Modeling Software in TRIZ

Horst T. Nähler, Barbara Gronauer, Tiziana Bertoncelli, Hartmut Beckmann, Jerzy Chrząszcz, Oliver Mayer

▶ **To cite this version:**

Horst T. Nähler, Barbara Gronauer, Tiziana Bertoncelli, Hartmut Beckmann, Jerzy Chrząszcz, et al.. Modeling Software in TRIZ. 22th International TRIZ Future Conference (TFC), Sep 2022, Warsaw, Poland. pp.261-272, 10.1007/978-3-031-17288-5_23 . hal-04449815

HAL Id: hal-04449815
https://inria.hal.science/hal-04449815v1

Submitted on 12 Feb 2024

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

Distributed under a Creative Commons Attribution 4.0 International License

This document is the original author manuscript of a paper submitted to an IFIP conference proceedings or other IFIP publication by Springer Nature. As such, there may be some differences in the official published version of the paper. Such differences, if any, are usually due to reformatting during preparation for publication or minor corrections made by the author(s) during final proofreading of the publication manuscript.

Modeling Software in TRIZ

Horst Nähler[1][0000-0002-8853-815X], Barbara Gronauer[2][0000-0001-7054-321X], Tiziana Bertoncelli[3], Hartmut Beckmann[4][0000-0001-5965-4672], Jerzy Chrząszcz[5] [0000-0003-3257-9726], Oliver Mayer[6]

[1] c4pi - Center for Product-Innovation, 36088 Hünfeld, Germany
[2] StrategieInnovation, 36088 Hünfeld, Germany
[3] ANSYS Germany GmbH, 83624 Otterfing, Germany
[4] Giesecke+Devrient GmbH, 81677 München, Germany
[5] Warsaw University of Technology, Institute of Computer Science, 00-665 Warsaw, Poland
[6] Bayern Innovativ GmbH, 90402 Nürnberg, Germany

naehler@c4pi.de

Abstract. Although many papers have been published in the past on the use of TRIZ in the field of Information Technology, questions still arise as to whether the TRIZ methodology is also suitable for software as purely intangible systems. Doubts remain because the methodology was developed at a time when patents for IT and software systems, in contrast to physical products, could not yet be considered in the underlying patent analysis.
Because of these questions, this paper examines the transferability of the TRIZ methodology to software systems on the basis of two fundamental TRIZ concepts: the Law of System Completeness and the Function Analysis for Processes. After an introduction regarding the immaterial nature of software systems, the transferability of TRIZ concepts to the software domain is shown with the help of case studies and ends with an outlook on possible further additions to the conceptual mapping of TRIZ to the software domain.

Keywords: TRIZ, Software, Information Technology, Function Analysis for Processes, Function Modelling for Software, Software Development.

1 Introduction

The application of TRIZ tools to the IT domain has already been extensively assessed and presented: e.g., [1] gives a comprehensive overview of software-related works, [2,3] assess function analysis for products, [4] contains aspects of Substance-Field modelling, [5] elaborates on the Laws of Engineering System Evolution with respect to information systems. Especially the transfer of the Inventive Principles to the IT domain has been covered by [6,7,8,9] and even a dedicated view on programming under TRIZ aspects is given in [10]. Moreover, a comprehensive book with a specific contradiction matrix was published already in 2008 [11]. Despite the number of publications, the authors repeatedly are faced with questions and doubts of TRIZ students if and how

TRIZ might be a useful asset in the modern world, especially when dealing with IT systems and specifically with software. These doubts are mainly fueled by the origins of TRIZ in the era of mechanical engineering and the associated wording of some classical TRIZ tools or examples given in classical TRIZ literature. Basic TRIZ concepts like the model of a complete system seem to exclusively address classical material engineering systems.

While the use of TRIZ concepts within primarily hardware-related IT topics requires manageable transfer performance, e.g., see examples in [11], the perceived "immateriality" of software makes it harder to find connecting points between TRIZ concepts and the software domain. Especially topics related to high level or platform independent programming languages are very much de-coupled from the hardware level of computer systems, making the application of the TRIZ method to software more difficult.

During the work of the authors on the topic of modelling IT systems in TRIZ [13], where we assessed IT systems as a combination of software and hardware, the following question arose: *How can TRIZ concepts and modeling approaches be mapped on pure software topics?*

Consequently, the goal of this paper is to expound how basic concepts of the Theory of Inventive Problem Solving can be mapped to the software domain. For this the authors focused on the conceptual modelling aspect of Altshuller's Law of System Completeness and the Function Analysis for Processes.

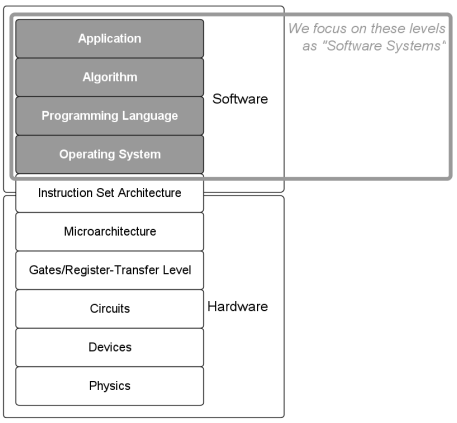

Fig. 1. Abstraction Layers in Computer Systems

By mapping both topics to the software domain the authors want to add to a conceptual and practical foundation of using TRIZ in the software domain. In this paper, the authors strictly focus on the software level of computer systems as described in [14,15,16]. Within this domain, we use the term "software system" as a differentiation to "engineering system" to highlight the immaterial nature of such systems (see Fig. 1). Additionally, we provided an overview of the definitions of terms used in Table 4.

2 A Complete Software System

As a fundamental concept of TRIZ, Altshuller's Law of System Completeness represents the basic structure of an engineering system. The concept describes necessary elements of a working engineering system. Building on this concept, additional practical instruments like the sequence of acquiring necessary parts from the supersystem or transferring functionalities back to the supersystem along the evolution of engineering systems were derived [17].

To approach software systems from a TRIZ viewpoint, the Law of System Completeness can be described in an analog way. However, several modifications reflecting the specific nature of software have to be introduced (see Fig. 2).

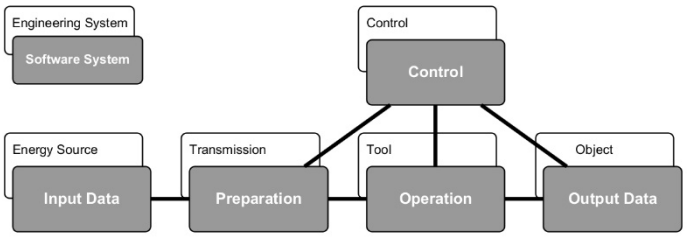

Fig. 2. Complete Software System

When mapped onto a software system, the Law of System Completeness reflects the structural composition of the system and the flow and modification of input data through a software system to create output data. The parts identified as required to achieve that functionality are described in Table 1.

Table 1. Parts of a Complete Software System

Parts of a Complete Software System		**RAW File Editing of Digital Photographs**
Input Data	Input data is the equivalent to the energy source for a logical software system. The input data kicks off and drives the software system.	RAW-file (raw image data)
Preparation	Preparation includes all necessary steps to modify data to bring it into a form and structure to be successfully processed. Preparation can include assembling, compiling, or padding of input data.	Import interface / translator for camera specific RAW data
Operation	The part of the software system executing the main function, all operations which create the output data.	Modification of Brightness, contrast, color,…; translation into compressed data (e.g. jpg)
Output Data	Output data is the outcome of the operations and can subsequently be used for other adjacent software systems, IT systems or users as input data.	JPG file (compressed image data)

The awareness of the individual parts that need to be present to form a working software system leads to conscious decisions regarding possible development directions:

1. Awareness of necessary components that form a working software system,

2. Awareness of where the component currently is located (subsystem/supersystem),
3. Exploring potential for bringing necessary components from supersystem into the software system at hand,
4. Exploring potential for bringing components currently located in the software system to the supersystem.

Examples for above mentioned mechanisms can be observed in contemporary developments. Data lakes provide vast amounts of heterogeneous input data, structured and unstructured. To efficiently process those data, software systems need to include the preparation functionality.

On the opposite, cloud applications have transferred the operations to the supersystem, which in this case is represented by software systems running on remote servers. Local software systems may just comprise interfaces for capturing input data and, if applicable, means for representing or providing output data for further use.

3 Process Function Model for Software

As indicated in the above section, software system performs specific operations on the input to generate the output. While TRIZ function analysis for products seems to be an appropriate tool for decomposition and identifying functional disadvantages regarding interactions between software modules, the authors focused on the process aspect of software systems. Therefore, the application of the TRIZ function analysis for processes in the software domain has been assessed.

In the terminology for TRIZ function analysis for processes [17], a software system can be divided into sequential or parallel operations (process steps) that perform certain functions (sub-steps). This structure represents the flow of data through the process and the modifications performed on the data. The function analysis for processes requires a conscious definition of the boundaries and the abstraction level of the system to be modelled.

One goal of TRIZ function analysis for processes is the evaluation of the functionality of each operation. This is achieved using several categories for functions inside each operation [17].

The categories listed in Table 2 were taken from the function analysis for processes for engineering systems and are applied here to qualify useful functions in software processes as well. The table reflects the importance of each function type, with productive functions being the most valuable (highest ranking value, e.g., 5) and corrective functions being the least valuable (lowest ranking value, e.g., 1).

Additionally, function disadvantages like harmful, insufficient, or excessive functions can be modelled in the context of function analysis for processes to analyze shortcomings or potential errors in a given software system.

Table 2. Function Categories for Function Analysis for Processes

Function category	Explanation	Example
Productive functions Function rank 5	Productive functions irreversibly change parameters of the output, in software systems they cause irreversible and intentional changes that are "visible" in the output data.	An example is Algebraic Multigrid (AMG) with which equation systems in structural analysis are solved.
Providing Supporting functions Function rank 4	Supporting functions only temporarily change parameters of the output. Those changes are not visible in the output data.	An example is multiplying an integer number with a factor of e.g. 1000 to increase accuracy of a calculation. However, this function causes the defect of offsetting the value by the factor used. This defect needs to be corrected by a subsequent corrective function.
Providing Transport functions Function rank 3	Transport functions change the location of data.	Exemplarily, in logical software operations this could be transfer instructions or logical shifts that move data.
Providing Measurement functions Function rank 2	Measurement functions provide information about parameters and properties of objects.	Any function that e.g., checks the value of data (variable, parameter, input data) can be considered a measurement function, consequently conditional jumps contain measurement functions.
Corrective functions Function rank 1	Corrective functions are directed towards a defect. Defects are caused by harmful, insufficient useful or excessive useful functions.	As mentioned above, a corrective function can be the division of an integer number to set the value back to the original state.

The case study presented below demonstrate the application of the TRIZ function analysis for processes in the software domain. While the first case study focusses on the function categories and identification of possible function disadvantages, the second case study explores the potential of trimming for processes in the context of software systems, leading to initial ideas for improving the given process.

3.1 Case study 1: Multiphysics simulation process

The following example illustrates a possible approach to build a function model for processes for an engineering software simulation workflow from the perspective of the software user, typically a research or application engineer, in charge of a virtual prototyping project or a digital twin construction. The workflow is built as a cascade of software modules, each of them either activated by a user or by an overlay software module. Those software modules are typically CAD (Computer Aided Design) and differential equation solvers (i.e., finite element, finite difference, boundary element methods) called in a given sequence.

We can see an application example, representing the usual sequence of operations for a multi-physics software engineering workflow based on weak coupling, where a first physical domain is analyzed and the outcome serves as input for the next step; this is widespread in electrothermal applications, where the electric or electromagnetic losses computed during the first analysis serve as input for the subsequent thermal analysis. Weak coupling, where fields are solved in a sequence, is normally preferred to strong coupling, where a set of equations representing all the fields interaction are

solved at the same time, since the different physical phenomena show in most application very different time constants.

Usually, losses are obtained solving for electric or magnetic field on a given geometrical domain: At first a geometry layout is produced and simplified to represent only the parts relevant to this analysis. This simplified geometry is then discretized in a number of subdomains where a numerical technique is used to solve for electric or electromagnetic fields. In the post-processing stage, the energy losses are computed from the field results; those losses are mapped to the geometry which in turn is again simplified and discretized; the subsequent numerical process to solve the thermal behavior of the system is then applied and thermal fields computed. This procedure can be again iterated if for instance a next step is introduced to analyze the effect of thermal fields on the structural behavior of the system.

The function model for this process is presented in tabular form [**Table 3**].

Table 3. Function Model for Multiphysics Simulation Process

Order	Operation	Function	Result	Function category
1	Initial Data Specification	Assign Design Parameters	Design Parameters fixed	Providing / supporting
2	Device Design	Compute Initial Design Details	Initial Design	Productive
3	Geometry Generation	Translate into Geometrical Entity Description	Geometry Layout	Providing / supportive
4	Symmetry Recognition	Identify Model Useful Subsection	Minimum Geometrical Module	Corrective
5	Defeaturing SW	Simplify unnecessary Details	Defeatured Geometrical Module	Corrective
6	Setup Definition	Assign Excitation, Boundary Conditions	Simulation Model	Productive
7	Mesh Generation	Definition of finite element shape and node positions	Geometry replaced by node assembly	Productive
8	Solution Physical Domain 1	Node EM field values computed	Node field values	Productive
9	Post-processing	Compute design performance data	Design performance parameters	Productive
10	New mesh generation	Definition of finite element shape and node positions	New node assembly	Productive
11	Solution Physical Domain 2	Node thermal field values computed	Node field values	Productive
12	Post-processing	Compute design performance data	Design performance parameters	Productive

Such a process function model can capture very well the temporal sequence of all the operations and can be as detailed as needed; it still neglects details to the specific numerical technologies adopted by each step, and this added granularity could be helpful for the developer.

Exemplarily, the following aspects highlight parts of the process where problematic situations can be identified.

Operation 7+12: The mesh generation stage is indeed productive, since it produces the finite element assembly upon which the field equations are solved. It must be tuned

wisely since overmesh (too fine granularity of the domain discretization) will lead to an exaggerated simulation time and hardware resource requirements. Undermesh (too coarse discretization) will lead to inaccurate solution results.

Operation 5: Simplification of a geometry layout is unavoidable when working with complex data files, since drawings aimed for production are not usable for FEM (Finite Element Method) applications. In terms of an economic simulation model, the level of detail of a geometry layout is a defect, causing excessive solution time. Therefore, the simplification of this data as well as reducing the model to a subsection (Operation 4) have been categorized as corrective functions addressing defects caused by the super-system outside the system boundary.

However, if the corrective function in operation 5 is excessive, it could bring about harmful effects which cause disadvantages: Modify the model to the point of not realistic geometries or delete features where relevant physical phenomena should be captured. This defect needs to be addressed by subsequent operations, like e.g., submodelling.

Following the function analysis for processes, the trimming rules for processes can now be used, which will be explained further in the next section. At this point it could already be mentioned that corrective functions should be avoided and can give the software developer hints to check whether this function can be changed or dispensed with. The following case study shows how this can be done.

3.2 Case Study 2: Secure Transfer and Encryption Process

The intention of this case study is to demonstrate the generation of new ideas on all solution levels by using process analysis and trimming rules [17] on pure software problems. For a well-targeted solution search, a much more detailed example would be necessary which goes far beyond the scope of this document.

The "secure transmission" of the data to the smart card means that an attacker must not be able to read or change the transmitted data without being noticed. In the world of smart cards, the data is therefore encrypted (protection against eavesdropping) and provided with a cryptographic checksum (protection against unnoticed changes). This also happens in this example.

It should also be noted that the data transfer to the smart card is still usually done using an APDU (Application Protocol Data Unit) protocol [19]. It is also common to encode the data in so-called TLVs (Type-Length-Value-Format) to save storage space [20]. This graphical version [**Fig. 3**] has been chosen instead of a tabular representation because is in line with the authors way of working and reflects common practice in his field.

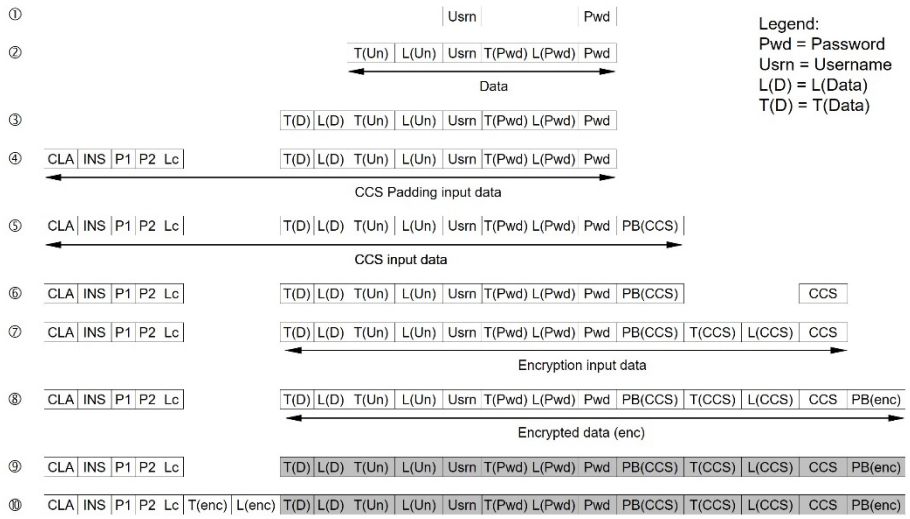

Fig. 3. Exemplary process model for secure transfer to a smart card.

The process shown in **Fig. 3** represents the protection of data transfer to smart cards that is commonly used today. In this example, the aim is to make the data structures used simpler and more efficient. For this purpose, the step-by-step construction of the data structures is considered as a process. The trimming rules for processes are then applied to this process to find approaches and ideas for new solutions. In this paper, only two steps with ideas for solution directions are presented. The complete example can be accessed in [11].

Step 2, supporting function: Add length to each username and password.

- Applied trimming rule (A): The operation requiring the supporting function is trimmed.
  The operation which requires the supporting function is the operation which extracts the username/password on the smart card. Idea: Instead of using username/password, the data of the APDU or parts of it is used as a credential. Since this token consists of username and password, the same method can also be used when comparing username/password with the stored credential.
- Applied trimming rule (B): The supported operation is changed so that it does not require any support now.
  Thus, the data extraction does not need a length anymore. Idea: Username and password have a fixed length, maybe padded with “empty characters”. This also would create great advantages regarding padding for cryptographic checksum and encryption which has a fixed size or can be omitted.
- Applied trimming rule (C): The supported function performs the supporting function itself.
  This means the data extraction method finds out the length by itself. Idea: Username and password length are coded by inserting a special “stop character” at the end instead of a length information.

Applied trimming rule (D): The analyzed supporting function is transferred to the preceding or subsequent operation.
Idea for preceding operation: The length is already part of the username and/or password.
Idea for subsequent operation: The length is encoded as a part of the "tag" byte.

Step 6, supporting function: Calculate the cryptographic checksum and add it to the APDU.

- Applied trimming rule (C): The supported function performs the supporting function itself.
  The supported function is the message integrity check after receiving the APDU in the smart card. Idea: The integrity of the message is checked by checking the integrity of username and password. This can be done by defining a set of valid characters for username and password. If the integrity of the message is attacked, the encryption causes many bytes to be modified in an arbitrary way which can be detected easily.

This case study stops at the point of initial idea generation due to the trimming rules. Of course, during substantiation of those ideas into concepts, project requirements have to be met. This case study did not go further due to confidentiality; however, it already shows substantial potential in trimming for processes applied to software systems.

4 Conclusion and Summary

In this paper we have shown that fundamental TRIZ concepts can be mapped to software systems, even when those are strictly considered "immaterial" or "logical" systems. Modeling concepts like the function analysis for processes, together with function categories and trimming for processes are able to describe and assess software systems and can be used to elaborate on function disadvantages and spark creative ideas for further development. These prerequisites open the door for further use of TRIZ problem solving tools.

Subsequent work could include the following aspects:

- Further evaluation of the model of a complete software system. Retrospective assessment of the development of software systems could underpin or correct the presented model. Observation of the sequences of acquisition of parts from and transfer to the supersystem [17], as well as the analysis of the nature and importance of control components in software systems.
- Function model for processes for software systems. Research about possible additional function categories, which specifically could be useful for categorizing functions in software systems.
- Value analysis based on function analysis for processes. Evaluate the "cost" aspect of single operations inside a software system to be able to assess the functionality vs. cost aspect and derive a ranking for trimming of operations. The question to be answered would be, what could quantify the negative aspect of an operation in a software system? Besides memory requirements and processing time needed for the

given operation, which additional “harmful” parameters can be identified and contrasted with the quantification of the function categories?

In practice TRIZ is already used in software domain. This paper shows a structured mapping of TRIZ tools and concepts to this domain to facilitate adoption while maintaining consistency with the existing theory. The extension of the systematic approach to software systems has been presented, along with two case studies.

5 Definitions

Table 4. Definitions.

Term	Definition
Algorithm	(1) A finite set of well-defined rules for the solution of a problem in a finite number of steps; for example, a complete specification of a sequence of arithmetic operations for evaluating sine x to a given precision. (2) Any sequence of operations for performing a specific task. [22]
Code	(1) In software engineering, computer instructions and data definitions expressed in a programming language or in a form output by an assembler, compiler, or other translator. (2) To express a computer program in a programming language. (3) A character or bit pattern that is assigned a particular meaning; for example, a status code. [22]
Data	(1) A representation of facts, concepts, or instructions in a manner suitable for communication, interpretation, or processing by humans or by automatic means. (2) Sometimes used as a synonym for documentation. [22]
Information	(1) Any communication or representation of knowledge such as facts, data, or opinions in any medium or form, including textual, numerical, graphic, cartographic, narrative, or audiovisual. An instance of an information type. [23] (2) Meaningful interpretation or expression of data. [24]
Information Technology (IT) (Term used equivalently in this paper: IT-System)	Any equipment or interconnected system or subsystem of equipment that is used in the automatic acquisition, storage, manipulation, management, movement, control, display, switching, interchange, transmission, or reception of data or information by the executive agency. The term information technology includes computers, ancillary equipment, software, firmware and similar procedures, services (including support services), and related resources. [25]
Process	(1) A sequence of steps performed for a given purpose; for example, the software development process. (2) An executable unit managed by an operating system scheduler. (3) To perform operations on data. [22]
Software	Computer programs, procedures, and possibly associated documentation and data pertaining to the operation of a computer system. [22]

References

1. Govindarajan, U.H., Sheu, D.D., Mann, D.: Review of Systematic Software Innovation Using TRIZ. Int. J. Systematic Innovation, 5(3), 72-90 (2019).
2. Beckmann, H: Function Analysis integrating Software Applications, TRIZ Future 2013 Proceedings, pp. 473-482.
3. Souchkov, V.: Extended Function Analysis - Overview. Presentation delivered during German Expert Day 2019. Sulzbach-Rosenberg, German. February 2019.
4. Petrov, V.: Using TRIZ tools in IT. TRIZ Developers Summit 2019. https://triz-summit.ru/file.php/id/f304862-file-original.pdf, last accessed 2022/03/19.
5. Padabed, I.: Contradictions and Laws of Evolution in Information Systems. TRIZ Developers Summit 2019. https://triz-summit.ru/file.php/id/f304852-file-original.pdf, last accessed 2022/03/19.
6. Beckmann, H.: Method for Transferring the 40 Inventive Principles to Information Technology and Software, Procedia Engineering 131 (2015) 993 – 1001.
7. Lady, D.: Information Technology System Cookbook: Introducing TrizIT: TRIZ for Information Technology. AIMS Publications 2013. ISBN 978-1478302513.
8. Goethals, F.: 20 Trends in Digital Innovations: A TRIZ-trend-structured overview of new business information technologies and innovative applications. Amazon Digital Services LLC, 2014, ASIN: B00P6GNE88.
9. Mishra, U.: TRIZ Principles for Information Technology. CreateSpace 2010 ISBN 978-818465184.
10. Nakagawa, T.: Software Engineering and TRIZ (1) Structured Programming reviewed with TRIZ, TRIZcon 2005, https://www.osaka-gu.ac.jp/php/nakagawa/TRIZ/eTRIZ/epapers/e2005Papers/eNakaTRIZCON-SE1/eTRIZCON2005-SE-050604.pdf, last accessed 2022/04/25
11. Mann, D. L.: Systematic (Software) Innovation, IFR Press (2008)
12. Ikovenko, S., Przymusiala, M., Yatsunenko, S., Barkan, M. G., Karendal, P., Kobyakov, S., Obojski, J., Vintman, Z.: State-of-the-Art TRIZ, Theory of Inventive Problem Solving. Novismo Ltd. 2019. ISBN 978-83-65899-05-7
13. Chrząszcz, J., Bertoncelli, T., Gronauer, B., Mayer, O., Nähler, H.: "Modeling IT-systems in TRIZ", submitted to TRIZ Future 2022 Conference.
14. Krste Asanovic, CS 152 Computer Architecture and Engineering, https://slidetodoc.com/cs-152-computer-architecture-and-engineering-lecture-1-4/, last accessed 2022/04/24
15. https://electronics.stackexchange.com/questions/353915/what-is-the-role-of-isa-instruction-set-architecture-in-the-comp-arch-abstract, last accessed 2022/04/24
16. M. S. Schmalz, Organization of Computer Systems: §1: Introductory Material, Computer Abstractions, and Technology, https://www.cise.ufl.edu/~mssz/CompOrg/CDAintro.html , , last accessed 2022/04/24
17. Lyubomirskiy, A., Litvin, S., Ikovenko, S., Thurnes, C.M., Adunka, R.: Trends of Engineering System Evolution (TESE) – TRIZ paths to innovation, TRIZ Consulting Group (2018).
18. Ikovenko, S.: Level 3 Training Manual. 2019.
19. https://en.wikipedia.org/wiki/Smart_card_application_protocol_data_unit, last accessed 2022/04/25
20. https://en.wikipedia.org/wiki/Type%E2%80%93length%E2%80%93value, last accessed 2022/04/25
21. https://triz-akademie.de/triz-software/
22. IEEE Standards Board, lEEE Std 610.121990, IEEE Standard Glossary of Software Engineering Terminology, 1990

23. NIST Special Publication 800-30 Revision 1, Guide for Conducting Risk Assessments, National Institute of Standards and Technology, 2012
24. Kissel, R., Regenscheid, A., Scholl, M., Stine, K., NIST Special Publication 800-88 Revision 1, Guidelines for Media Sanitization, National Institute of Standards and Technology, 2014
25. FIPS PUB 200, Minimum Security Requirements for Federal Information and Information Systems, Federal Information Processing Standards Publication, National Institute of Standards and Technology, 2006