About the WUMM modelling concepts of a TRIZ ontology

Hans-Gert Gräbe

March 15, 2021

1 Background

In 2019, a group of TRIZ specialists around A.G. Kuryan and M.S. Rubin launched the *TRIZ Developer Summit Ontology Project* (TOP), to achieve a review of the status quo and a more accurate ontological mapping of the TRIZ theory corpus. The work is a natural continuation of earlier efforts by other authors [9, 10] to outline a *TRIZ Body of Knowledge.* While the latter focused on a guide through the literature, TOP is concerned with the identification of essential concepts and essential relationships between these concepts using a modern semantic approach. The status of TOP was presented at the TRIZ Developer Summits in 2019 and 2020 and fixed in two publications [4, 5]. In a webinar series[1] first approaches of a detailed modelling of several sub-areas of TRIZ were presented. The project operates its own website `https://triz-summit.ru/Onto_TRIZ/` on which consolidated results are published.

The main results so far have been a mapping of the continents of the TRIZ world as a *Top Level Ontology* as well as a (still developing) division of that world into *Ontomaps* as specifically defined areas, which are to be modelled in more detail. Moreover, a *thesaurus* of about 500 terms as essential TRIZ concepts has been identified, which are to be defined more precisely. The glossary [8] by V. Souchkov in its version 1.2 serves as basis for this work. In the meantime a first list of 100 terms [12] has been published on the TOP website.

The main disadvantage of the TOP approach so far is the inconsistent use of semantic means. Such means are used in the background and in the internal processes of the TOP team, but even a clear namespace concept for URIs[2], the public availability of the results in an RDF store or at least as files in a relevant format – all this is missing, not to mention a SPARQL endpoint for querying the concepts.

However, such an infrastructure was developed and set up in the context of the WUMM project [15] and used for the representation of actors and activities of a *TRIZ Social Network* `https://wumm-project.github.io/TSN.html` (persons, conference reports, presentations, certificates). The data is publicly available in our github repo `RDFData` at [14] and forms the basis for a prototypical presentation platform [16] that uses simple semantic tools[3] to present

[1]See `https://wumm-project.github.io/OntologyWebinar` for links to the presentations and an English summary of the talks and discussions.

[2]URI – Unique Resource Identifier, one of the basic RDF concepts. This string is the *digital identity* of a concept and allows to add independently information about ≪the same thing≫ in a distributed environment.

[3]PHP and bootstrap using the EasyRdf PHP library – the code is publicly available in the github repo `web` at [14] for study and reuse in own platforms.

different facets of the data. Via a SPARQL endpoint [18] experts can make their own complex queries to the dataset.

This technical basis is the starting point for remodelling parts of the TOP outcome in the course of a *WUMM TRIZ Ontology Companion Project* (WOP) [19]. In addition to our own modelling (so far the TRIZ Principles, the TRIZ Inventive Standards and the TRIZ Business Standards), the *Top Level Ontology* and the division into *Ontomaps* are available in this format. The work on a *thesaurus* as well as the presentation of different approaches to a common glossary is actively accompanied. In the WOP approach the differing definitions of different TRIZ schools can coexist more clearly side by side than it is conceptually possible (and is probably not aimed at) in the TOP approach. This aspect, together with the focus on multilinguality, for which individual translation projects can easily be delimited based on the relevant RDF concepts, represent the essential additional contributions of the WOP approach.

The aim of this paper is to explain the basic modelling and semantic assumptions, concepts and settings of the WOP approach in more detail.

2 Basics of the WUMM Ontology Project

The WUMM ontology project accompanies the TOP activities in order

1. to carry out a remodelling according to semantic standards,
2. to enhance the material multilingually and
3. to build an LOD[4] infrastructure on this basis,

and thus to improve the basis for the necessary social coordination processes.

For this purpose, the SKOS ontology [6] is used with the concepts (K)

- `skos:Concept`, `skos:prefLabel`, `skos:altLabel` – concept naming
- `skos:definition`, `skos:example`, `skos:note` – concept properties
- `skos:narrower`, `skos:broader`, `skos:related` – concept relations.

SKOS provides an initial descriptive framework for conceptualisations. For the meaning of the individual concepts, we refer to [6] and the explanations below.

2.1 URIs and Namespaces

One of the central problems of transferring the existing data stocks on TRIZ concepts is the allocation of meaningful URIs, since the individual glossary entries in the existing TOP sources are identified solely by their labels. The OSA platform[5] is no exception to this since the URIs assigned there (both for the nodes and the edges of the constructed RDF graph) are not publicly visible.

[4]LOD is the abbreviation for *Linked Open Data*, a world of interlinked data and ≪worlds of concepts≫ steadily growing during the last 15 years. See `https://lod-cloud.net/`.

[5]The OSA platform is used as an TOP internal ontology editor, see `https://wumm-project.github.io/TOP` for more information about the platform, its odds and evens.

One of the first decisions that must be made for the allocation of URIs is the *definition of namespaces* that correspond to the different modelling contexts. Since an *ontology modelling* basically has the purpose of being applied in *modellings of real-world systems*, at least these two modelling contexts have to be distinguished. The modelling context of a (prototypical) real-world system will usually only play a role in examples in which the effect of ontology modelling decisions is practically demonstrated. At the level of ontology modelling, we further distinguish between the parts of the concepts that are largely uncontroversial[6] and the parts of the concept for which special conceptual approaches have been developed within the WUMM Ontology Project (WOP). For these different abstraction layers we use the following namespaces:

- `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).

During the computer based transformation of the datasets into a valid RDF format, a first suggestion for URIs in the namespace `tc:` was automatically generated and then further consolidated in several steps. An essential task still to be done is to finish this final consolidation of URIs, i.e. to merge URIs generated from different sources that refer to the same concept.

2.2 Provenance of Explanations

Another problem of this ontological modelling is the representation of the provenance of the individual explanations. For this purpose the SKOS concepts listed under (K) were replaced for each individual source by notations from the namespace `od:` in order to address the ≪worlds≫ of the individual TRIZ schools separately. The same applies to the use of provenance-dependent subclasses of `skos:Concept`.

Such notational variations are for example

- `skos:Concept` → `od:GSAThesaurusEntry`, `od:VDIGlossaryEntry` ...
- `skos:definition` → `od:SouchkovDefinition`, `od:VDIGlossaryDefinition` ...
- `skos:example` → `od:VDIGlossaryExample` ...

etc. Here `GSAThesaurus` stands for the thesaurus published on the Altshuller website [1], `VDIGlossary` for the VDI glossary [13] and `SouchkovDefinition` for the glossary [8] by V. Souchkov. All these data were available or provided to the WUMM project in a machine-readable format, transformed by us into suitable RDF formats and are available as open source both as files in the github repo `RDFData` at [14] and in our RDF Store [17]. See the RDF data itself, which can also be queried via our SPARQL endpoint [18].

This can be used to build a *combined glossary* where definitions from different TRIZ schools of the same concept co-exist. This is implemented prototypically[7] im such a way, that for each concept represented by a URI, a link displays all RDF triples in which this concept occurs as a subject or object. Further links in this representation can be used to navigate in the entire RDF graph (more precisely: in its respective connected component).

[6]These are mainly the *concepts* to be included in a glossary. We assign URIs of a `skos:Concept` to them and model their names as `skos:prefLabel`.

[7]Seee `http://wumm.uni-leipzig.de/ontology.php`.

3 Modelling a TRIZ ontology

3.1 SKOS basics

The SKOS ontology allows to express concepts and their relations in a lightweight way. The class `skos:Concept` and the predicates `skos:narrower`, `skos:broader` and `skos:related` are used for this purpose. The first two predicates describe hierarchical relationships between concepts[8], the third is used for non-hierarchical relationships.

Relationships between concepts can be of very different structure. Hierarchical relationships, for example, can model (transitive) subconcept relationships in taxonomies as well as whole-part relationships, which are inherently non-transitive when concepts of different qualities are related. Both types of conceptualisation have an intentional as well as an extensional aspect – the new units of meaning, especially their emergent properties, can neither be adequately described by mere enumeration of their subconcepts nor by the ≪legitimate interpretation of sense≫ of the purposes of their constitution in the sense of [2]. In the SKOS primer [7] these modelling aspects are described in more detail, especially the modelling of class-instance and whole-part relationships. We follow the recommendation in [7, Sect 4.7] and introduce subpredicates of the generic SKOS predicates listed above for different modelling contexts. More detailed modelling rules for such contexts are described and discussed below.

Since this project is about modelling a unified space of TRIZ concepts, further SKOS aggregation concepts such as `skos:ConceptScheme`, `skos:Collection`, `skos:OrderedCollection` etc. are not used. The aggregation of different concepts in collections (assignment to TRIZ generations [5, Table 1] or in concept classes `Basic`, `Model`, `Rule` and `TermGroup` [5, Fig. 4]) is realised via special predicates.

3.2 TRIZ ontology basics

All TRIZ concepts revolve around the central notion of a *system*, its planning, creation, operation, maintenance, further development, etc. Following the widely accepted understanding of that concept in the TRIZ community, a system is essentially a collection of components that interact in a specific way to produce the characteristic functionalities of the system. The subsystems referred to as *components* provide own functions for this, but the functionalities of the system do not result from a simple addition of the component functions, but as an emergent system property from their interaction. For the modelling of systems, their structural organisation (the ≪machine≫ in the sense of [11]) and their workflow organisation (≪how the machine works≫, ibid.) are equally important.

The systemic approach is thus self-similar and fractal; the terms ≪system≫ and ≪component≫ are largely used synonymously depending on the respective *modelling focus.* From a modelling perspective, *components* are regarded as black boxes whose functioning in the description dimension is given by their *specification*, and in the execution dimension by their *guaranteed specification compliant operation*, provided that the operating conditions (in particular the throughput of material, energy and information required for its operation) are

[8]From the SKOS Primer [7]: ≪The subject of a `skos:broader` statement is the more specific concept involved in the assertion and its object is the more generic one≫, i.e. `A skos:broader B` expresses that A is a subconcept of B. `skos:narrower` is the inverse property to `skos:broader`.

ensured within the infrastructure (environment). The components thus constitute a *world of technical systems* in the sense of the explanations in [3], to which we refer for further details of this conceptualisation.

In this understanding, *systems* are components that are considered as white box under the *given modelling focus* in order to project, create, improve or change their implementation. The latter is summarised under the aspect of *system transformation* in order to transform a ≪system as it is≫ into a ≪system as required≫. The well-known hill schema implies that in this procedure ultimately four versions of the description of the system are to be considered in parallel, see below. If one further understands the development of a system as a result of such transformations, then a fifth version of the system description

5. The system as it was realized

must be taken into account in order to adequately describe the contradiction between plan and reality and thus to understand system development as an iterative *transformation of transformations.*

Such readjustment allows to model iterative system transformations, in which at the beginning of every iteration the difference between the ≪system as required in the previous step≫ and the ≪system as it was realized≫, can be analysed and based on this analysis also the ≪model of the system as required≫ can be transformed.

3.3 Abstraction levels of modelling

An ontology is about ≪modelling of models≫, because the clarification of terms and concepts aimed at with an ontology is intended to be practically used in real-world modelling contexts. This ≪modelling of models≫ references a typical engineering context, in which the *modelling* of systems plays a central role and serves as basis of further planned action (including project planning, implementation, operation, maintenance, further development of the system).

In this process, *several levels of abstraction* are to be distinguished.

0. The level of the *real world system* to which the engineering task refers. This level is only *practically* accessible. The model to be developed at level 1 must be appropriate to cover all problems arising in the process of development and use of the system and express the inherent contradictoriness of the system.
   This contradictory nature of the system can be formulated only in language form, i.e. on the model level and *applying* the concepts available there. These concepts must therefore not only be able to describe the system itself, but also cover a description of the necessary aspects of its operation.
1. The *level of modelling* the system. The ontology provides the language means, concepts (RDF subjects) and properties (RDF predicates), which are to be *applied* at this level. This level is also the *level of methodological practice.*
2. The *level of the meta-model* as the actual ontology level on which the systemic concepts are *defined.* This definition is processed *applying* the methodological concepts whose linguistic means are made available on meta-level 2.
3. The *modelling meta-level 2* at which the methodological concepts are defined.

3.4 The TOP concept of a system

A central concept in TOP modelling is the distinction between the stages of

(1) the system as it is,
(2) the TRIZ model of the system as it is,
(3) the TRIZ model of the system as required, and
(4) the system as required.

The TOP glossary [12] explains the differences as follows

(1) The *system as is* is a system in its original state before it is analysed and transformed into a new ≪system as is≫.
(2) The *TRIZ model of the system as is* is formed from the ≪system as is≫ by means of various TRIZ models: component-structural and functional models, su-field or ele-field models, description of contradictions or of typical conflict schemes, etc. Depending on the chosen model type, the model will be transformed into the ≪TRIZ model of the system as required≫.
(3) The *TRIZ model of the system as required* is formed from the ≪TRIZ model of the system as is≫ by procedures which correspond to the selected model transformation method (functional, su-field, ele-field, solution of the contradictions in requirements and properties, etc.). The transition is performed along the line

   $$\begin{gathered}\ll\text{System as is}\gg \;\to\; \ll\text{TRIZ model of the system as is}\gg \\ \to\; \ll\text{TRIZ model of the system as required}\gg \;\to\; \ll\text{System as required}\gg\end{gathered}$$

   in accordance with the scheme of a TRIZ Model.
(4) The *system as required* is a system derived from the ≪system as is≫ through a transformation, based on the ≪model of the system as required≫.

The referenced notions of *TRIZ Model* and *System* ar explained as follows:

- A *system* is a set of elements in relationship and connection with each other, which forms a certain integrity, unity. The need to use the term ≪system≫ arises when it is necessary to emphasize that something is large, complex, not fully immediately understandable, yet whole, unified. In contrast to the notions of ≪set≫ and ≪aggregate≫, the concept of a system emphasises order, integrity, regularities of construction, functioning and development. The notion of system is part of the system and functional approach, and is used in the system operator.
- A *TRIZ model* is a schematic notation of gradual transition from the problem to TRIZ model of the problem, then to TRIZ model of the solution and then to the solution itself; or from the system to TRIZ model of the system, then to TRIZ model of the new system and then to actual change of the system (≪system as required≫). The TRIZ model includes the basic components of inventive thinking: analysis, synthesis, evaluation.

This distinction of modelling stages, which is important for the planning and subsequent execution of the transformation of the system, is applied to all sub-concepts in the TOP modelling approach.

As already explained above, the difference between the ≪system as it is≫ (1) and the ≪TRIZ model of the system as it is≫ (2) cannot refer to the real-world system itself, since it is accessible to (verbal) transformations only via its description form and thus its modelling. At this point, a reference to the hill model is more likely to be assumed, in which, in the sense of [11], where one has to progress from the problem description and model (1) – rather than from the ≪system≫ – to the TRIZ task model (2), further to the TRIZ solution model (3) and finally to a description of the solution to be implemented in reality (4). (1) and (4) refer to the variety of domain-specific modellings and ontologies, which have little to do with TRIZ and a TRIZ ontology beyond the methodological dimension. On the contrary, in the context of the transition (1) → (2), domain-specific approaches must be *translated* into suitable TRIZ concepts. The same applies to the retransition (3) → (4). Hence content-related TRIZ concepts concentrate on phases (2) and (3) and methodological TRIZ concepts on the transitions (1) → (2), (2) → (3) and (3) → (4). Hence the transition (1) → (2) itself contains two steps – a readjustment (1) → (1′) of the domain-specific model to prepare it for application of TRIZ concepts and the application (1′) → (2) of these concepts, as it is separated, e.g. in the TRIZ application concept of the TRIZ Trainer [11].

3.5 Modelling Systems and TRIZ Models

Since system models in (1) and (4) come with additional modelling information based on a great variety of domain-specific ontologies the TRIZ ontology is an add-on only and (domain-specific) ontology integration engineering approaches are required anyway. It is thus justified to concentrate solely on the concerns of modelling the TRIZ-relevant aspects in all four modes.

In the TOP approach, the distinction of these modes is consistently introduced for all system-relevant concepts at the level of subconcepts. Since this distinction only becomes relevant at the level of modelling concrete systems, alone the verbal means for distinguishing these purposes must be provided at the ontology level. The WOP approach takes a different route here and models this connection not through subconcepts, but as a predicate `od:belongsTo` with value from the `tc:StageValue` range

```
tc:System, tc:Model, tc:SystemAsIs, tc:SystemAsRequired,
tc:ModelAsIs, tc:ModelAsRequired
```

to assign to a concept *instance* of a concrete modelling one of the modelling stages as discussed above. This also makes it easy to extend the `rdfs:range` of this property if, for example, further system versions are to be included in the modelling of a special system in the context of an application of the system operator.

This significantly reduces the number of concepts to be distinguished, which is also indicated by reasons of homogeneity, since the different stages of this system model must be structurally similar in order to be able to capture structural continuities within its development, and thus must be modelled by a single uniform concept.

For this purpose, the WOP approach distinguishes (at the moment) the concept categories `od:Component` for describing the *structural* organisation of a system and its parts, and `od:PropertyDomain` for describing the structure of *property domains* of a system and its parts. Both are used as values of the predicate `od:WOPCategory`, following a modelling approach of categories of V. Souchkov in his glossary.

4 Typical modelling situations

4.1 Class-instance relation

In OO programming classes are usually extensionally conceptualised as a concept of member functions and attributes that are common to all instances of that class. We do not consider here the special possibility to define also static attributes and functions for a class. In this sense the class concept generalized the concept of its members. A special way to conceptualize classes with a finite number of instances are *morphological tables*.

Within the WOP approach this relation is modelled using the predicates `od:allowedValues` (a subproperty of `skos:narrower`) and `od:valueOf` (a subproperty of `skos:broader`). The class concept belongs to the WOP category `od:PropertyDomain`. No distinction is made between attribute (left column of a morphological table) and the attribute value range (set of values in the right column of a morphological table).

Example: Colour (red, green, yellow, blue).

```
ex:MeiersCar a ex:Car; od:hasColour tc:green .

od:hasColour a rdfs:Property;
  rdfs:domain ex:Car;
  rdfs:range tc:Colour .

tc:Colour a skos:Concept, od:AdditionalConcept ;
  skos:prefLabel "Colour"@en, "Farbe"@de ;
  od:WOPCategory od:PropertyDomain ;
  od:allowedValues tc:red, tc:green, tc:yellow, tc:blue .

tc:green a skos:Concept, od:AdditionalConcept ;
  od:valueOf tc:Colour ;
  skos:prefLabel "green"@en, "grün"@de .

...
```

4.2 Class hierarchy relation

In the WOP approach class hierarchies are modelled with the predicate `od:hasSubConcept` that refines `skos:narrower` and `od:subConceptOf` as its inverse predicate.

Example: A flow has several static components (source, channel, receiver, control unit). A flow itself is a component of a system and hence belongs to the WOP category `od:Component` as its subcomponents do.

```
ex:MeiersFlow od:hasStaticFlowComponent ex:SpecialPumpX32 .
ex:SpecialPumpX32 a tc:Pump .

od:hasStaticFlowComponent a rdfs:Property;
  rdfs:domain tc:Flow;
  rdfs:range tc:StaticFlowComponent .

tc:StaticFlowComponent
  od:hasSubConcept tc:ControlUnit, tc:Receiver, tc:Source, tc:Channel ;
  a skos:Concept, od:AdditionalConcept ;
  od:WOPCategory od:Component ;
  skos:prefLabel "static components of the flow"@en,
    "statische Flusskomponenten"@de .
tc:ControlUnit
  od:subConceptOf tc:StaticFlowComponent ;
  od:hasSubConcept tc:Pump, tc:Valve ;
  od:WOPCategory od:Component ;
  skos:prefLabel "Control Unit"@en, "Steuerungssystem"@de ;
  skos:altLabel "Management System"@en, "Managementsystem"@de .
tc:Pump
  od:subConceptOf tc:ControlUnit ;
  od:WOPCategory od:Component ;
  skos:prefLabel "pump"@en, "Pumpe"@de ;
  a skos:Concept, od:AdditionalConcept .
```

References

[1] Altshuller Web Site. Basic TRIZ terms.
`https://www.altshuller.ru/thesaur/thesaur.asp`

[2] Peter L. Berger,Thomas Luckmann. The Social Construction of Reality: A Treatise in the Sociology of Knowledge. Anchor Books, 1966. ISBN 978-0-385-05898-8.

[3] Hans-Gert Gräbe. Human and their technical systems. In Proceedings of the TRIZ Future Conference 2020, p. 399-410. An enlarged German version is available as `http://dx.doi.org/10.14625/graebe_20200519`.

[4] Andrej Kuryan, Valeri Souchkov, Dmitri Kucharavy. Towards ontology of TRIZ. Proceedings of TRIZ Developers Summit 2019 Conference, Minsk, 2019.
`https://wumm-project.github.io/Ontology.html`

[5] Andrej Kuryan, Michail Rubin, Nikolaj Shchedrin, Olga Eckardt, Natalja Rubina. TRIZ Ontology. Current State and Perspectives. TDS 2020 (in Russian).
`https://wumm-project.github.io/Ontology.html`

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

[7] SKOS Simple Knowledge Organization System Primer.
`https://www.w3.org/TR/2009/NOTE-skos-primer-20090818/`.

[8] Valeri Souchkov. Glossary of TRIZ and TRIZ-related terms, version 1.2. The International TRIZ Association, MATRIZ 2018.
`https:`
`//matriz.org/wp-content/uploads/2016/11/TRIZGlossaryVersion1_2.pdf`

[9] Simon Litvin, Vladimir Petrov, Michail Rubin (2007). TRIZ Body of Knowledge.
`https://triz-summit.ru/en/203941`.

[10] Simon Litvin, Vladimir Petrov, Michail Rubin, Victor Fey (2012). TRIZ Body of Knowledge.
`https://matriz.org/wp-content/uploads/2012/07/TRIZ-Body-of-Knowledge-final.pdf`

[11] Nikolai Shpakovski et al. The TRIZ Trainer. `https://triztrainer.ru`, `https://triz-trainer.com`,

[12] TRIZ 100 Glossary. A short glossary of key TRIZ concepts and terms (in Russian). `https://triz-summit.ru/onto_triz/100/`.

[13] VDI-Norm 4521 Blatt 1. Erfinderisches Problemlösen mit TRIZ – Grundlagen und Begriffe (Inventive problem solving with TRIZ – Basics and terminology). April 2016.

[14] The WUMM github repositories. `https://github.com/wumm-project`.

[15] The github pages of the WUMM Project. `https://wumm-project.github.io/`.

[16] The web demonstration pages of the WUMM Project. `http://wumm.uni-leipzig.de`.

[17] The RDF store of the WUMM Project. `http://wumm.uni-leipzig.de/rdf`.

[18] The SPARQL endpoint of the WUMM Project. `http://wumm.uni-leipzig.de:8891/sparql`.

[19] The WUMM TOP Companion Project. `https://wumm-project.github.io/Ontology.html`