A Proposal for Modelling TRIZ Functional Analysis

Tarek Stelzle

March 28, 2021

1 Aim of the work

The aim of this paper is to elaborate a proposal for an ontological modelling of the areas of *TRIZ Functional Analysis* based on the approaches in [3], [11], [12] and further own investigations. The work fits into the activities of the *WUMM Ontology Project* [7] to model core TRIZ concepts using modern semantic web means. The work consists of two parts – a *turtle file*, in which the semantic modelling is performed based on the SKOS framework [6], and *this elaboration*, in which the backgrounds and motivations of the concrete modelling decisions are detailed.

The paper is structured in the following way. In section 2 the information sources are mentioned and further explained. In the following section the conceptualisation will be shortly explained. In the next section 4 the Functional Analysis as described in [3] will be summarized. The next section introduces Pyhton tools which have been implemented to make the *turtle*-file creation easier. Following it will be shown how and why the *turtle*-file has been created. Extending the last semester will be some example implementations of the *turtle*-file for Functional Analysis in section 7. In the last section a conclusion about the paper wil be given.

2 Starting point

The starting point for the building the ontolog for Functional Analysis is going to be the book *Systematische Innovationsmethoden* [3]. The definitions, demenstrations and explanations will be used to implement the ontology.

Other definitions for Functional Analyis will be picked from [12]. As this is a great summary of many TRIZ related terms.

These two information sources and the following more detailed explained Webinar from Nikolai Shchedrin are the starting point in building the ontology for Functional Analysis.

2.1 Webinar of Nikolai Shchedrin

Furthemore Nikolai Shchedrin made a talk about building an ontology for *Function* and *Function Analysis* [11]. He presented some insights on how to structure the ontolgy.

2.1.1 Function Classification

First he introduced, that there should a be a new classification for the functions. There should be three types:

1. Function of the subsystem
2. Function of the upper system
3. Function of the surrounding objects

This will be implemented in the ontology.

2.1.2 Model Of A Function

Furthermore he introduced the *Model of a function*. With this a function is further described. Not only does it show the Action and the two components which interact, but it furthermore shows the parameter, the type of the function and the degree of execution.

The *Functional Model* is a graph representation of the system which is analyzed. Every node in the graph is a component or a subsystem of the system. The functions are represented by edges. This will be further explained in **??**.

The *Model of a function* can be helpful for building the *Functional Model*.

2.1.3 Objectives Of A System

As explaind from Nikolai Shchedrin the objective of a system can be divided into three objectives.

1. Primary Objective
2. Secondary Objective
3. Auxiliary Objective

The Primary Objective is the objective, which the system was build for.

Secondary Objects of the system are functionalities which are offered by the system, but for which it was not mainly built.

And the Auxiliary Objectives fulfill the purpose to get the Primary Objective to work.

2.1.4 Primary Function

Furthermore a function can be classified in one of these three functions.

1. Primary Function
2. Core Function
3. Auxiliary Function

The Primary Function is Primary Objective and its technical execution.

The Core Function represents a function, which directly helps executing the Primary Function.

The Auxiliary Functions describe the set of functions which help run other subsystems.

3 The Conceptualisations

The conceptualisations to be developed follow the basic assumptions and positings that are elaborated in more detail in [2]. In particular, the following namespace prefixes are used:

- `ex:` – the namespace of a special system to be modelled.
- `tc:` – the namespace of the TRIZ concepts (RDF subjects).
- `od:` – the namespace of WUMM's own concepts (RDF predicates, general concepts).

The central task is to model the functions and all belonging parameters which are use to describe these. In the examples it is shown that with the ontology it is going to be possible to build functions for the Functional Analysis, which can then be further investigated with one of the *Matrixes* and the *40 Principles*.

A function in a concrete example should then look like this.

```
ex:Steer
  a tc:function
  tc:Interaction ex:DriverSteeringwheel ;
  tc:Action ex:Turn ;
  tc:SubjectActionObject ex:DriverSteeringwheel ;
  tc:QualityOfFunction tc:UsefulFunction ;
  skos:prefLabel "Steer"@en, "Lenken"@de ;
  skos:Definition "Drehen des Lenkrads,
      durch den Fahrer um die Richtung des Fahrzeuges zu ändern."@de .
```

4 Functional Analysis

In this section a summary about the terms and definitions explained in [3] will be given. This should help explaining in the following section, why the *turtle*-file was modeled in that way.

4.1 Concept

In Functional Analysis the idea is to represent a system by various functions. This representation helps in organizing and structuring the system. To tackle the optimization problem a more precise look is taken at the non-usefula and contradictory functions in the system.

The two main objectives of Functional Analyis in TRIZ are formulated in the following.

1. Recognizing Of Problems
   Hereby it is objective to find as many as possible non-useful or contradictory functions in the system.
2. Trimming Of Components
   To optimize the system some components have to be redesigned. During this process the functionality of the component must not be changed.

With these two main objectives there are five tasks to handle in Functional Analysis.

1. Recognizing Interactions Between Objects
2. Recognizing Problems Within The System
3. Formulating Open Problems
4. Innovative Redesign
5. Optimizing Systems By Trimming
6. Bypassing Patents

4.2 Quality Of A Function

To further analyze the functions it is recommended to give each function a level of quality. This quality of a function makes it easier to find problems within the system.

Accorind to the book there are five different quality levels.

1. Useful Function:
   A function works as intended and the result is only positive.
2. Useful, But Insufficient Function
   A function has a positive impact but the result is not satisfieing.
3. Useful, But Bad Controllable Function
   A function with a positive impact but is not satisfieing as the result cannot be controlled. Hence it is wrongly timed.
4. Useful, But Excessive Function
   A function with a positive impact but a bigger result than necessary.

5. Harmful Function
   A function which has a negative impact.

More precise definitions can be found in the book [3] or in the Glossary of Souchkov [12]. As both of these are merged into the *turtle*-file the defintions can now also be found there.

Nikolai Shchedrin mentioned also a new function quality in his web-seminar. This is called *Useful Function With Disadvanteges*. As mentioned on the website [13] this class includes Redundant Functions, Insufficient Functions, Bad Controllable Functions and Missing Functions.

As there is no source mentining the Definition of a Redundant Function and a Missing Function, these will be interpreted in the following way.

1. Redundant Function:
   A function with a positive result, which is implemented a second time in the system.
2. Missing Function:
   A useful function which is not implemented in the system and therefore missing.

4.3 Functional Model

The Functional Model structure the function and the components in the system. In the book the following steps are recommended when building a Functional Model. It is also mentioned that steps two to four can be made in one step.

1. Making a list of components
2. Determine interactions
3. Linking functions to components (subjects)
4. Determining the direction of function (arrow)
5. Define the quality of the function

With this a graph-like structure is built. Every node in this graph represents a component or subsystem within the system. An edge represents a functions. This edges can have different styles and form representing the quality of the function.

For easier reading and analyzing the Functional Model it is possible to group components according to self-defined properties.

4.4 Positive And Negative Aspects

Using the Functional Analysis helps you to fully understand the system you are working with. Furthermore you can exchange knowledge between colleagues and clarify misunderstandings.

A Functional Analysis makes also sense, when no problem is known. This is good if you want to improve your system without knowing specific problems.

A negative attitude of the Functional Analysis, is that only known systems can be analyzed.

5 Implementation Of Tools

5.1 Creating Turtle File

For easier and faster creation of the "Matrix 2003" a small python tool was implemented. This tool reads the matrix from a *csv*-file and generates a *turtle*-file.

5.1.1 The CSV-File

First of alle *csv* file needs to be in the following syntax, because otherwise the matrix is wrongly interpreted. Every line in the csv file is a row in the matrix. Furthermore every comma seperates a column entry in a a row. The different entries in the matrix field are seperated by dashes.

For reading the matrix the tool "tabula" was used [9]. With this tool the Matrix2003 from the book "Systematische Innovation" was interpreted as a txt file. As this did not have the right syntax the small script, named *convertTxtMatrixToCsv.py*, was created. With this the matrix is converted into the right syntax, as mentioned above.

Using The Script Pyhton will need to be installed on the computer. The command to run the script will then be:

python convertTxtMatrixToCsv.py <csv-file-name>

The output *csv*-file will be called *createdMatrix.csv*. The file *temporary.txt* is only created for storing a the information temporary. It can be deleted afterwards.

5.1.2 Creating The Matrix

The following explains the implementation of the *createMatrix.py* script.

The script uses two other files. The first ist the *principles.txt*. In this file alle the forty principles are written down, with their belonging number at the beginnning. Hence the script can map an entry index to the belonging principle name.

The second file, which is also used is called *parameters.txt*. In there are all the forty-eight parameters which represent the column and rows in the matrix. These names are similar to those for the Altshuller Matrix. The nine new parameters, which were introduced for the Matrix2003 are the following.

1. QuantityOfInformation
2. FunctionEfficiency
3. Noise
4. HarmfulEmissions
5. CompatitbilityOrConnectability
6. Security
7. Vulnerability
8. AestheticsOrAppearance

9. ComplexityOfControl

The layout for the Turtle file is the same as the layout for the "Altshuller Matrix". As this makes it easier for later adjustments or improvements of the matrixes.

At first the script creates the header for the Matrix2003, which is the same as in the "Altshuller Matrix". Then it defines the owl entry for the Matrix. Afterwards the script iterates over the entries in the *csv* file and builds the matrix in the following way.

```
<http://opendiscovery.org/rdf/Matrix/E.06.34>
  od:decreasingParameter tc:EaseOfUse ;
  od:increasingParameter tc:SurfaceOfTheStationaryObject ;
  od:recommendedPrinciple tc:PreferredAction, tc:Asymmetry,
      tc:Mediator, tc:SelfService ;
  a od:MatrixEntry .
```

Figure 1: Entry of the Matrix2003 Turtle File

Using The Script For running the script python will need to be installed on the computer. The script can then be started with the following command:

python createMatrix.py <csv-file-name>

Afterwards there will be a file called *created_matrix.ttl*. This will be the result of the script.

5.2 Translating

For translating the various tags a python script was implemented, which takes a turtle files as input and adds the missing language tags. Therefore the script checks each line which already has a language tag. Then it makes an api call with the first tag as the source language. It add the language tags for english, german and russian. Each tag is a seperate api call. For translation the Google Translator is used.

Using the Script Python has to be installed on the computer. Furthermore does the script need the *deep_translator* library [10]. The library can be installed with the following command:

pip install deep-translator

Afterwards the script can be started. <*turtle-file-name*> specifies the file for which the missing language tags should be added.

python translateText.py <turtle-file-name>

This will output a file name *trasnlated_<turtle-file-name>.ttl*. This is the new turtle file with the added language tags. Furthemore all translated words or sentences will be shown in the terminal output.

6 Creating And Modeling the Turtle-File

7 Example For Functional Analysis

8 Conclusion

References

[1] Genrich Altshuller (1979). Creativity as an exact science (in Russian). English version: Gordon and Breach, New York 1988.

[2] Hans-Gert Gräbe (2021). About the WUMM modelling concepts of a TRIZ ontology. https://github.com/wumm-project/Leipzig-Seminar/blob/master/Wintersemester-2020/Seminararbeiten/Anmerkungen.pdf.

[3] Karl Koltze, Valeri Souchkov (2017). Systematische Innovationsmethoden (in German). Hanser, Munich. ISBN 978-3-446-45127-8.

[4] Alex Lyubomirsky, Simon Litvin, Sergei Ikovenko et al. (2018). Trends of Engineering System Evolution (TESE). TRIZ Consulting Group. ISBN 9783000598463.

[5] Nikolay Shpakovsky (2016). Tree of Technology Evolution. English translation of the Russian original (Forum, Moscow 2010). https://wumm-project.github.io/TTS.html

[6] SKOS – The Simple Knowledge Organization System. https://www.w3.org/TR/skos-reference/.

[7] The WUMM Project. https://wumm-project.github.io/

[8] Matrix2003: https://triz-journal.com/wp-content/uploads/2018/04/Screen-Shot-2018-04-30-at-15.20.25.png.

[9] tabulapdf: https://github.com/tabulapdf/tabula.

[10] Deep Translator: https://pypi.org/project/deep-translator/.

[11] Nikolai Shchedrin (2020). Webinare des TRIZ-Ontologie-Projekts "Funktion" und "Funktionsanalyse".

[12] Valeri Souchkov (2018). GLOSSARY OF TRIZ AND TRIZ-RELATED TERMS .

[13] Nikolai Shchedrin (2020). https://triz-summit.ru/onto_triz/mod/metod/triz/fa/model_fa/func_syst_model/func/func_type/disadv_f/.