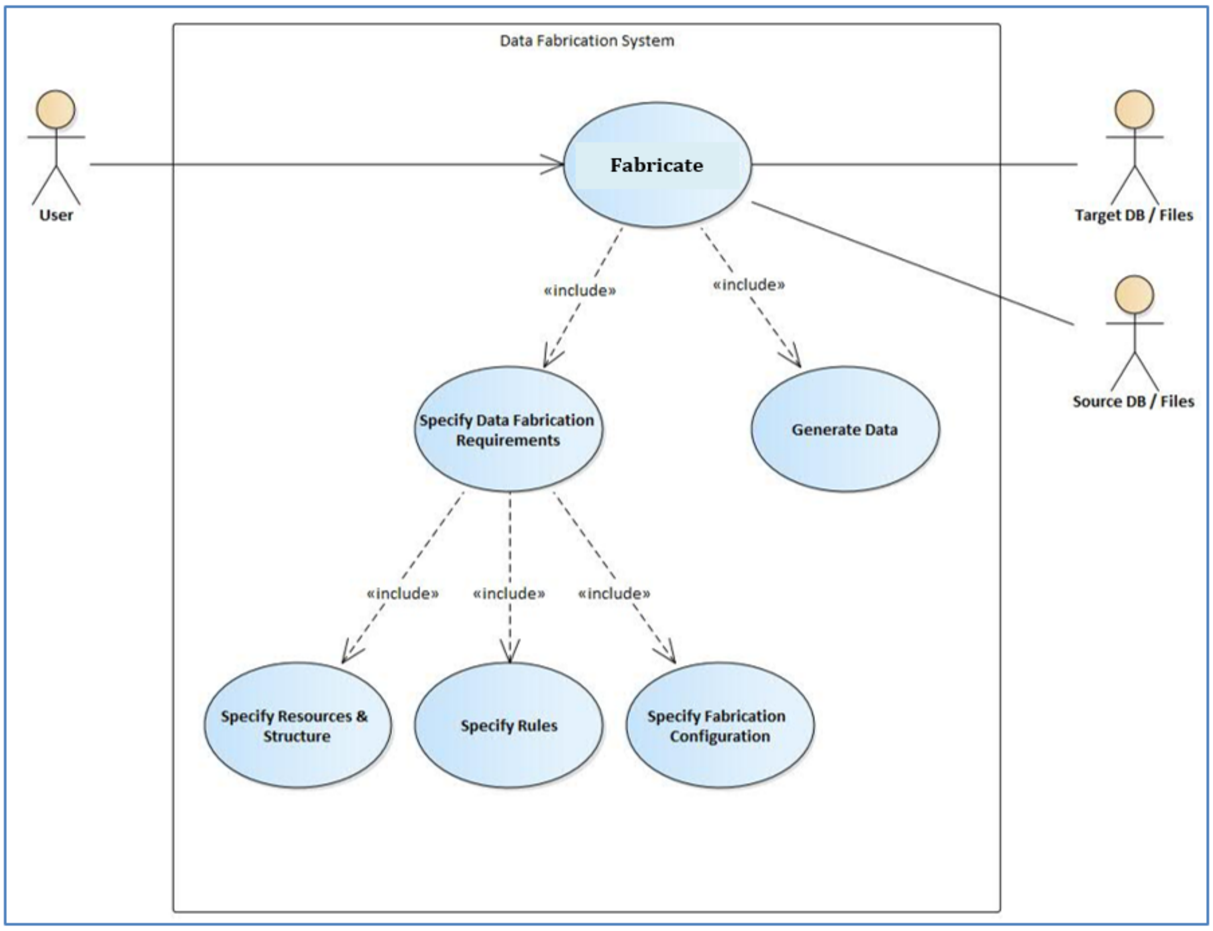

Figure 2: UML use case diagram for the TDF

The primary TDF use case for fabricating synthetic data contains two actors: a user (initiator) and Database/File (participant). This use case includes two sub-use cases: data modelling and data generation. The data modelling use case includes three sub-use cases: resources and structure definitions, constraint rules definitions and fabrication configuration definitions. The data structure, i.e., schema, tables, columns for data bases, is automatically imported however structural hierarchy of data elements (structs, arrays, tables, fields, and types need to be manually defined by the user. The constraint rules are required to construct a model of the data. Input and output resources are standard relational databases (e.g., DB2, Oracle, PostgreSQL, SQLite), standard file formats (e.g., Flat file, XLS, CSV, XML, JSON) and streaming via MQTT protocol.

In rule guided fabrication, the database logic is extracted automatically and is augmented by application logic and testing logic modelled by the user.

The application logic and the testing logic can be modelled using rules that the platform provides. The platform is also extendible and new rule types can be added by users and automatically integrated into the platform in an organization-wide manner.

Once the user requests the generation of a certain amount of data into a set of test databases, the platform internally ensures that the generated data satisfies the modelled rules as well as the internal databases consistency requirements.

The platform can generate data from scratch, inflate existing databases, move existing data, and transform data from previously existing resources, such as old test databases or even production data. The platform provides a comprehensive and hybrid solution that can create a mixture of synthetic and real data according to user requirements.

IBM Data Fabrication Platform uses a proprietary constraint satisfaction programming solver [3-24] that has been used for verifying IBM hardware systems for over a decade. The solver finds a solution for all the requirements in a data fabrication task. The solver is capable of handling very complex problems in a timely manner.

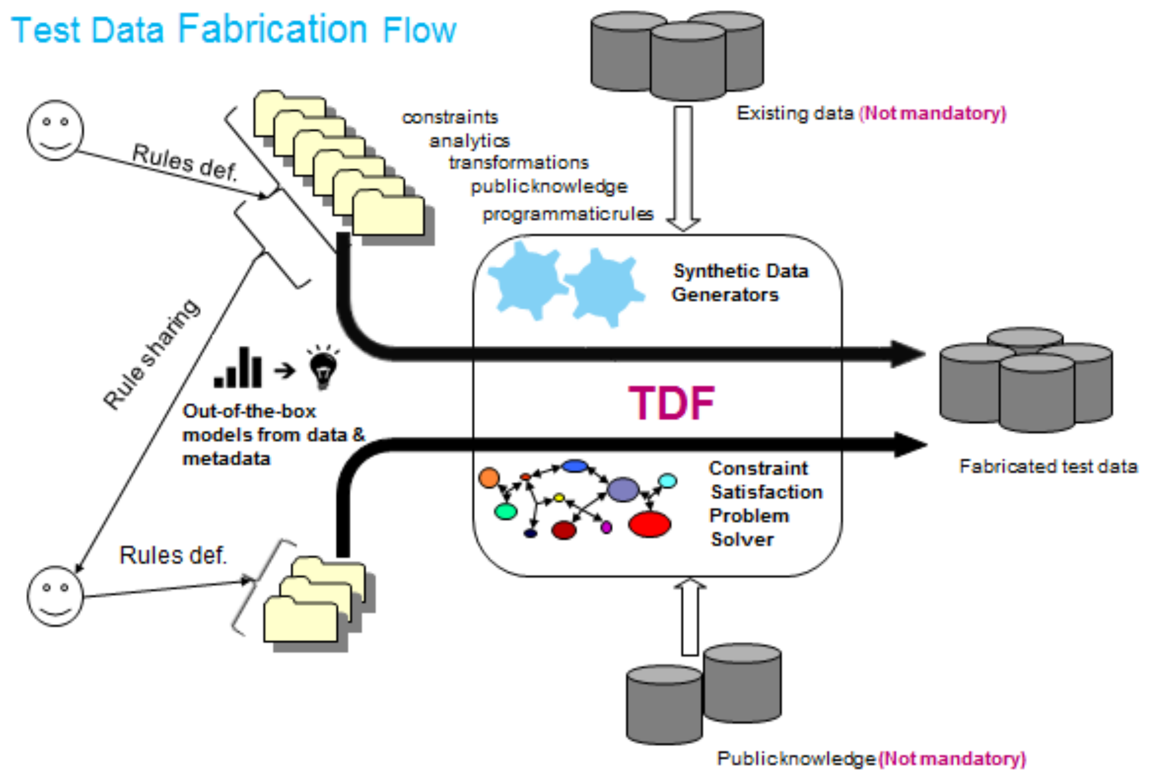
Test Data Fabrication Flow
Rules def.
constraints
analytics
transformations
public knowledge
programmatic rules
Existing data (Not mandatory)
Synthetic Data Generators
Rule sharing
Out-of-the-box models from data & metadata
TDF
Fabricated test data
Constraint Satisfaction Problem Solver
Rules def.
Public knowledge (Not mandatory)

Figure 3: Test Data Fabrication (TDF) platform flow

3 Synthetic Data Fabrication using CSP

To overcome the shortcomings of existing data generation techniques, TDF uses a solution that generates data using a Constraint Satisfaction Problem (CPS) solver. This methodology is generic and flexible for various types of use cases yet also very safe, as all user constraints must be satisfied. The solution does not require access to real or masked data, or to historic actual queries, which all might involve some violation of privacy regulations. Data generation can be constrained directly by the users. These constraints can direct the generation towards desired testing objectives or to realistic database statistics and query behavior.

The platform uses the database schema or the file hierarchical structure, the user requirements via variables and constraints and a fabrication configuration specifying which rules to use and where to write the generated data into. The constraint-problem is solved, and the solution is used to construct the records needed to populate the database.

The fabrication process is described next. The user has provided a data project which contains the structure of the data, the constraint rules and the fabrication configuration. In order to construct a constraint satisfaction problem for the solver, the platform analyzes the table metadata and gets table columns' data type and other properties, e.g., referential integrity constraints. The platform selects a sub-set of the relevant rules and tables using the fabrication configuration. In addition, relevant parent tables may be added due to referential integrity dependencies. Additional default rules are sometimes required as well (PK, Unique Column, string and binary column widths, etc.)

This information is used for the construction of a database table dependency graph. For each table in that graph, starting at root nodes, structural record dependencies are built recursively.

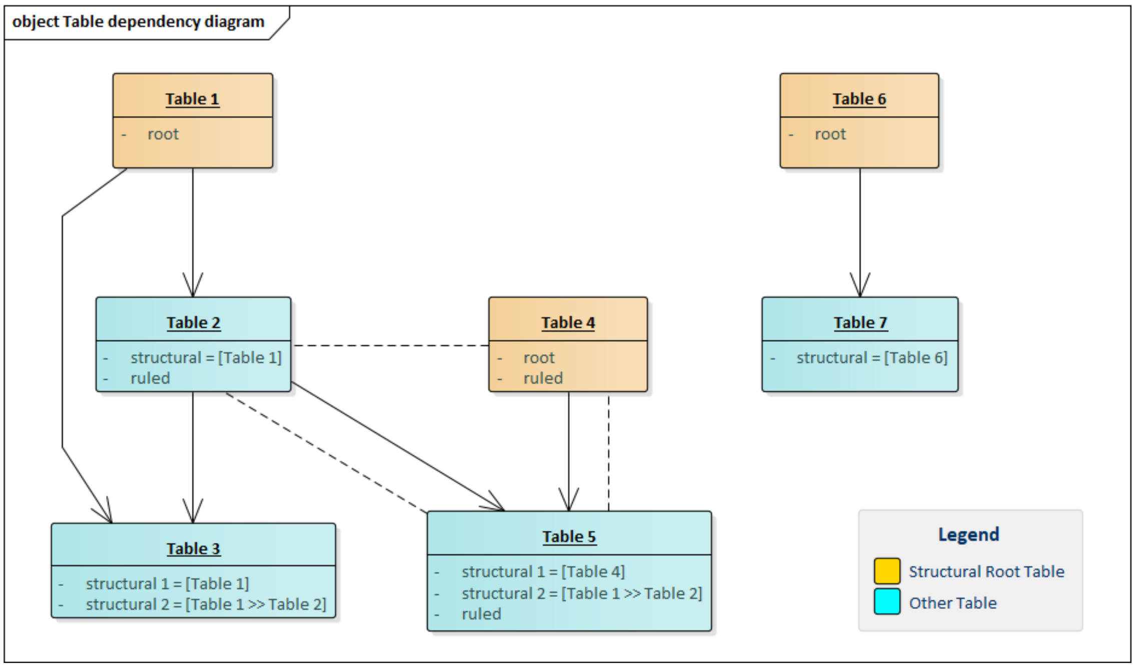

Figure 4: Table dependency diagram

Once the above graphs are built, the fabrication pattern is computed where each target table record is assigned to one of the following fabrication modes: 'New', 'Reuse' or 'Other'.

Given the patterns, the graph and the rules, a CSP problem can be created. The CSP has a language used to model a real-world problem. A problem consists of variables and rules.

The solution is an assignment of one value for each variable where all the rules are satisfied. The solver finds a random solution. Each time it solves a problem, a new solution is produced. The user does not specify how to solve the problem, but rather focuses on what to solve using the specified rules for a legal solution.

Each table is translated in a CSP variable struct or a CSP vector of variable structs, according to the record type, each table column is translated into a CSP variable. Each of the rules is added to the problem as a CSP constraint. A simple example of the CSP problem structure can be seen in the figure below. The CSP problem defined here is the famous eight queens puzzle [25][26].

```
variable integer arr[8];

constraint queens: forEach (q, 0, sizeOf(arr)-1, 0 <= arr[q] <= 7);
constraint sameRow: allDiff (q, 0, sizeOf(arr)-1, arr[q]);
constraint sameDiag: forEach (i, 0, sizeOf(arr)-1, \\
                        forEach (j, 0, sizeOf(arr)-1, ((i!=j) -> (abs(i-j) != abs(arr[i]-arr[j])) )) );
```

Figure 5: A CSP example

Finally, the problem is submitted to the solver. The solution is recursively parsed starting the iteration root following the topology of (vectored) record variables. In the case of database project, an SQL insert statement for all table records is created and that SQL insert statement is submitted to the DB for execution. If streaming option is enabled and properly configured, the solution is converted to the relevant format and sent to the messaging broker specified in the configuration. In the case of file projects, the solver result is converted into the relevant file format.

4 Modelling Automation

Modelling data via constraint rules is a difficult and time-consuming task. Therefore, the automation of this process is highly important. Modelling complex dependencies between the data requires deep understanding of the data and its relationships and the uses of that data. This means it would be very difficult to automate such complex cases. Fortunately, they are not very frequent.

Modelling in cases where several tables include tens or even hundreds of columns (or fields in file projects), becomes too difficult for manual work very quickly.

Modelling of previously learned common terms could be recorded and applied in future projects. On top of this, interesting data characteristics could be learned through data analytics applied to real data sets that could not be used due to privacy considerations or other reasons.

The platform will eventually be able to derive data rules which were automatically learned and extracted from the data analysis results.

The idea behind this is to fabricate data like the training data thus realistic. Data similarity in this context is defined as different data that has similar statistical properties both column wise and cross-column wise. Such properties include numerical, dates, words, distributions, means, variances, quantiles, confidence intervals, patterns, distinctness, uniqueness, cross-column correlations and more.

The platform modelling automation feature will use either basic inputs such as a column (or field in file projects) name, its data type and any database constraint rules, in order to find the best match for existing rules in a set of categorized, pre-defined rules. A more advanced option will use data analysis to automatically construct the data modelling rules to generate similar data. For that end, an extendible API for data analysis was developed, which enables the platform to accept various types of analytics results formats. Internally, the automatic rule creation component works using this API, to translate the results to constraint rules.

A simple example could be: given a table ‘PERSON’ and a column ‘AGE’, and its estimated distribution N(30,5), the module will add this rule:

```
normalDistributionNumber(PERSON.AGE,30,5);
```

Using data analysis may result in information leakage. An example where this may happen and how to mitigate this could be a database containing a column with credit card numbers. The data analysis will probably analyse it as uniformly distributed data where each value’s cardinality is one. Also, the number of unique values and the number of distinct values is going to be similar to one another but also similar to the total number of the sampled values. This use case could be a generic filtering criterion.

5 Concluding remarks

This document described the approach of the I-BiDaaS data fabrication platform tool and the modelling automation aspects. The data to be generated for the use case scenarios defined in WP1 and populated into the platform are defined in D2.1. These data specifications must be modelled in the platform using constraint rules. The modelling automation may already be able to partially assist with the modelling of the less complicated rules. Generated data can be sent to databases, files and streaming brokers. The automation component, provided analytics results, could be able to add meaningful derived constraint rules to improve the generated data quality.

6 References

[1] "Create high-quality test data while minimizing the risks of using sensitive production data." *IBM InfoSphere Optim Test Data Fabrication*, IBM, 2017, https://www.ibm.com/il-en/marketplace/infosphere-optim-test-data-fabrication.

[2] "Test Data Fabrication." *Security and Data Fabrication*, IBM Research, 2011, https://www.research.ibm.com/haifa/dept/vst/eqt_tdf.shtml.

[3] "Constraint Satisfaction." IBM Haifa Research, IBM, 2002, https://www.research.ibm.com/haifa/dept/vst/csp.shtml.

[4] Y. Richter, Y. Naveh, D. L. Gresh, and D. P. Connors (2007), "Optimatch: Applying Constraint Programming to Workforce Management of Highly-skilled Employees", International Journal of Services Operations and Informatics (IJSOI), Vol 3, No. 3/4, pp. 258 - 270.

[5] Y. Naveh, Y. Richter, Y. Altshuler, D. Gresh, and D. Connors (2007), "Workforce Optimization: Identification and Assignment of Professional Workers Using Constraint Programming", IBM J. R&D.

[6] Y. Naveh, M. Rimon, I. Jaeger, Y. Katz, M. Vinov, E. Marcus, and G. Shurek (2006), "Constraint-Based Random Stimuli Generation for Hardware Verification", AI magazine Vol 28 Number 3.

[7] E. Bin, R. Emek, G. Shurek, and A. Ziv (2002). "Using a constraint satisfaction formulation and solution techniques for random test program generation", IBM Systems Journal, 2002.

[8] Merav Aharoni, Odellia Boni, Ari Freund, Lidor Goren, Wesam Ibraheem, Tamir Segev (2015), "Rectangle Placement for VLSI Testing", CPAIOR 2015: 18-30

[9] O. Boni, F. Fournier, N. Mashkif, Y. Naveh, A. Sela, U. Shani, Z. Lando, A. Modai (2012) "Applying Constraint Programming to Incorporate Engineering Methodologies into the Design Process of Complex Systems" Proceedings of the Twenty-Fourth Conference on Innovative Applications of Artificial Intelligence, Toronto, Ontario, Canada. AAAI 2012.

[10] Y. Ben-Haim, A. Ivrii, O. Margalit and A. Matsliah (2012) "Perfect Hashing and CNF Encodings of Cardinality Constraints", SAT 2012, Trento, Italy.

[11] E. Bin, O. Biran, O. Boni, E. Hadad, E. K. Kolodner, Y. Moatti, D. H. Lorenz (2011), "Guaranteeing High Availability Goals for Virtual Machine Placement", ICDCS 2011.

[12] Jeonghee Shin, John A Darringer, Guojie Luo, Merav Aharoni, Alexey Y Lvov, G Nam, Michael B Healy (2011), "Floorplanning challenges in early chip planning", SOCC Conference, 2011 IEEE International, pp. 388—393

[13] Y. Naveh (2010). "The Big Deal, Applying Constraint Satisfaction Technologies Where it Makes the Difference". Proceedings of the Thirteenth International Conference on Theory and Applications of Satisfiability Testing (SAT'10).

[14] S. Asaf, H. Eran, Y. Richter, D. P Connors, D. L. Gresh, J. Ortega, M. J. Mcinnis (2010). "Applying Constraint Programming to Identification and Assignment of Service Professionals". Accepted for presentation in The 16th International Conference on Principles and Practice of Constraint Programming (CP2010). The paper received the Best Application Paper Award.

[15] B. Dubrov, H. Eran, A. Freund, E. F. Mark, S. Ramji, and T. A. Schell, (2009). "Pin Assignment Using Stochastic Local Search Constraint Programming" in Proceedings of the 15th International Conference on Priniciples and Practice of Constraint Programming (CP'09), Edited by Ian P. Gent, pp 35-49.

[16] Y. Richter, Y. Naveh, D. L. Gresh, and D. P. Connors (2007), "Optimatch: Applying Constraint Programming to Workforce Management of Highly-skilled Employees", IEEE/INFORMS International Conference on Service Operations and Logistics, and Informatics (SOLI), Philadelphia, pp. 173-178.

[17] S. Sabato and Y. Naveh (2007), "Preprocessing Expression-based Constraint Satisfaction Problems for Stochastic Local Search", Proceedings of The Fourth International Conference on Integration of AI and OR Techniques in Constraint Programming for Combinatorial Optimization Problems (CP-AI-OR).

[18] Y. Naveh, M. Rimon, I. Jaeger, Y. Katz, M. Vinov, E. Marcus, and G. Shurek (2006), "Constraint-Based Random Stimuli Generation for Hardware Verification", IAAI 2006.

[19] Y. Richter, A. Freund, and Y. Naveh (2006), "Generalizing AllDifferent: The SomeDifferent constraint", Proceedings of the 12 International Conference on Principles and Practice of Constraint Programming - CP 2006, Lecture Notes in Computer Science, Volume 4204, pages 468-483.

[20] Y. Naveh and R. Emek (2006). "Random stimuli generation for functional hardware verification as a CP application - a demo", IAAI 2006.

[21] Y. Naveh (2005). "Stochastic solver for constraint satisfaction problems with learning of high-level characteristics of the problem topography" CP 2005

[22] F. Geller and M. Veksler (2005), "Assumption-based pruning in conditional CSP", in van Beek, P., ed., CP, "Principles and Practice of Constraint Programming - CP 2005" of Lecture Notes in Computer Science (3709), 241-255 Springer.

[23] R. Dechter, K. Kask, E. Bin, and R. Emek (2002). "Generating random solutions for constraint satisfaction problems", AAAI 2002.

[24] D. Lewin, L. Fournier, M. Levinger, E. Roytman, G. Shurek (1995). "Constraint Satisfaction for Test Program Generation", Internat. Phoenix Conf. on Computers and Communications, March 1995.

[25] Hoffman, E. J., et al. “Constructions for the Solution of the m Queens Problem.” Mathematics Magazine, vol. 42, no. 2, 1969, pp. 66–72. JSTOR, JSTOR, www.jstor.org/stable/2689192.

[26] Watkins, John J. (2004). Across the Board: The Mathematics of Chess Problems. Princeton: Princeton University Press. ISBN 0-691-11503-6.