Modelling TRIZ Flow Analysis with RDF

Immanuel Thoke

May 7, 2021

Contents

1 Abbreviations

FAM	–	Flow Analysis Model
HTTP	–	Hypertext Transfer Protocol
ICE	–	Internal combustion engine
LETS	–	Laws of Evolution of Technical Systems
MINT	–	German acronym for Mathematik, Informatik, Naturwissenschaft und Technik
RDF	–	Resource Description Framework
SFM	–	Substance field model
STEM	–	acronym for science, technology, engineering and mathematics
TOP	–	TRIZ Ontology Project
TRIZ	–	Russian acronym for теория решения изобретательских задач, see TIPS
TIPS	–	Theory of Inventive Problem Solving
WUMM	–	Widersprüche Und Management-Methoden
ZRTS	–	Russian acronym for Комплекс законов и тенденций развития технических систем, see LETS

2 Aim of the work

The aim of this paper is to elaborate a proposal for an ontological modelling of the areas of *TRIZ Flows* and *TRIZ Flow Analysis* based on the materials of the TRIZ Ontology Project (TOP) of the TRIZ Developer Summit [22] and further own investigations. The work fits into the activities of the *WUMM Project* [25] to accompany the TOP project and to transfer it into the language of modern semantic concepts [26]. The work therefore consists of two parts – a *turtle file*, in which the semantic modelling is performed based on the SKOS framework [20], and *this elaboration*, in which the backgrounds and motivations of the concrete modelling decisions are detailed.

3 Starting point

Flows and flow analysis play a rather peripheral role in TRIZ. In the standard reference [17] on a *TRIZ Body of Knowledge* it is listed as item 2.5.8, but in the 7 references to the literature listed there are no systematic explanations about the role of flows in TRIZ theory, but only individual examples in which concrete flows (such as magnetic flow) played a role in concrete modelling.

The most comprehensive source on issues of TRIZ modelling of flows and methodological issues of flow analysis is the thesis [14] by Yuri Lebedyev, which he submitted in 2015 under the supervision of S.A. Logvinov for graduation as TRIZ Master.

Such a limitation is justified, since the aim of this seminar paper is not to completely cover the TRIZ theory of flow analysis, but only to identify its essential concepts and describe their connection with modern semantic RDF means based on the SKOS ontology. The basis for this is above all the presentation of Olga Eckardt in the webinar of the TOP project in October 2020 [1], in which basic approaches of Lebedyev's work have obviously been incorporated.

We start from the definition of both concepts in the TOP Glossary [23]:

> **Flow** is the directional movement in space of particles of mass of matter, as well as the directional movement of energy or information. Flow has dual properties: the properties of a substance of which the flow consists, and the properties of a field, which is formed as a result of the directional movement of particles of matter. Flows can be useful, harmful and parasitic. The flow model contains static components: source, channel, receiver, control system. Flow is a special case of a process in which directional movement occurs in physical space.
>
> **Flow analysis** is a method of systems analysis to establish relationships in a system to flow, find resources, and determine compliance with existing system requirements. The result of a flow analysis is a model of the flow as is and a model of the flow to be. Flow modification techniques are used to develop systems with flows and to solve inventive problems in them.

The use of these concepts is further included there in a single place in the context of cause-effect analysis (ibid.):

> Cause-effect analysis is performed:
>
> - When the causes of an undesirable effect are not clear (when we cannot go from an administrative contradiction to a technical one $AC \rightarrow TC$.
> - When it is necessary to clarify the causes of an undesirable effect (to deepen the understanding of the causes of an undesirable effect, e.g. after a functional or a flow analysis).

Other sources are much more sparing in their statements on these two individual terms, but list other flow-related concepts, for example in [19]:

- Delay Zone – A location in a flow in which the integral flow speed is significantly lower than local flow speed. A Delay Zone is a typical disadvantage identified by Flow Analysis.
- Flow – A sequence of events that have the same common feature.
- Flow Analysis – An analytical method and a tool which identifies disadvantages in flows of energy, substances, and information in a technical system.
- Flow Disadvantage – A disadvantage of a technical system being analysed identified during Flow Analysis. Examples: Bottlenecks, ≪Gray Zones≫, ≪Stagnation Zones≫, etc.
- Flow Distribution Analysis – A part of Flow Analysis that identifies distribution of flows and their disadvantages.
- Stagnation Zone – A part of a flow in which the flow stops temporarily or permanently. A Stagnation Zone is a typical disadvantage identified by Flow Analysis.
- Transmission – One of the key components (subsystems) of a Complete Technical System which according to the Law of System Completeness of a technical system transmits a flow of energy required to operate a working unit from an engine to the working unit.

4 The Conceptualisations

Being more than just a static methodology, there have been several approaches to standardise the syntaxes and semantics of the conceptual and methodological knowledge of TRIZ. Launching the TRIZ Ontology Project (TOP) several main contributors of the MATRIZ school are pushing forward this longterm goal from a very core institution of the TRIZ history. In awareness of the importance to establish uniform language models within human-machine-interface environments the WUMM [25] TOP Companion Project [26] is pushing even further "to transform that information [...]", which has been provided by the TOP, "[...] into RDF as the machine readable standard of the Web 3.0 (also called 'Semantic Web')." [26].

"RDF is a standard model for data interchange on the Web. [...] [It] extends the linking structure of the Web to use URIs to name the relationship between things as well as the two ends of the link (this is usually referred to as a 'triple'). Using this simple model, it allows structured and semi-structured data to be mixed, exposed, and shared across different applications." [27].

Given that framework, we can map the concepts of TRIZ onto a structure which can be read and processed by computers using HTTP and link them to substantial resources creating an environment to collaborate in innovative projects across the world using semantic technologies.

To collect and manage this development of the RDF source code, a git repository [29] is used and has been forked for developing the TRIZ Analysis Ontocard feature as the main goal of this work. The README manual of the repository provides necessary preliminaries for using and continue developing the project.

The main part of the modelling workflow is the appropriate design how to translate the ontological structure of the given topic into the RDF scope of three-word-sentences. Therefore the use of several namespaces to map each ontological aspect onto the architecture of subjects, predicates and objetcs is the key to deliver a well structured RDF Model.

The conceptualisations to be developed follow the basic assumptions and settings, that are elaborated in more detail in [6]. 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).

Furthermore several standard ontologies are used:

- `skos:` – the Simple Knowledge Organization System Namespace (mainly for labelling).
- `foaf` – the namespace of Friend of a Friend to describe real world objects.
- `rdfs:` – the namespace of RDF Schema (RDF properties and their domains).
- `cc:` – the namespace of the Creative Commons License (to address copy left approach).
- `owl:` – the namespace of the Web Ontology Language (to address web ontology conventions).

As in Souchkov's Glossary we use URIs as

`tc:HarmfulAction, tc:HarmfulFunction,`
`tc:HarmfulInteraction, tc:HarmfulMachine`

that combine the property and the subject of the property except for `tc:FlowSubstance`.

As you can see, the WUMM project introduced the `od:` namespace to extend the given ontological base of the TOP by its own concepts and provide explicit definitions of certain abstract models, which are yet unclarified.

To see which predicates are used in the `od:` namespace, you can query the WUMM SPARQL Endpoint [30]

```
SELECT DISTINCT ?p WHERE ?s ?p ?o FILTER regex(?p, 'od').
```

5 Modelling the Flow Analysis Ontocard

Flow Analysis, as a rather niche method in TRIZ is grounded by two major contributions: While Lyubomirsky [15] started with the introduction of a broad set of methods to fix certain problems assigned to flow as a functional magnitude, Lebedyev [13] [14] catched up on it and developed a systematic approach of analysis of flows based on the well known principles of TRIZ. Finally Eckardt [1] started to adopt this work integrating it as a part in the TOP.

As mentioned before and to rely on proper methods of the design of software components a few small changes to the taxonomy of Eckardt's ontology needed to be induced with several layers of abstraction (which are existing prototypically but are not described explicitely) and specification of the used predicates. The description of the modelling will mainly focus on these parts, while explaining the structure of the ontology itself remains exemplarily, as it has always been specified in the quoted material.

"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." [6]

0. The level of the *real-world system*.
1. The level of *modelling the real world system*.
2. The *level of the meta-model* as the actual (TRIZ) ontology on which the systemic concepts are *defined*.
3. The *modelling meta-level 2* at which the methodological concepts are defined.[1]

The methodological practice described in this work is mainly based at level 2, meaning the modelled objects and subjects in terms of RDF refer to the concept of classes used in object-oriented programming. Therefore all properties of RDF objects described here are generic class properties, which have to be instantiated separately as a RDF triples of `rdfs:types` referring a certain object of class properties.

The Flow Analysis Model can be separated in two parts. The *Flow Model* refers to the descriptive and functional properties of a flow, its (static) components as abstract or concrete real world objects, that define its properties and description methods, like textual or graphical

[1] See [6] for more details.

representations. The *Flow Analysis (Model)* describes the given context of a flow analysis, e.g. its requirements, rules that derive from general TRIZ principles or the LETS, which form an overall solution process containing methods how to solve problems using specific flow innovation methods.

5.1 `tc:Flow`

A flow is, on the one hand, a component within a system but on the other hand can describe a system as a dynamic entity in the flow of time. The descriptive and functional properties are modeled along the rules for a morphological table, i.e. defining a PropertyDomain and a list of allowed values, modelled with `od:allowedValues`, respective `valueOf` as listed below. The model of Eckardt is slightly extended to cover the full model used by Lebedyev. That is `tc:FlowDelegation` and the *Carrier Flow Model*, which is basically obligatory to model the flow of flows and modeled here as a special type of `tc:FlowFunctionality`.

Property	PropertyDomain
od:belongsTo	tc:StageValue
od:hasFlowType	tc:FlowType
od:hasFlowSubstance	tc:FlowSubstance
od:hasFlowFunctionality	tc:FlowFunctionality
od:hasFlowDefect	tc:FlowDefect
od:hasFlowSource	tc:FlowSource
od:hasMethodOfDescription	tc:MethodOfDescription

PropertyDomain	PropertyValues
tc:FlowType	tc:ComplexFlow, tc:DiscreteFlow, tc:ContinuousFlow
tc:FlowDelegation	tc:OpenFlow, tc:ClosedFlow
tc:FlowSubstance	tc:Information, tc:Energy, tc:Substance
tc:FlowFunctionality	tc:UsefulFlow, tc:HarmfulFlow, tc:ParasiticFlow, tc:CarrierFlow
tc:FlowDefect	tc:InsufficientFlow, tc:ExcessiveFlow, tc:BadControllableFlow, tc:AbsentFlow
tc:FlowSource	tc:ExternalSource, tc:InternalSource
tc:MethodOfDescription	tc:TextDescription, tc:ParametricDescription, tc:GraphicDescription

Here you can see the triples associated with the flow subject:

```
tc:Flow
  a skos:Concept, od:SouchkovGlossaryEntry ;
  od:WOPCategory od:Component;
  od:SouchkovCategory tc:FlowAnalysis ;
  od:SouchkovDefinition """A sequence of events that have the same common
    feature."""@en ;
  od:LebedyevDefinition """Motion of material, energetic or informational
    objects within a system"""@en ;
```

```
skos:prefLabel "Flow"@en, "Fluss"@de, "Поток"@ru ;
skos:altLabel "Stream"@en, "Strom"@de .
```

`tc:StageValue` refers to the general concept of `tc:ModeValue`s we'll see in action in the example, which is introduced later on. It defines different models of flows, which are used along the flow analysis transformation process. `od:Component` with its predicate `od:WOPCategory` defines whether a component of the analysed system is covered or with `od:PropertyDomain` properties of the flow as a systematic approach is described. Each subject, that refers to a TRIZ concept is a `skos:Concept`, as well as a `od:AdditionalConcept`. To indicate that a `skos:definition` is due to a certain author, we use subpredicates like `od:LebedyevDefinition` or `od:SouchkovDefinition`. The `od:SouchkovCategory` refers to a specific categorial object in Souchkov's glossary [19].

5.2 Subconcepts of `tc:Flow`

A flow consists of several physical parts (source, channel, receiver, control unit) that are components by their own and hence modeled with the predicate `od:subConceptOf`. For the moment they are attached to the flow with a single predicate `od:hasStaticFlowComponent`. Yet, they have several subcomponents, which are listed below.

Property	PropertyDomain
od:hasStaticFlowComponent	tc:StaticFlowComponent

Components	SubComponents
tc:ControlUnit	tc:Pump, tc:Valve
tc:Receiver	(has no subcomponents yet)
tc:Source	tc:Current, tc:Potential
tc:Channel	(has no subcomponents yet)

5.3 `tc:FlowAnalysis`

Eckardt's model of flow analysis is separated into two levels of abstraction. The *Top Level View* contains a self descriptive methodological approach, which parts of the flow analysis are relevant to carry out a flow transformation process. The *Technical View* describes how a transformation is prescribed by its compliance rules and its LETS and how requirements are met by solving the contradictions with several `tc:TechniqesOfModification`.

The ontological representation of such a process is extremely difficult and is not considered here. The ontology of the flow analysis is restricted to the description of the *formative elements of the analysis* and the creation of a manually generated *FlowTransformation* document.

5.3.1 Top Level View

A Flow Analysis is an methodological process. Its *Top Level View* conncects the given context and a set of rules, that define procedural policies to create a project plan, the

`tc:FlowTransformation` (a `foaf:Document`, in a yet hypothetic *Flow Transformation Description Language*) with the defined `tc:Flow` model in several stages to define steps of evolution of the technical system and how to process them to accomplish the desired transformative action[2].

Property	PropertyRange
od:describesFlow	tc:Flow
od:analysesFlow	tc:Flow
od:inContext	tc:Context
od:followsRules	tc:RulesOfFlowTransformation
od:usesDescriptionMethods	tc:Flow, ex:FlowModelAsIs, ex:FlowModelAsRequired, tc:FlowTransformation
od:generatesFlowTransformation	tc:FlowTransformation
od:applyChanges	tc:Flow

While these properties usually operate from the Domain of `tc:FlowAnalysis`, `od:applyChanges` gathers all approaches of the Flow Analysis finally in the `tc:FlowTransformation` and thus can be applied to the real system (level 0).

`tc:Context` encapsulates all information about the exact conditions of the analysis and thus are modelled as `od:subConceptOf`.

Concept	hasSubConcept
tc:Context	tc:Goal, tc:Requirement
tc:Requirement	tc:IncreaseUsefulFlow, tc:ReduceHarmfulFlow, tc:ReduceParasiticFlow

5.3.2 Technical View

The Technical View combines two central concepts how and why methods are applied in TRIZ. It connects the `tc:RulesOfFlowTransformation` with the *ModelAsIs* and the *ModelAsRequired*. It contains their rules of construction, integration of evaluation, their requirements and rules when to use which `tc:TechniquesOfModification` to establish a target transformation.

With `od:buildModelAsIs` and `od:buildModelAsRequired` we define the predicates to describe the process how specific instances of a *Flow Model* are built to define starting and desired target point of the flow transformation. Here, the `tc:RulesOfFlowTransformation` are attached, that build a framework of design directives, not only how the models are developed but also how these models are connected to establish a valid solution out of the issued circumstances of flow analysis. While `tc:RulesOfConstruction` and `tc:RulesofEvalution` refer to

[2]Addendum graebe: The abstraction levels (p. 6) are important here. The description of the methodology takes place at level 3, the application of the methodology uses language elements of level 2, the transformation of a concrete flow is described at level 1. Only at level 1 different instances of the flow exist next to each other as result of subsequent transformations. And these instances have much in common and do vary only a little. Our ontologisation assumes that different instances in the namespace `ex:` are described by different RDF graphs if necessary, the transformation itself is described in an instance of `tc:FlowTransformation` as `foaf:Document`. This is a version of the problem of whether one can step into the same river twice. From an ontological point of view, the answer to this question is clearly "yes".

a certain ex:FlowModelAsIs, the tc:RulesOfConstructionOfInconformRequirements and the tc:RulesOfApplyingLawsOfDevelopmentOfTechnicalSystems describe how the evaluated contradictions in the ex:FlowModelAsIs can be resolved via the knwon LETS to develop a ex:FlowModelAsRequired.

The tc:TechniquesOfModification is the last core piece of this ontologisation of Flow Analysis. It is a set of methods that are derived from certain LETS or so called trends occuring in technical systems that prescribe certain processings [10]. *They* are od:issuedAt the tc:ModelAsIs to solve a certain tc:FlowDefect. Each of these methods are modeled as od:subConceptOf tc:TechniquesOfModification and are categorised by the static components they are intended to change. Additionally, they are categorised by the overall requirement set in context.

For example:

```
tc:TechniquesOfModification
  od:issuedAt tc:ModelAsIs ;
  od:forms tc:ModelAsRequired ;
  od:hasSubConcept tc:ModificationOfReceiver, tc:ModificationOfSource,
    tc:ModificationOfChannel, tc:ModificationOfControlUnit ;
  skos:prefLabel "techniques of modification"@en,
    "Techniken zur Modifikation"@de ;
  skos:altLabel "techniques of transformation"@en,
    "Techniken zur Transformation"@de ;
  a skos:Concept, od:AdditionalConcept .

tc:ModificationOfChannel
  od:subConceptOf tc:TechniquesOfModification ;
  od:hasSubConcept tc:removeBottleNeck, tc:removeDeadZone,
    tc:removeGreyZone ,tc:changeFlowMedium, tc:outsourceFlow,
    tc:reduceAmountOfTransformations, tc:improvePermeability,
    tc:reduceLength, tc:ModificationOfControlUnit, tc:combineUsefulFlows,
    tc:channelFlow, tc:implementBottleNeck, tc:implementDeadZone,
    tc:reducePermeability, tc:extendLength ;
  skos:prefLabel "modification of the channel"@en,
    "Modifikation des Kanals"@de ;
  skos:altLabel "transformation of the channel"@en,
    "Transformation des Kanals"@de ;
  a skos:Concept, od:AdditionalConcept .

tc:removeBottleNeck
  od:subConceptOf tc:ModificationOfChannel, tc:IncreaseUsefulFlow ;
  skos:prefLabel "remove bottleneck"@en, "Flaschenhals entfernen"@de ;
  a skos:Concept, od:AdditionalConcept .
```

6 Example for Flow Analysis

6.1 Model of an Internal Combustion Engine

To illustrate how this machine readable ontology can be used to model real world problems, we introduce an example on how to improve the exhaust flow of an internal combustion engine (ICE), which have been proposed by Lebedyev [14]. We show how the flow model is implemented, address some weaknesses that occured using the given Ontocard and translating it into RDF and show how the Flow Analysis is applicated.

We used *Wikipedia*[3] to adjust the given example and meet the typical terminology. The granularity stays on the level of the generality of the used terms. This is an important property of ontological modelling. You can go up and down that ladder as the used analytical methods and vocabulary allow you to do. In the example this can be seen in the specific categorisations and assignments to each level of abstraction (see 6.), which change when you enter the terminology of an ICE or are abstracted, when only the general categories of the flow analysis concepts are met. This depends a lot on the given context, which is why there is yet no general solution when to use which level of abstraction is sufficient. Intuitively, the more the problem is located in specific components, that refer to a special context in that domain, the more concrete the terminology needs to be.

First we present the basic model of an ICE:

```
ex:BasicInternalCombustionEngine
  a tc:Flow ;
  od:hasFlowType tc:ComplexFlow ;
  od:hasFlowSubstance ex:Fuel, ex:Transmission, ex:Heat, ex:Work,
    ex:Cooling, ex:Exhaust ;
  od:hasFlowFunctionality tc:CarrierFlow ;
  od:hasFlowDelegation tc:OpenFlow ;
  od:hasFlowSource tc:InternalSource ;
  od:hasStaticFlowComponent ex:FuelTank, ex:FuelPump,
    ex:FuelSupplyPipe, ex:Atmosphere, ex:Compressor, ex:AirSupplyPipe,
    ex:CoolingTank, ex:CoolingPump, ex:CoolingWaterJacket,
    ex:IntakeValve, ex:InletCamshaft, ex:CombustionChamber,
    ex:SparkPlug, ex:Cylinder, ex:Piston, ex:ConnectionRod,
    ex:Crankshaft, ex:ExhaustCamshaft, ex:ExhaustValve,
    ex:ExhaustPipe ;
  od:hasMethodOfDescription tc:ParametricDescription,
    ex:ICE3DModel ;
  skos:prefLabel "Basic internal combustion engine"@en,
    "Grundlegendes Modell eines Verbrennungsmotors"@de ;
  skos:note """Basic construction model from wikipedia
    <https://en.wikipedia.org/wiki/Internal_combustion_engine>""" .
```

Here you can see some decisions on abstraction and findings. We focused on concrete descriptions of `tc:FlowSubstance` and `tc:StaticFlowComponent`s as they are sufficient to show the

[3] `https://en.wikipedia.org/wiki/Internal_combustion_engine`

problems of an exhaust flow and how to model an easy improvement.

Thus, the ICE is already a complex machine, where you have at least two interdependent system levels, i.e. a flow of flows. But the given Ontocard doesn't define rules how systems of systems, respectively flows of flows, are modelled, because the Flow Model lacks a model of self description. Thats why this modelling is still experimental. Therefore we made some assumptions:

- A `tc:FlowSubstance` on the second level is a `tc:Flow` on the first level.
- The `tc:FlowFunctionality` of a second order flow is always a `tc:CarrierFlow`[4].

So let us have a look at the `ex:Exhaust` flow:

```
ex:Exhaust
  a tc:Flow ;
  od:hasFlowType tc:DiscreteFlow ;
  od:hasFlowSubstance tc:Substance ;
  od:hasFlowFunctionality tc:HarmfulFlow ;
  od:hasFlowDelegation tc:OpenFlow ;
  od:hasFlowSource tc:InternalSource ;
  od:hasStaticFlowComponent ex:CombustionChamber, ex:ExhaustPipe,
    ex:Atmosphere, ex:ExhaustValve, ex:ExhaustCamshaft ;
  skos:prefLabel "The exhaust flow"@en, "Abgasstrom"@de .
```

`ex:Exhaust` is a `tc:HarmfulFlow` with several static components. These components are not exclusive for the exhaust flow. They are used in other flow systems as different flow components. For example the `ex:CombustionChamber`, here modelled as `a tc:Potential`, is `a tc:Channel` in the system of the `tc:Heat` flow. To inherit context sensitivity to these subjects, we use `od:specialCase`: [28]

```
ex:CombustionChamber
  od:specialCase ex:Heat, ex:InternalCombustionEngine ;
  a tc:Channel ;
  skos:prefLabel "Combustion chamber"@en, "Verbrennungskammer"@de .
```

6.2 Improvement of the `tc:Exhaust` Flow

We now have a model of a *system as is*, which can be used for improvement. As already mentioned, we want to describe the process of exhaust flow innovation. Given the context of `ex:PollutionRestrictions`, we define the goal `ex:SatisfactionOfPollutionRestrictions` with the requirements to reduce the exhaust flow while the work flow needs to be preserved as good as possible

```
  ex:InnovationOfExhaustFlow od:hasRequirement ex:ReduceExhaustFlow,
    ex:PreserveWorkFlow .
```

[4] See chapter 7 in [14].

We define a specific `foaf:Document` to hold all information about the innovation project, that describes which flows need to be described and analysed, following a *ModelAsIs*, as well as a *ModelAsRequired*, (called `ex:ICEExhaustionModel` and `ex:buildICEExhaustionInnovationModel`).

```
ex:InnovationOfExhaustFlow
  a tc:FlowAnalysis ;
  od:generatesFlowTransformation ex:ExhaustionFlowTransformation ;
  od:describesFlow ex:Exhaust, ex:ICEExhaustionModel,
  ex:ICEExhaustionInnovationModel ;
  od:analysesFlow ex:Exhaust, ex:Fuel, ex:Air, ex:AirFuelMixture, ex:Heat,
    ex:Work ;
  od:inContext ex:PollutionRestrictions ;
  od:hasGoal ex:SatisfactionOfPollutionRestrictions ;
  od:hasRequirement ex:ReduceExhaustFlow, ex:PreserveWorkFlow ;
    ex:buildICEExhaustionModel ex:ICEExhaustionModel ;
    ex:buildICEExhaustionInnovationModel ex:ICEExhaustionInnovationModel ;
  tc:applyRulesofApplyingLawsOfDevelopmentOfTechnicalSystems
    ex:ICEExhaustionInnovationModel ;
  skos:prefLabel "Innovation of the exhaust flow"@en,
    "Innovation des Abgassstroms"@de .
```

Using the `tc:RulesofEvalution` we can define an `ex:EvaluationOfExhaustFlow`, that detetcs a `tc:FlowDefect` which needs to be fixed, which can be found in the `ex:buildICE-ExhaustionModel`:

```
ex:EvaluationOfExhaustFlow
  a tc:RulesOfEvaluation ;
  od:detects ex:HighDispersedExhaust ;
  skos:prefLabel "Evaluation of the exhaust flow"@en,
    "Evaluation des Abgassstroms"@de .

ex:HighDispersedExhaust
  a tc:ExcessiveFlow ;
  skos:prefLabel "high dispersed exhaust"@en,
    "Abgasstrom mit (zu) vielen Schwebstoffen"@de .

ex:ICEExhaustionModel
  a tc:Flow, tc:ModelAsIs ;
  od:hasFlowType tc:ComplexFlow ;
  od:hasFlowSubstance ex:Fuel, ex:Transmission, ex:Heat, ex:Work,
    ex:Cooling, ex:Exhaust ;
  od:hasFlowFunctionality tc:CarrierFlow ;
  od:hasFlowDefect ex:HighDispersedExhaust ;
  od:hasFlowDelegation tc:OpenFlow ;
  od:hasFlowSource tc:InternalSource ;
  od:hasStaticFlowComponent ex:FuelTank, ex:FuelPump, ex:FuelSupplyPipe,
    ex:Atmosphere, ex:Compressor, ex:AirSupplyPipe, ex:CoolingTank,
```

```
    ex:CoolingPump, ex:CoolingWaterJacket, ex:IntakeValve,
    ex:InletCamshaft, ex:CombustionChamber, ex:SparkPlug, ex:Cylinder,
    ex:Piston, ex:ConnectionRod, ex:Crankshaft, ex:ExhaustCamshaft,
    ex:ExhaustValve, ex:ExhaustPipe ;
  od:hasMethodOfDescription tc:ParametricDescription, ex:ICE3DModel ;
  od:generatesFlowTransformation ex:ExhaustionFlowTransformation ;
  skos:prefLabel "ICE exhaustion model"@en, "Abgasmodell des Verbrennungsmotors"@de .
```

Using these informations, certain `tc:TechniquesOfModification` can be applied to fix the `tc:FlowDefect` and fullfill the set `tc:Requirements`.

```
ex:installCatalyticConverter
  a tc:reducePermeability ;
  skos:prefLabel "Install catalytic converter"@en,
    "Katalysator einbauen"@de .

ex:addGasolineAdditive
  a tc:compensateWithSecondFlow ;
  skos:prefLabel "Add gasoline additive flow"@en,
    "Fluss für Kraftstoffzusatz hinzufügen" .
```

They lead to an additional flow `ex:FuelAdditive` and upgraded channel component of the `ex:Exhaust` flow.

```
ex:FuelAdditive
  a tc:Flow ;
  od:hasFlowType tc:DiscreteFlow ;
  od:hasFlowSubstance tc:Substance ;
  od:hasFlowFunctionality tc:UsefulFlow ;
  od:hasFlowDelegation tc:ClosedFlow ;
  od:hasFlowSource tc:InternalSource ;
  od:hasStaticFlowComponent ex:IntakeValve, ex:InletCamshaft,
    ex:FuelAdditivePump, ex:FuelAdditiveTank, ex:FuelAdditiveSupplyPipe,
    ex:CombustionChamber ;
  ex:FuelSuppylPipe, ex:CombustionChamber ;
  skos:prefLabel "Flow of fuel additive"@en,
    "Fluss des Kraftstoffzusatzes"@de .

ex:ExhaustPipeWithCatalyticConverter
  a tc:Channel ;
  skos:prefLabel "Exhaust pipe with catalytic converter"@en,
    "Auspuff mit Katalysator"@de .
```

This results in the final *ModelAsRequired* `ex:buildICEExhaustionInnovationModel`:

```
ex:ICEExhaustionInnovationModel
  a tc:Flow, tc:ModelAsRequired ;
```

```
od:hasFlowType tc:ComplexFlow ;
od:hasFlowSubstance ex:Fuel, ex:FuelAdditive, ex:Transmission, ex:Heat,
  ex:Work, ex:Cooling, ex:Exhaust ;
od:hasFlowFunctionality tc:CarrierFlow ;
od:hasFlowDelegation tc:OpenFlow ;
od:hasFlowSource tc:InternalSource ;
od:hasStaticFlowComponent ex:FuelTank, ex:FuelAdditiveTank, ex:FuelPump,
  ex:FuelAdditivePump, ex:FuelSupplyPipe, ex:FuelAdditiveSupplyPipe,
  ex:Atmosphere, ex:Compressor, ex:AirSupplyPipe, ex:CoolingTank,
  ex:CoolingPump, ex:CoolingWaterJacket, ex:IntakeValve, ex:InletCamshaft,
  ex:CombustionChamber, ex:SparkPlug, ex:Cylinder, ex:Piston,
  ex:ConnectionRod, ex:Crankshaft, ex:ExhaustCamshaft, ex:ExhaustValve,
  ex:ExhaustPipeWithCatalyticConverter ;
od:hasMethodOfDescription tc:ParametricDescription, ex:ICE3DModel ;
od:generatesFlowTransformation ex:ExhaustionFlowTransformation ;
skos:prefLabel "ICE exhaustion innovation model"@en,
  "Innovatives Abgasmodell des Verbrennungsmotors"@de .
```

7 Conclusion, critical review and further development

7.1 Outside view – embedding the Flow Analysis Ontocard

For now this TRIZ Ontocard remains mostly isolated from its embedded environment within the ontological hierarchy of the TOP and rules how and when to apply a flow analysis method within the main process of inventive problem solving as it is proposed by the TRIZ model. A long term goal may be, that from an autonomous engineers perspective it should be possible to gain a programmatic manual how to solve a specific inventive or systemic problem only with the given RDF of the TRIZ Ontology, for which it has to be serialisable.

To embed the Flow Analysis Ontocard either a top down or a bottom up approach is applicable. For top down it needs to be clarified how Flow Analysis as one of the Top Level Ontologies fits in the structure with the remaining parts, especially LETS, the TRIZ application areas and the TRIZ Model. As it seems Eckardt already considered the connection to LETS and the TRIZ Model, for which these areas should be prioritised in the further development.

Further, not only in terms of ontologisation, it remains unclear where exactly the methodological framework of flow analysis is settled, how it can be derived or transformed from other system analysis methods, which methods from different approaches it includes or how other techniques may benefit from including flow analysis methods in terms of multiperspective modelling. There are several concepts, which use similar semantics or patterns to describe a problem, only on different levels of abstraction, especially those, which work on graph like structures. First mentioned here should be the process analysis framework and additionally function analysis as they intersect on a general conceptual level.

Looking at processes, functions and flows in TRIZ terminology [19] [1] a flow model is a *special process model*, where the process is a function in time between each of the components of the process [10]. In the flow analysis model a flow is also a function, though it appears not as a function between components, but as an emergent property of its components properties,

which do not have functional properties only, but relational properties too, especially the control unit and the channel, which can not be grouped by source or receiver components bijectively in time. So it is neither a classical process model nor a function model in its strict sense, and combines aspects of both as flow is a function in time between nodes, which combine multifold properties of different components, but with interdependent properties like ratio of flow rate and capacity. In the status quo these aspects are not covered explicitely by the flow analysis model.

Further, the concept of *substance field models* (SFM) is closely related to the FAM. "Analog to the function model the components are also connected via a function, but in the substance field model the energy or a specific form of energy is added." [10]. Thus, the substance field model matches the systemic structure of flow composition, leading to its definition as a physical quantity as a product of a field and a (sur)face [4]. Even more, SFM uses the terminology of useful and harmful effects, too [10]. Therefore the linkage between those methods should be generalised in the future.

Next mentioned should be its connection to the *root conflict analysis*. As already mentioned in the introduction, cause and effect analysis and the identification of cause-effect-chains is a well established concept in TRIZ. As many rules of flow analysis and its techniques to change flows obviously imply a root conflict analysis, there is not yet an ontological relation between these concepts. This should be made explicit.

Yet, there are several other concepts in TRIZ that can be exploited using Flow Analysis. Generally speaking, all models which describe progress in time and map the ability and change of motion of some potential can be analysed using FAM. So it seems worth to review the concept of evolution of technical systems in terms of flow analysis, at least ontologically.

While having focused mainly on the MATRIZ school, there exists a flow model in OTSM-TRIZ as well. Khomenko proposed a *Problem Flow Approach*, which uses a more abstract concept of flow and combines aspects of network modelling of problems and solutions and technology flow characteristics [7] [8] [9]. This should be analysed in detail to gain knowledge about how these concepts align with the approach of the MATRIZ school or can uncover additional perspectives of flow modelling.

7.2 Inside view – methodological critics

Eckardt's ontology of flow analysis, as it is, matches the basic methodological approach of TRIZ on how to change a problematic system in terms of requirement engineering. It has a descriptive flow model and categorial connections to certain rules, that are needed to construct and evaluate a *Flow Model As It Is* and how they are used to transform it into a *Flow Model As Required*, while including some techniques how to change its static components to aim for the preferred requirements. Beside that basic descriptive perspective it does not meet the need how the demanded modifications of flow systems are being processed, as it has been already noticed by Eckardt (algorithms and a matrix of methods of transformation of the model as is to the model as it should be as a possible next step [1]). The proposed *Techniques of Modification* used by Eckardt for this transformation are based on the work of Lyubomirsky. While Eckardt implemented his categories of techniques how to increase useful flows and reduce of harmful flows by changing different static components, a more detailed analytic point of view needs to address which problematic effects can be solved using certain

techniques. Furthermore, we need a map which decodes what properties of components can cause components to occur harmful properties, like grey zones, stagnation zones and so on. But to establish these performative aspects the descriptive flow model needs to be improved as well. To allow the modelling of higher order flow systems, we've upgraded Eckardt's proposal to the concepts introduced by Lebedyev [14] and added experimental design guidelines to connect flows on different systemic levels in RDF. These should be reviewed and adjusted to a profound level of acceptance.

Beside having properties for static components, which has already been regarded by Lyubomirsky and improved by Lebedyev (like capacity or conductivity), *dynamic properties of flow*, for example resonance or flow rate, have to be defined to model how specific flow properties emerge out of the constellation of certain configurations of its static components and how they can be affected changing them. Reflecting this, a concept of how the process of changing components change the properties of the emerging flow and how specific flow properties can be achieved by changing certain static components is needed.

This concludes, that an additional level of abstraction is needed to account for the interconnection of flow and its components. The question is, how it can be modelled that a stagnation zone is only bad for useful flows but can be used to reduce harmful flows? At this point, only singular flows are issued by the FAM precisely, meaning that it is not clear, if a global flow system is issued or a specific local flow. But it has to be considered, that synergetic effects on the one hand can cause contrary reactions on the other hand. A flow analysis therefore must take the complexity of flow networks into account. Beside extending the descriptive flow model to spotlight on the systemic causes of these problems, this is also a methodological issue. At the moment FAM focuses on methods, that emerge out of experienced based best practices in an engineering context. While establishing methodological concepts, that aim for generalisation of categorial problem classes, they are mainly built for qualitive substance and energy flows only. Not to forget, that there are some blueprints addressing information flows, too, like the problem of conversion of flows and the concept of carrier and mediating flows, but these remain cursory. This rises the question: Which domains can be addressed by the actual model and which not? And furthermore, are the foundations of the TRIZ model sufficient to address complex problems in flow analysis, that cannot clearly distinct between useful and harmful flows. If we look at these techniques, they try to fulfill requirements orienting on one parameter at the time. But in practice, especially in a network of flows, you need to take multiple parameters into account, respectively to distinguish between global and local parameterisation. Therefore a quantisation or numerical framework is needed, like it is applied by graph theoretical concepts and optimisation theory [3]. Mentioning the discussion of the *TRIZ Ontology Webinar* [31], the inclusion of STEM or MINT concepts is a huge black spot. Neither general concepts like *Material Flow Analysis* [16] or *Fluid Dynamics* [2] are considered, nor well established ones on information flow, like *Supply Chain Management* or *Traffic Flow Control* [24]. It is clear that the integration of each of these concepts is a load of work but necessary to obtain a general model of flow analysis in the future.

References

[1] Olga Eckardt (2020). Онтологические диаграммы. Модель потока и потоковый анализ (Ontology diagrams. Flow model and flow analysis) (in Russian). TRIZ Ontology Project – Webinar, Oct. 27, 2020. https://wumm-project.github.io/2020-10-27.

[2] Fluid Dynamics – English Wikipedia. https://en.wikipedia.org/wiki/Fluid_dynamics. Last visited 07 May 2021.

[3] Flow Network – English Wikipedia. https://en.wikipedia.org/wiki/Flow_network. Last visited 07 May 2021.

[4] Fluss (Physik) – German Wikipedia. https://de.wikipedia.org/wiki/Fluss_(Physik). Last visited 03 May 2021.

[5] Hans-Gert Gräbe (2020). Die Menschen und ihre Technischen Systeme. LIFIS Online. doi:10.14625/graebe_20200519.

[6] Hans-Gert Gräbe (2021). About the WUMM modelling concepts of a TRIZ ontology. Manuscript, April 2021. https://wumm-project.github.io/Texts/WOP-Basics.pdf.

[7] Nikolay N. Khomenko (2007). OTSM Problem Flow Networks approach: application for competitive analysis of patents and other sources of information. 2000-2007. https://otsm-triz.org. Last visited 07 May 2021.

[8] Nikolay N. Khomenko (2007). OTSM Network of Problems: Draft for Discussion and Improvements. https://otsm-triz.org. Last visited 07 May 2021.

[9] Nikolay N. Khomenko. Problem Flow Technology. List of relevant publications. https://otsm-triz.org. Last visited 07 May 2021.

[10] Karl Koltze, Valeri Souchkov (2018). Systematische Innovation. Hanser Verlag, München.

[11] Andrey Kuryan, Mikhail Rubin, Nikolay Shchedrin, Olga Eckardt, Natalia Rubina (2020). TRIZ Ontology. Currect State and Perspectives. https://wumm-project.github.io/Ontology.html

[12] Simon Litvin, Vladimir Petrov, Mikhail Rubin, Vikctor Fey (2007). TRIZ Body of Knowledge.

[13] Yuri Lebedyev (2011). Classification of flows in technical systems.

[14] Yuri Lebedyev (2015). Совершенствование инструментов потокового анализа (Improvement of flow analysis tools). Work for certification as Master of TRIZ (in Russian). https://r1.nubex.ru/s828-c8b/f2134_d6/Yuri%20Lebedev_TRIZ%20Master%20dissertation_RUS.pdf

[15] Alex Lyubomirsky (2006). Law on efficiency improvement use of substance, energy and information flows.

[16] Material Flow Analysis – English Wikipedia. https://en.wikipedia.org/wiki/Material_flow_analysis. Last visited 07 May 2021.

[17] Vladimir Petrov, Michail Rubin, Simon Litvin (2007). Fundamentals of TRIZ Theory of Inventive Problem Solving (in Russian). Reprinted in 2020. ISBN 978-5-4496-8183-6.

[18] Vladimir Petrov (2019). TRIZ. Theory of Inventive Problem Solving – Level 1. Springer International Publishing.

[19] Valeri Souchkov (2018). Glossary of TRIZ and TRIZ-related terms. Version 1.2. https://matriz.org/wp-content/uploads/2016/11/TRIZGlossaryVersion1_2.pdf.

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

[21] Supply Chain Management – English Wikipedia. https://en.wikipedia.org/wiki/Supply_chain_management. Last visited 07 May 2021.

[22] Онтология ТРИЗ (TRIZ Ontology Project – TOP) of the TRIZ Developer Summit. https:\\triz-summit.ru/Onto_TRIZ/.

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

[24] Traffic Flow – English Wikipedia. https://en.wikipedia.org/wiki/Traffic_flow_(computer_networking). Last visited 07 May 2021.

[25] The WUMM Project. https://wumm-project.github.io/About.html. Last visited 22 February 2021.

[26] The WUMM TOP Companion Project. https://wumm-project.github.io/Ontology. Last visited 22 February 2021.

[27] The W3C Resource Description Framework (RDF). https://www.w3.org/RDF/. Last visited 27 April 2021.

[28] RDFC – RDF with Contexts. A proposal. https://www.w3.org/2011/rdf-wg/wiki/RDFwithContexts. Last visited 02 May 2021.

[29] The RDFData git repository. https://github.com/wumm-project/RDFData. Last visited 22 February 2021.

[30] The WUMM SPARQL Endpoint. http:\\wumm.uni-leipzig.de:8891/sparql. Last visited 22 February 2021.

[31] The TRIZ Ontology Webinar. https://wumm-project.github.io/2020-10-27. Last visited 07 May 2021.