CoLInS 2022

“An Approach and a Software Tool for Automatic Source Code Generation driven by Business Rules”

Andrii Kopp

Ph.D. in Information Technology,
Associate Professor

kopp93@gmail.com

Dmytro Orlovskyi

Ph.D. in Information Technology,
Associate Professor

orlovskyi.dm@gmail.com

Department of Software Engineering and Management Intelligent Technology

Educational and Scientific Institute of Computer Science and Information Technology

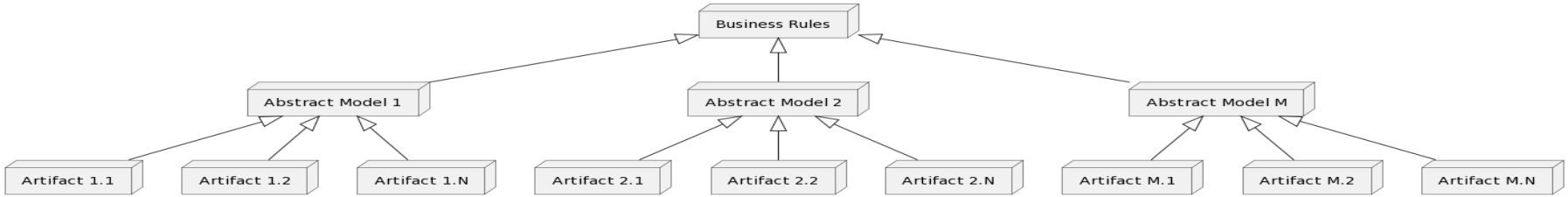
KHARKIV POLYTECHNIC INSTITUTE
1885
NTU

Motivation

- The time between specifications capturing and the product delivery is critical for the software development process and its stakeholders.
- The source code generation could significantly increase the software development process by shortening the time between requirements gathering and delivery using automatic programming and low-code solutions.
- The main idea of automatic programming is to “tell the computer what to do rather than how to do the task”.
- Hence, automatic programming should be supported by some definitive high-level language that is closer to a natural language than a programming language.
- This paper aims at automatic source code generation from natural language statements given as business rules to facilitate the software development process by bridging a gap between business analysis and engineering.

Problem Statement

- We attempt to use the SBVR (Semantic of Business Vocabulary and Rules) OMG (Object Management Group) standard to provide a unified solution that can be integrated with other software development utilities.
- We want to improve the previously proposed approach (based on the translation of fact business rules into SQL DDL (Data Definition Language) scripts) by using the Model-Driven Development (MDD) paradigm when business rules are given as subject domain descriptions to build an abstract model with multiple possible implementations.



Abstract models serving as sources to generate the source code or other artifacts

Translation of SBVR Business Rules into the Triplestore Model

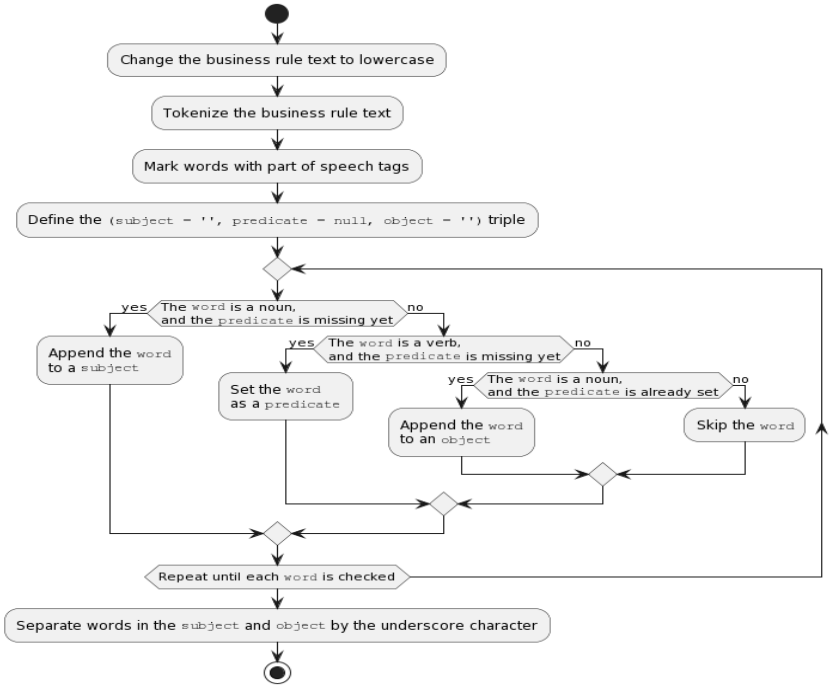
Having the business rules, we can obtain a set of triples:

$$T = \left\{ t_i = \langle s_i, p_i, o_i \rangle \middle| i = \overline{1,n} \right\},$$

where:

- t_i is the i-th triple, $i = \overline{1,n}$;
- s_i is the subject within the i-th triple t_i, $i = \overline{1,n}$;
- p_i is the predicate within the i-th triple t_i, $i = \overline{1,n}$;
- o_i is the object within the i-th triple t_i, $i = \overline{1,n}$;
- n is the number of business rules and corresponding triples.

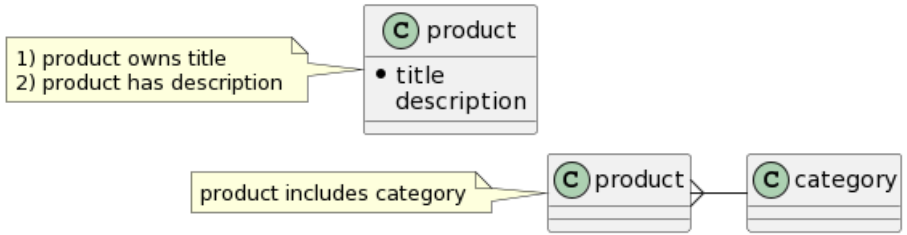
We can store triples, obtained using the input business rules, and retrieve triples to build object-oriented, entity-relationship, and other models.



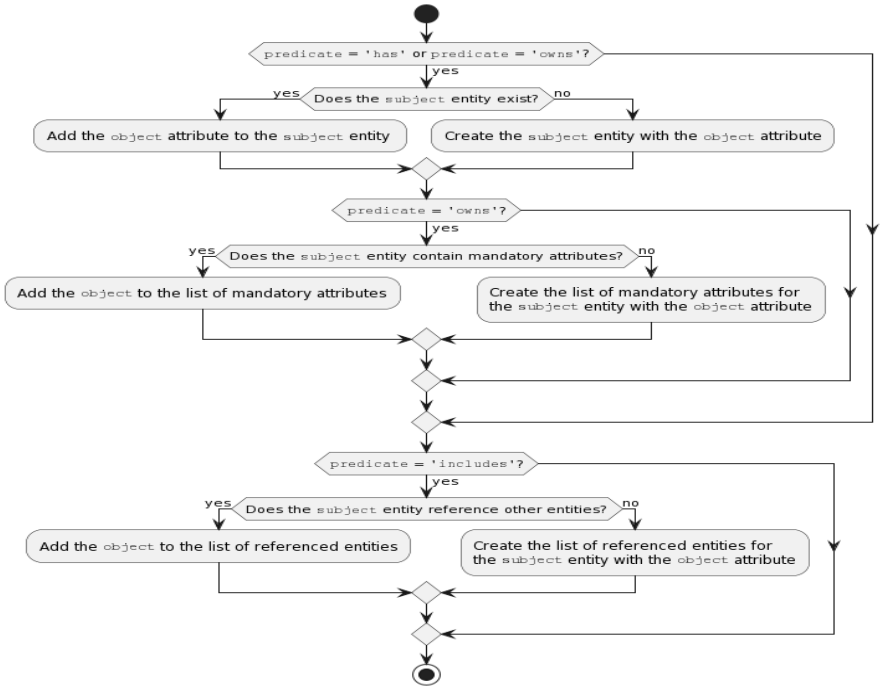
Translation of the Triplestore Model into the Data Model

To restrict the syntax of fact business rules, we propose to consider only several verbs according to their purpose:

- "has" means unnecessary attributes that can hold null or missing values;
- "owns" means mandatory attributes that cannot hold null or missing values;
- "includes" verb should be used to detect relationships between entities.

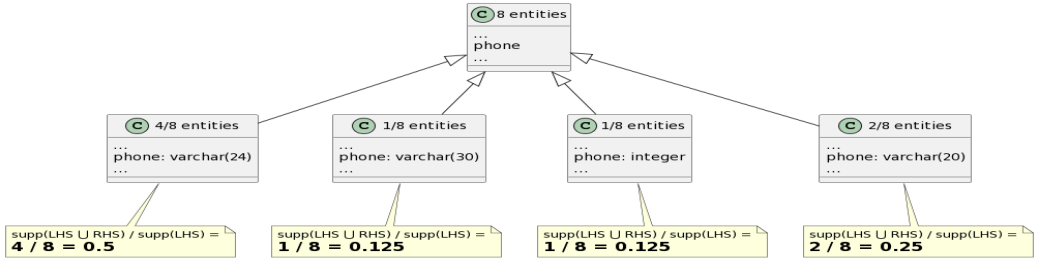


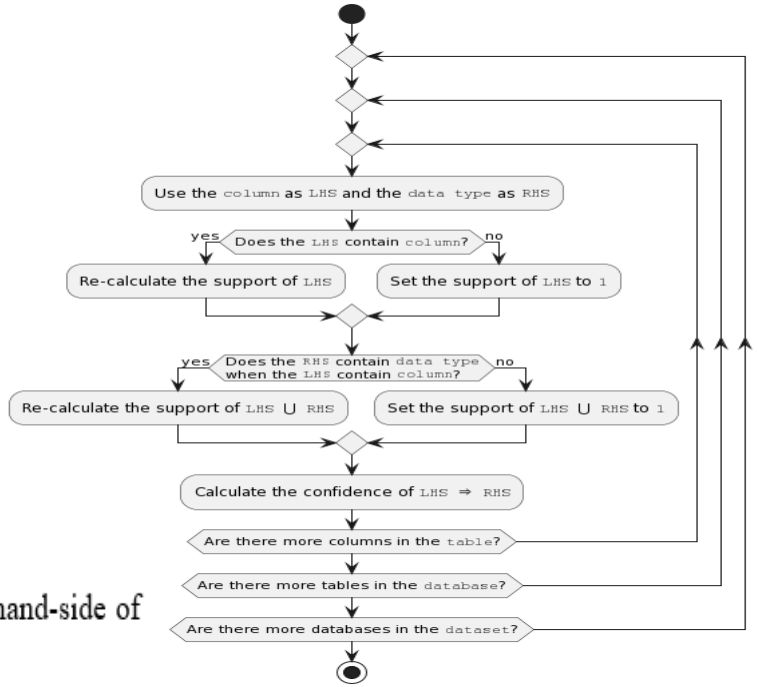
Having the data model elements discovered we can build different software implementations of these models (SQL DDL scripts, programming language classes, etc.)



Suggestion of Attribute Data Types based on Association Rules

We propose to use the "Spider" dataset maintained by Yale students. This dataset includes 200 databases with multiple tables covering 138 different domains.





$$\mathrm{conf}_k^v(LHS_k \Rightarrow RHS_k^v) = \frac{\mathrm{supp}(LHS_k \cup RHS_k^v)}{\mathrm{supp}(LHS_k)}, k = \overline{1,p}, v = \overline{1,w},$$

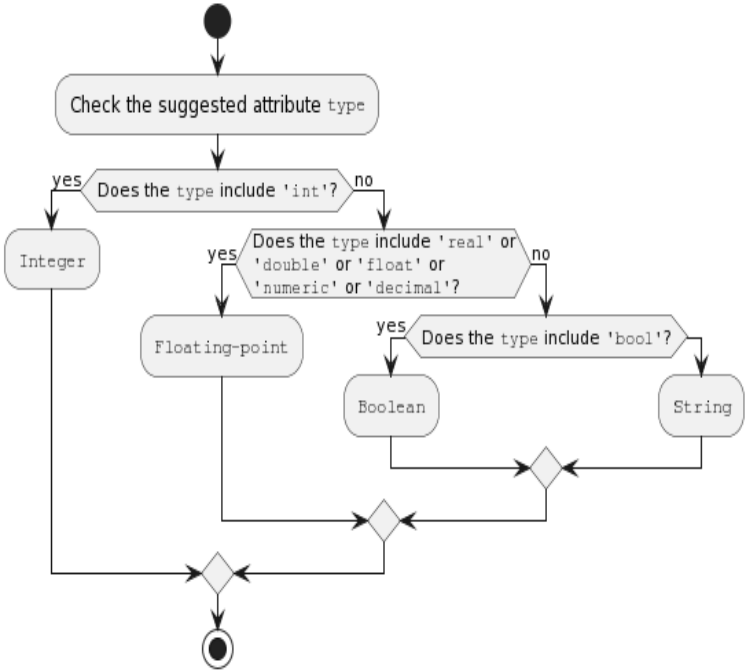
- LHS_k is the k-th attribute placed as the left-hand-side of the rule;
- RHS_k^v is the v-th data type associated with the k-th attribute placed as the right-hand-side of the rule;
- supp is the number of rules that contain a given set of items.

Adjustment of Suggested Attribute Data Types

Using the algorithm and the mapping between languages and generic data types, various software development components can be generated based on the data model: classes or structures, database scripts, smart contracts, or other source code that declare data structures.

Generic type Technology	Integer	Floating-point	Boolean	String
Java	int	double	boolean	String
C#	int	double	bool	string
SQL	int	real	smallint(1)	varchar(255)
Solidity	int	int	bool	string

Corresponding data types used in the most popular enterprise programming languages and SQL-based database management systems

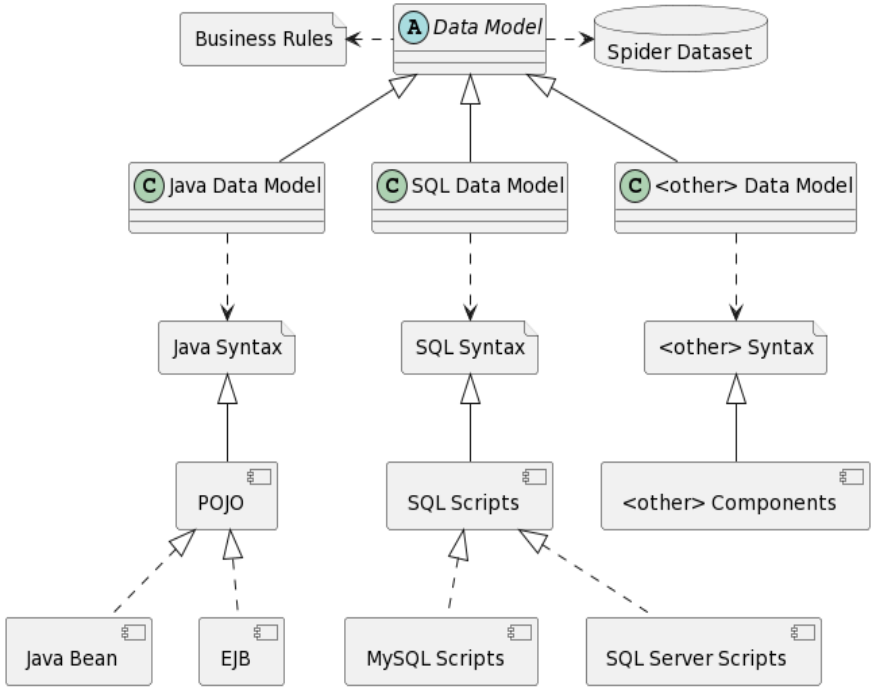


Implementation of Software Components based on the Data Model

The proposed approach assumes building the abstract data model from the business rules to define entities (or concepts), their attributes, and relationships among them.

Therefore, the data model based on the business rules and data type association rules can be used to automatically generate almost any software component for which are only necessary:

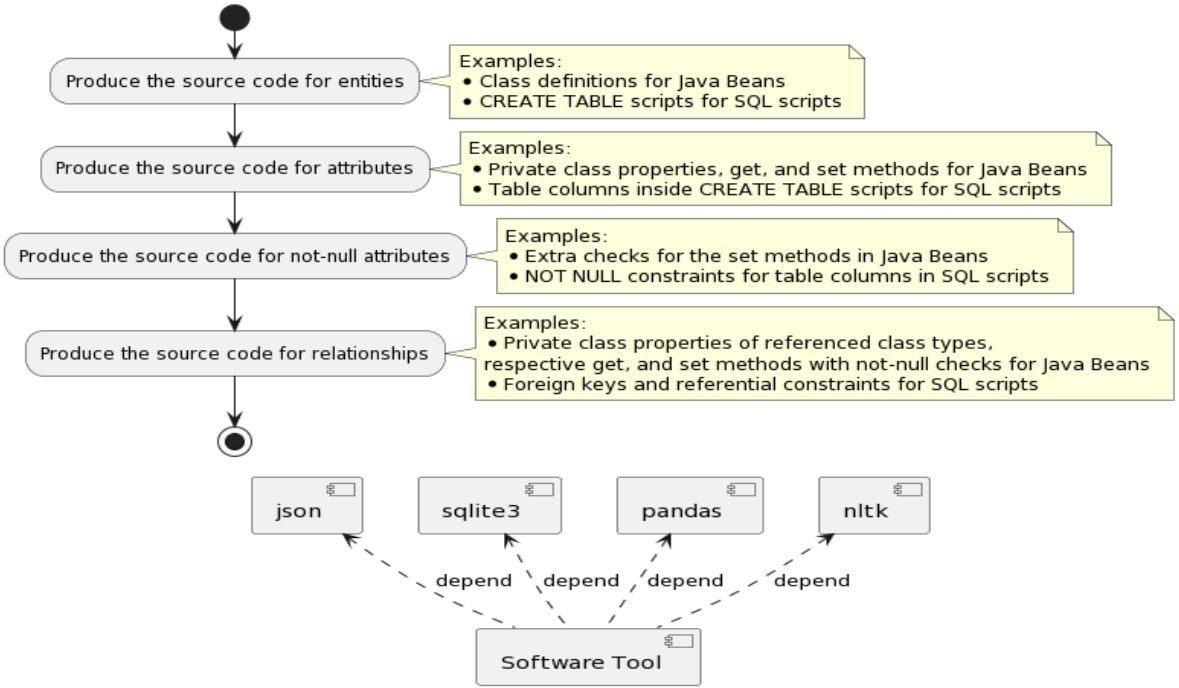
- rules on entity representation according to a given syntax;
- rules on attribute representation, including mandatory ones, according to a given syntax;
- rules on relationship representation according to a given syntax;
- rules on attribute data type representation according to a given syntax.



Software Implementation of the Proposed Approach

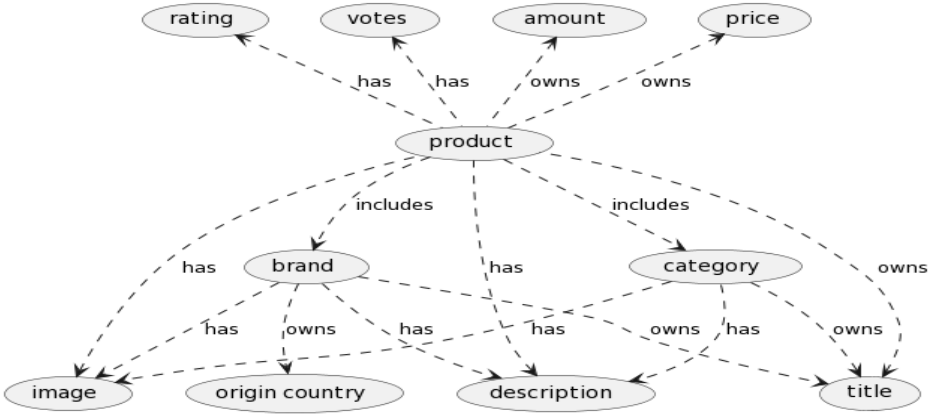
We have implemented the software solution using the Python programming language because of its relative simplicity, flexibility, and rich collection of packages, including the packages for natural language processing and database operations:

- **json** – the JSON (JavaScript Object Notation) format encoder and decoder;
- **sqlite3** – the API (Application Programming Interface) for SQLite databases;
- **pandas** – the open-source data analysis and manipulation tool;
- **nltk** – computational linguistics library known as the Natural Language Toolkit (NLTK).



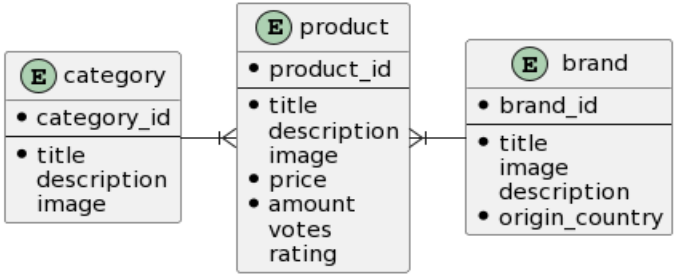
Source Code Generation from Business Rules (1)

product owns title
product includes brand
product has description
product has image
product owns price
product owns amount
product includes category
product has votes
product has rating
category owns title
category has description
category has image
brand owns title
brand has image
brand has description
brand owns origin country



The triplestore model based on business rules

The data model based on triplestore model

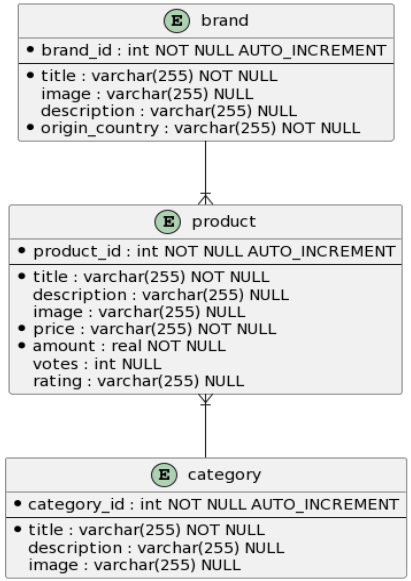


Source Code Generation from Business Rules (2)

The suggested data types based on association rules

Attribute	Data type	Confidence
title	String	0.60
description	String	0.25
image	String	–
price	String	0.29
amount	Float	0.33
votes	Integer	0.50
price	String	0.29
amount	Float	0.33
votes	Integer	0.50

The source code of Java and SQL software components generated from business rules



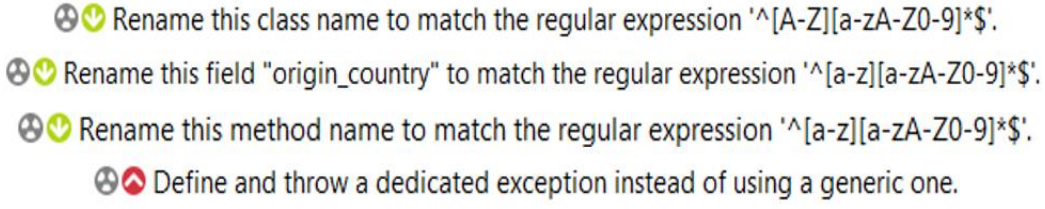
SQL

Java Beans

Verification and Validation of the Generated Source Code

The static analysis of Java Beans code using the SonarLint for Eclipse IDE shows four types of identified issues. These issues include invalid class names, field names, and method names, as well as usage of generic exceptions instead of dedicated ones. Execution of generated Java Beans demonstrates successfully compiled classes.

Execution of generated SQL database creation scripts demonstrates successfully created tables on the MySQL server. Data manipulation SQL statements demonstrate the integrity and consistency of created database tables.




```
MariaDB [product_test]> --- test foreign key constraints
MariaDB [product_test]> INSERT INTO product (title, description, image,
    -> price, amount, votes, rating, brand_id, category_id)
    -> VALUES ('Xiaomi Mi 11', 'Smartphones and accessories', 'mi11.png',
    -> 89.99, 999, 4321, 4.3, 0, 0);
ERROR 1452 (23000): Cannot add or update a child row: a foreign key constraint fails (`product_test`.
`product`, CONSTRAINT `product_ibfk_1` FOREIGN KEY (`brand_id`) REFERENCES `brand` (`brand_id`))
MariaDB [product_test]>
MariaDB [product_test]> --- test null columns
MariaDB [product_test]> INSERT INTO product (title, description, image,
    -> price, amount, votes, rating, brand_id, category_id)
    -> VALUES (NULL, NULL, NULL,
    -> NULL, NULL, NULL, NULL, NULL, NULL);
ERROR 1048 (23000): Column 'title' cannot be null
```

Contribution to Intelligent Source Code Generation Systems

- Despite the detected limitations (missing dedicated exceptions and naming violations in Java code) of the proposed approach and the software tools, the generated SQL database creation scripts and the Java Bean classes are valid and correspond to the given business rules.
- Generated artifacts can be used in an information system software development project after minor tuning – to customize data types and meet coding conventions.
- Therefore, this study encourages projects to move toward Intelligent Software Engineering practices that assume the usage of intelligent techniques in Software Engineering.
- The proposed approach and the software tool based on Natural Language Processing techniques assume the implementation of an intelligent source code generation system that is supposed to bridge the gap between software requirements (given as business rules) and the design of the information system's data layer.
- Furthermore, the elaborated intelligent system will result in an automatic source code generation environment that can augment traditional IDEs.

THANK YOU FOR ATTENTION!