



Article

TRIZ for Digital Systems Engineering: New Characteristics and Principles Redefined

Kari Lippert ^{1,*}  and Robert Cloutier ² ¹ Department of Defense USG, Ft. George G. Meade, MD 20755, USA² College of Engineering, University of South Alabama, Mobile, AL 36688, USA

* Correspondence: kari.lippert@gmail.com

† retired.

Received: 26 June 2019; Accepted: 5 August 2019; Published: 11 August 2019



Abstract: While innovation from the systems engineer is desirable at every step in all phases of systems engineering, there must be a methodology to evaluate alternatives. A formal methodology, complete with verification and validation of the results, was developed in 1946 by Soviet engineer Genrikh Saulovich Altshuller and is known as “The theory of inventor’s problem solving”, or TRIZ. This approach improves the way a systems engineer’s thinking progresses about a problem’s solution from “what is” towards “what will be” in the innovative development of a solution. The original distinguishing features of systems used in TRIZ were derived from innovations addressing physical, mechanical system, and few of them apply to digital systems. This paper presents additional characteristics that should be considered in the Reduction phase when applying TRIZ to innovation in digital systems engineering and a redefinition of the principles. With the additions of these distinguishing features for digital systems, TRIZ will become an invaluable tool for the digital systems engineer.

Keywords: digital systems; innovation; systems engineering; TRIZ

1. Introduction

A key goal of systems engineering is to effectively evaluate alternatives throughout the systems engineering process. The systems engineer is responsible for developing design ideas, comparing options, resolving conflicts, and optimizing the performance of the system. Systems engineers have long utilized their diverse backgrounds and skills to harness innovations based on connections [1,2]. There are few ways to systematically explore these mental connections which are where innovation is usually found, and this lack of systemization makes innovative ideas seem like the result of chance or inspiration. While innovation from the systems engineer is desirable at every step in all phases of systems engineering, there must be a methodology and way to evaluate the innovation alternatives.

A formal methodology, complete with a method for verification and validation of the innovation, was developed in 1946 by Soviet engineer Genrikh Saulovich Altshuller. This methodology, “Теория решения изобретательских задач” (Teoriya Resheniya Izobretatelskikh Zadatch), or “The theory of inventor’s problem solving”, is commonly referred to by its acronym TRIZ. The theory offers guides to thinking that identify, isolate, and resolve the problems (contradictions) that prevent an optimal solution. Altshuller refined the application of his theory developing ARIZ (Алгоритм решения изобретательских задач—АРИЗ), an “Algorithm of Inventive Problem Solving”. This approach is a list of about 85 step-by-step procedures to solve complicated invention problems [3]. Designed to shape thinking, ARIZ navigates through the various aspects of the problem space, navigates through the desired solution space, and focuses on the transformations between the two.

Altshuller describes TRIZ as “a methodology that disciplines thinking to stimulate the inspiration that leads to daring solutions to problems” [4]. The thinking about a problem’s solution must begin

with the “what is” of the problem space and move towards the “what will be” of the solution space [5]. Application of the TRIZ methodology provides the bridge between these two; it is the problem space which is characterized and resolved. TRIZ has the reputation as the only systematic problem solving and innovation tool-kit available [6].

The TRIZ tool-kit is comprised of several tools. Each is most effective when applied against a particular type of problem [6] but they can all be used in conjunction with each other. The tools that comprise this methodology include inventive principles, evolution laws, smart little people, ideality, substance-field analysis, flow analysis, feature transfer, standard solutions, separation principles, multiscreen (9-windows), trimming, and contradictions. This work focuses on the tool that used most often: contradictions. This process is very powerful for breaking out of existing design paradigms and entering into new and exciting ones [7].

TRIZ can be considered as a set of all methods necessary for solving an inventive problem from the beginning to the end, and many approaches have been developed combining TRIZ with other methods: Design for Six Sigma (DFSS) [7,8]; the philosophies of Robust Design [9] and Quality Function Deployment [10,11]; Suh’s Axiomatic Design [12]; Design Structure Matrices [13]; and Syntectics [14]. Recently, a study was made of over 200 case studies where TRIZ had been used throughout industry [15]. TRIZ has become a powerful tool in the industrial world, helping companies innovate solutions their customers have not yet specified. At Samsung, for example, “TRIZ is now an obligatory skill set if you want to advance within Samsung” [16].

The generic solution triggers resulting from TRIZ help structure problems and suggest directions for specific solutions. It is important to note that there exists a considerable gap between the generic solution triggers which have been developed through TRIZ and the desired specific solution [17]. TRIZ models are excellent models for reasoning about the problem and the solution, but they cannot replace special knowledge of any given domain. This improvement in reasoning about a system, however, is what strengthens the ability of any systems engineer who includes TRIZ in their toolbox.

Since recent improvements in the field of digital systems engineering have been incremental and anything but daring, TRIZ provides an attractive avenue for innovation in this domain. As TRIZ was originally constructed to describe mechanical, physical systems, there is concern whether the “universal” principles of TRIZ can be applied to digital systems. It clearly applies to the physicality of a digital system (hardware), but its application to the intangible world of bits and bytes (software and data) is not so clear. Analogies of the principles to the world of software have been proposed [18–22] and attempts have been made to integrate the concepts of TRIZ into systems engineering tools and processes [1,23,24]. Concluding that the “39 constraints were not applicable” [2] in their use of TRIZ to innovate IT solutions has called the contradiction matrix into question. As a result, Kasravi calls for an evaluation of the usefulness of TRIZ in the domain of digital information systems.

This paper presents such an evaluation with a brief introduction to TRIZ and its most popular tool, the contradiction matrix, which links system characteristics with system principles. The paper then defines 22 additional characteristics of systems that apply to digital systems and therefore were not a part of the original methodology which was derived from physical/mechanical systems. The corresponding research to extend the contradiction matrix to link the additional characteristics to systems principles is on-going.

2. Background

2.1. The TRIZ Process

As illustrated in Figure 1, TRIZ is a cyclic process composed of two phases: Analysis and Synthesis. The engineer brings the ideal functional model developed in the analysis phase into the synthesis phase to develop an innovative solution that resolves a problem or conflict. Solutions from synthesis are returned to analysis to reformulate any remaining problems. Repeating the cycle with the solution may provide further improvement or even initiate additional innovations.

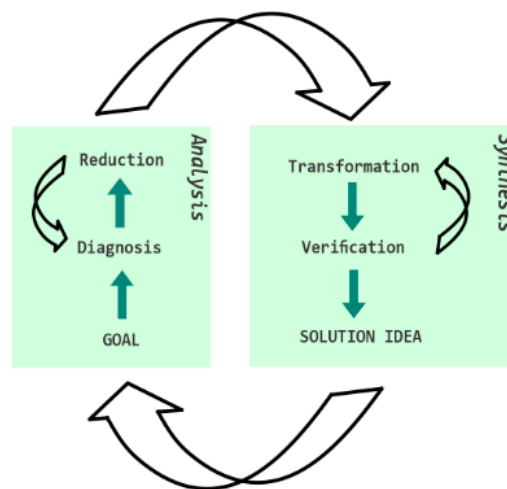


Figure 1. The cyclic process of the theory of inventor’s problem solving (TRIZ).

2.1.1. Diagnosis

The Diagnosis step is that part of the Analysis phase where the goal is defined, and the useful primary function of the system is determined. Analysis of the goal, “what will be”, in comparison with “what is”, leads to the description of the problem in terms of a technical or physical contradiction.

Close and careful consideration of the contradiction enables the definition of the operational zone (OZ). If the OZ is improperly defined, the solution will not be optimal. Proper definition of the OZ can also help the engineer to quickly recognize real possibilities or limitations when generating a solution [25]. While considering the OZ, it is important to apply systems thinking [26] and to consider that in every system all of the parts are connected either directly or indirectly. This perspective helps ensure that the correct contradiction is being resolved. Contradictions should be refined by some method, such as root cause analysis [27], to ensure the cause of the contradiction is real and not just perceived.

As an example, consider the modern cultivation of grapes. Grapevines are quite flexible and require support to keep the grape harvest off of the ground and accessible to harvesters. This support is commonly provided by trellises to which the grapevines are secured. After the harvest, the grapevines are prepared for winter. This involves detaching the vines from the trellis so they may be lain on the ground and covered to protect them from cold weather. In the spring, the covering is removed and the vines are once again attached to the trellises. Diagnosis suggests that the OZ is the interaction of the vines with the stationary trellis and the weather. Time is largest the resource that is consumed along with material to secure the vines.

2.1.2. Reduction

Reduction is the next step of the Analysis phase in which the ideal functional model is constructed. The OZ is described in detail, and the available resources are determined. Additional TRIZ tools can be used in this phase including flow analysis, smart little people, and 9-windows. The result is the contraction to be solved. A contradiction is generally characterized by one, two, or a small number of component features. Each feature can be described as having a plus (or improving) aspect and a minus (or degrading) aspect. Thirty-nine features were defined by Altshuller [4] for physical/mechanical systems. Most of these features also apply to digital systems. Once the systems engineer has identified the improving and degrading features in the OZ, the TRIZ principles which are most appropriate are selected.

Original Features

Several of the features distinguish a moving object from a fixed, or stationary object. As would be expected, fixed objects do not change their position in space independently or as a result of external

forces, while moving objects can. An obstacle to properly utilizing the contradiction matrix is a proper understanding of the features, or characteristics, from the few words provided in their name (see Table 1). Generally accepted definitions are provided in Appendix A for the 39 original features [4,28].

Table 1. Original system characteristics.

Productivity
Universality, Adaptability
Level of automation
Reliability
Precision of manufacture
Precision of measurements
Complexity of construction
Complexity of inspection and measurements
Ease of manufacture
Ease of use
Ease of repair
Loss of information
External damaging factors
Internal damaging factors
Length of the stationary object
Length of the moving object
Surface of the stationary object
Surface of the moving object
Volume of the stationary object
Volume of the moving object
Shape
Speed
Functional time of the stationary object
Functional time of the moving object
Loss of time
Quantity of material
Loss of material
Stability of the object's structure
Force
Tension, pressure
Weight of the moving object
Weight of the stationary object
Temperature
Brightness of the lighting
Power
Energy use of the moving object
Energy use of the fixed object
Loss of Energy

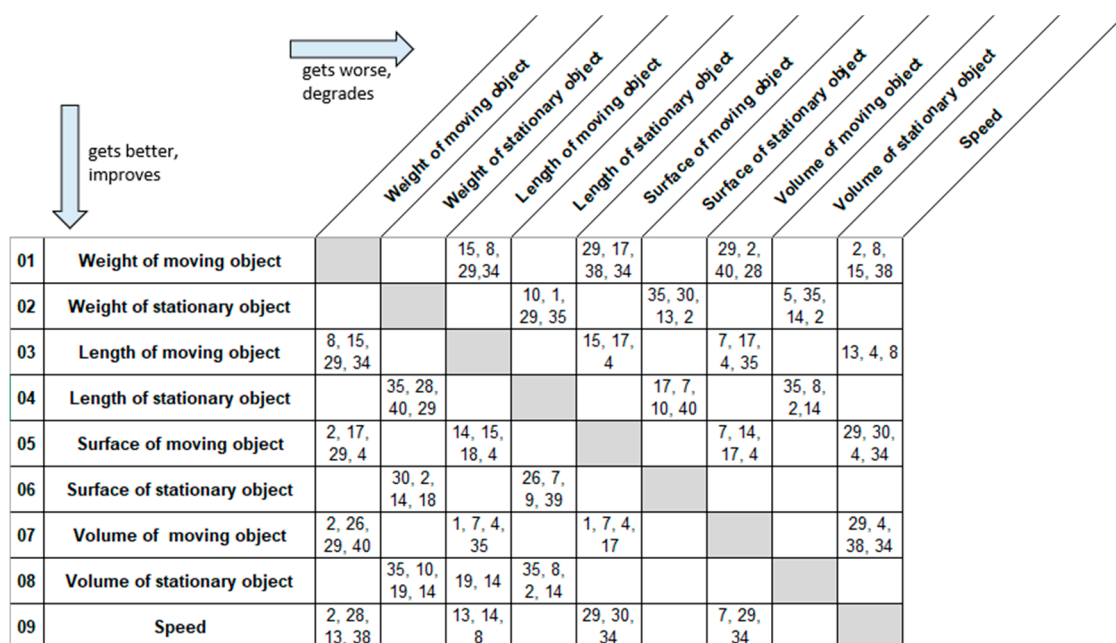
In the example of the grapevines, the ideal final result is a system where the vines are supported by the trellis, the vines are protected in winter, and the vines are not removed from the trellis. Support is inherent through use of the trellis, so that does not introduce a contradiction. Not removing the vines from the trellis and protecting them during cold weather is a contradiction: the upright trellis supporting maximal harvest allows vines to be damaged or killed in cold weather. Consulting the list of original features (below), the following seem to be most applicable to this contradiction:

1. External damaging factors
2. Functional time of stationary object
3. Loss of time

2.1.3. Transformation

In the transformation step of the Synthesis phase, the TRIZ principles that will be applied to create the necessary transformations are selected. This is accomplished through application of any or all the original TRIZ principles that can be applied to the contradiction in order to resolve it [4].

Even though inspiration may happen during the preferred method of evaluating the principles systematically [25], it can be tedious to consider all combinations when only a few are likely to apply. An alternative approach uses the plus and minus characteristics of the technical features in the contradiction to reference a cell in a matrix. A portion of the contradiction matrix is shown in Figure 2. The full matrix is provided as supplemental material. This contradiction matrix limits the principles to be considered in the solution space [5,29]. The matrix is consulted by locating the cell at the intersection of the improving (plus) feature and the degrading (minus) feature. Within the intersecting cell will be a list of TRIZ principles that should be of greatest use in resolution of the contradiction. The matrix effectively reduces the number of principles which must be considered individually. However, it should be used with caution as it can narrow the list of principles prematurely. The solution will use the transformations described by the principles to eliminate the contradictions and to achieve the ideal final state.



		Weight of moving object	Weight of stationary object	Length of moving object	Length of stationary object	Surface of moving object	Surface of stationary object	Volume of moving object	Volume of stationary object	Speed
01	Weight of moving object			15, 8, 29, 34		29, 17, 38, 34		29, 2, 40, 28		2, 8, 15, 38
02	Weight of stationary object			10, 1, 29, 35		35, 30, 13, 2		5, 35, 14, 2		
03	Length of moving object	8, 15, 29, 34				15, 17, 4		7, 17, 4, 35		13, 4, 8
04	Length of stationary object		35, 28, 40, 29			17, 7, 10, 40		35, 8, 2, 14		
05	Surface of moving object	2, 17, 29, 4		14, 15, 18, 4				7, 14, 17, 4		29, 30, 4, 34
06	Surface of stationary object		30, 2, 14, 18		26, 7, 9, 39					
07	Volume of moving object	2, 26, 29, 40		1, 7, 4, 35		1, 7, 4, 17				29, 4, 38, 34
08	Volume of stationary object		35, 10, 19, 14	19, 14	35, 8, 2, 14					
09	Speed	2, 28, 13, 38		13, 14, 8		29, 30, 34		7, 29, 34		

Figure 2. A portion of the contradiction matrix.

In the example, there were three contradictions. These contradictions are examined to determine which improves as the other degrades. This examination indicates that the external damaging factor (the weather) gets worse as the functional time of the stationary object (grapevines on the trellis)

increases. Using these two system characteristics in the matrix (shown in Figure 3), the following principles are suggested:

		29	30	31	32
14	Strength	3, 27	18, 35, 37, 1	15, 35, 22, 2	11, 3, 10, 32
15	Functional time of moving object	3, 27, 16, 40	22, 15, 33, 28	21, 39, 16, 22	27, 1, 4
16	Functional time of stationary object		17, 1, 40, 33	22	35, 10
17	Temperature	24	22, 33, 35, 2	22, 35, 2, 24	26, 27
18	Brightness of the lighting	3, 32	15, 19	35, 19, 32, 39	19, 35, 28, 26

Figure 3. Intersection of external damaging factors and functional time of stationary object in the contradiction matrix. 17—Transition into another dimension, 1—Segmentation, 40—Composite materials, 33—Homogeneity.

The order of the principles is taken from the matrix and is the suggested order of consideration. Transition into another dimension (17) could be achieved through alteration of the position of the vines; they move from horizontal to vertical and back again, but the trellises do not. The solution might lie in figuring out a means by which this could happen. Segmentation of the trellises would allow portions of the trellis to move independently of one another. Composite materials suggest a change to the structure and composition of the trellis that would enable the transition into the other dimension; homogeneity is generally the case with trellises and would not be an improvement on the system. This leads to a (once) novel solution of a hinged trellis. The base of the trellis can remain in the ground, but in winter, the hinges can be released, and the flexible vines laid on the ground to be covered as if they were unattached from the trellis. This solution will eliminate the wasted resources of time and materials involved with the attaching and detaching of the vines each year.

2.1.4. Verification

Verification is the step where the solution is assessed in terms of its advantages and disadvantages. It focuses on the ideal final result (IDF) and the functional ideal model. The problem state of the system is disregarded in all but one of the verification rules. The rules that are used to verify the innovation as a solution are as follows:

1. The rule of the resolution of the contradiction: The solution must solve the contradiction, or it is not a solution.
2. The rule of the benefits of super-effects: The positive effects of the solution must outweigh the negative effects.
3. The rule of durability: The solution must have the option for future improvement, development or reuse based on a potential change in the operational zone.

4. The rule of the background check: An innovative solution must really be new. An existing solution is an acceptable solution to the contradiction but cannot be considered an innovation.
5. The rule of the methods check: The methods used in the solution must be new to be an innovation.

In our example, our solution resolves the contradiction by allowing the time of the functional time of the stationary object (grapevines on the trellis) to be maximized while enabling the minimization of the external damaging factor (the weather). Positive effects of the solution reduce the consumption of two resources; time and securing materials. The solution is durable in that there is the possibility for improvement of the materials in use and the way the vines are covered. A background check was not done on this solution as it is a description of an existing innovation provided solely as illustration. The methods used in the solution, a hinged trellis, is (was) new as a trellis was previously considered to be a solid, fixed object. These verifications indicate that this solution was, at some point, an innovation.

3. Methods

3.1. Application of TRIZ to Data and Information Processing Systems

The systems being built today are larger, more complex, and much less focused than their predecessors. These information processing and analysis systems are deliberately and carefully designed by system engineers and implemented to meet known and specified needs about known information and environments. Current systems engineering practice requires knowledge in advance about the information a system will process to construct the data models which then become the foundation of the system. Increasingly, data requirements are less well defined, and the data itself is growing in complexity. Future information processing systems must be capable of embracing change since their complexity is driven by both multidisciplinary scope and the increasing complexity of the information which they process. This represents a significant contradiction in systems engineering: the data model, which is the foundation of the system design, is fragile.

The initial application of TRIZ to the problem of fragile data models indicated that the application of modularization, a principle of agility, results in a modular data model. It is reasonably simple to envision a data model made from interchangeable components, much like Lego™ or even an Erector Set™. The need for standardization in the building blocks has been recognized since the late 1970s [30]. Properly constructed, the building blocks become quantum units and can be reused in multiple places. In a proactive environment, these building blocks and models can be prepared prior to their specified need. In a reactive environment they are tailored and assembled as needed. Combinations of the building blocks can be virtually unlimited—observe the richness of computer information exchange. The inability to validate the modular data model solution indicated that this was not an ideal solution so the use of the TRIZ process was repeated to improve the modular data model.

The ideal final result is an adaptive data model is created, managed, maintained, and utilized by the agents and the users alike. It combines the concepts of patterns [31] and metaphor [32–34] to enable the construction of a representation of a digital ecosystem. It discusses the development of a digital ecology, a foundation for the interconnected, interrelated, and interdependent digital species that manage data and information. It is easiest to envision as a multiagent system [35] where the ecological species are agents who mediate with brokers on behalf of users to find any desired information. The agent populations in this envisioned digital ecology would coevolve with new data appearing in the environment and new user requirements and age off in a manner determined by dictate or by lack of use.

3.2. Digital System Features

During the characterization of the operational zone, it became evident that several system characteristics, or features, that would be critical to the design could not be expressed through the original features given by Altshuller. Many of the original characteristics pertain to digital systems and have resulted in the successful application of TRIZ to modern engineering. However, several features that are commonly considered during system engineering trade-off studies for digital systems

are not a part of the original TRIZ matrix. The following are proposed as part of a useful and necessary extension of the characteristics, or features, of systems.

Availability

The simplest representation of availability is as a ratio of the expected value (E) of the uptime of a system to the aggregate of the expected values of up and down time:

$$A = E_{\text{uptime}} / (E_{\text{uptime}} + E_{\text{downtime}}) \quad (1)$$

Classification of availability is based on the span of time to which the availability refers and to the type of downtime used. Common classifications include instantaneous, mean, steady state, inherent, achieved, and operational.

Complexity of control

Complexity is determined by the number of integrated factors being dealt with simultaneously. When applied to the control of a system, it is the number and interrelations of the functions of a system that adjusts its operations as required.

Capacity

The highest rate at which information can be reliably transmitted over a communication channel is a measure of its capacity.

Efficiency

Efficiency is a measurable concept, quantitatively determined by the ratio of useful output to total input. It indicates the system's ability to avoid waste of materials, energy, and time in operation.

Resource consumption

Resource intensity is a measure of the quantity of resources (e.g., time, energy, materials) needed for the completion of a process or activity.

Elegance

In general, elegance is unusual effectiveness, simplicity, and consistency of design. In engineering, a solution may be considered elegant if it uses a nonobvious method to produce a solution which is highly effective and simple. An elegant solution may solve multiple problems at once, especially problems not thought to be inter-related.

Security

Security is freedom from, or resilience against, potential harm (or other unwanted coercive change) from external forces. It refers to protection from hostile forces, but it has a wide range of application. Security also includes secrecy and containment of information when applied to digital system.

Scalability

Scalability a measure of the capability of a system, network, or process to handle a growing amount of work, or its potential to be enlarged to accommodate that growth.

Number of dependencies

In software engineering, coupling is the degree of interdependence between software modules. It is a measure of how closely connected two routines or modules are and indicates the strength of the relationships between modules. In digital systems, the number of dependencies can be measured as physical interdependence or as coupling. Low coupling is often a sign of a well-structured computer system and a good design.

Precision of output

Precision of output in a digital system is a measure of the numerical precision required, as higher precision requires more capacity in communication, storage, and display.

Precision of operation

Precision in operations is a measure of reproducibility. Variance can be introduced into systems through random numbers, varying levels of precision of calculations, and precision of inputs.

Latency

The time delay experienced by a system during data communication is its latency.

Fidelity of data

Fidelity refers to the degree to which a model or simulation reproduces the state and behavior of a real-world object, feature or condition. Fidelity is therefore a measure of the realism of a model or simulation, outputs of digital systems.

Throughput

In a digital system, the throughput is the maximum rate at which data can be processed.

Data rate

The speed with which data can be transmitted from one device to another. Data rates are often measured in megabits (million bits) or megabytes (million bytes) per second.

Mean time to repair

Mean time to repair (MTTR) is a basic measure of the maintainability of repairable items. It represents the average time required to repair a failed component or device. Expressed mathematically, the MTTR formula is the total maintenance time divided by the total number of maintenance actions over a specific period.

$$\text{MTTR} = \text{total maintenance time/number of repairs} \quad (2)$$

Mean time between failure

Mean time between failures (MTBF) is the predicted elapsed time between inherent failures of a mechanical or electronic system during normal system operation. MTBF can be calculated as the average time between failures of a system. The term is used for repairable systems, while mean time to failure (MTTF) denotes the expected time to failure for a nonrepairable system. The definition of MTBF depends on the definition of what is considered a failure for the system.

Useful life

Useful life is the estimated lifespan of any component of a digital system. Useful life may be impacted by dependencies between system components.

Memory capacity

The memory capacity is the maximum or minimum amount of memory a computer or hardware device can have, or the required amount of memory required to run a process or program.

Memory efficiency

This metric represents how efficiently the memory subsystem is used by the digital system. It is a component of algorithmic efficiency which can be thought of as analogous to engineering productivity for a repeating or continuous process.

Digital size

The size of a digital system can be measured in several ways but is often expressed as a measure of how much storage it consumes. Typically, storage size is expressed in units of measurement based on the byte.

Ecological impact

Ecological impact is the effect of system activities on living organisms and their nonliving environment. Digital systems have ecological impact through their power and cooling needs, as well as their component production and disposal.

3.3. Definition of the 40 System Principles Extended for Digital Systems

Each of the principles is defined below. While not recommended, this list can be consulted directly without use of the matrix. It is a source of innovative thought when used in this fashion but can be very time consuming. If using the matrix, the number preceding the principle corresponds with the contradiction matrix.

The definition of each of the principles has been extended to include a brief discussion of how this principle is manifest in digital systems. This research did not find a need to extend the list. Work is on-going in the extension of the contradiction matrix to accommodate these new features and relate them to the 40 principles.

Segmentation

Segmentation is the process of dividing a system into parts. Such a division permits the isolation of system properties. The divisions applied can be real or artificial, psychological, or physical. Isolation of the harmful properties into an individual segment allows such properties to be considered independently of the rest of the system. In addition, segmentation can alter system properties and introduce new properties.

Dividing an object into independent parts results in segmentation. Division of a means of transport into smaller and smaller independent parts (used to improve air flow) is an innovation that has been applied in improving the cooling of electronics [36]. Segmentation has already been implemented within computer systems in transitioning from mainframes to personal computers. In software systems, this segmentation has been proposed to be the division into intelligent agents [18]. In keeping with the true sense of division of parts, the application of segmentation to programming is done by object-oriented methods, or functional modularity. This type of segmentation is used in systems engineering to achieve decoupling between systems, to implement service-oriented architecture (SOA), and to develop a work breakdown structure.

Making an object sectional—easy to take apart and put together—was an innovation which resulted in “ready to assemble” furniture. When applied to software and systems, this approach invokes the principles of modular design. A third approach applies when an object is already segmented and requires increasing the degree of segmentation. In systems engineering, this is manifest by the definition of increasing levels of sub system definition.

Extraction (Separation)

In contrast to segmentation, extraction involves the removal of things from the system. Extraction requires the separation of useful or harmful parts or properties from the whole system. This allows what is needed, or conversely what is not needed, to be taken from the system. This can be achieved by extracting the cause of the undesirable property or function. In a computer system, extraction is represented in the identification and removal of bugs and in the correction of system failures. Similarly, extraction of the disturbing part or property from an object is seen in the development of an X-ray shield that has cut-outs in the shape of the lungs to better protect the parts of the body that do not need to be exposed. In a computer, the removal of heat is an extraction. In process improvement,

extraction is achieved by the application of Lean principles [37]. Extracting only the necessary part or property from an object is exemplified by the isolation of a signal from a noisy transmission; or, in the case of computer systems, the extraction of user interface components (monitors and keyboards) from the processing unit. Extraction can enable reuse and can be formalized through the development of patterns. It implies a refinement of the system.

Local Quality

Changing the characteristics of something in a specific area (location) to gain the desired functionality will, in turn, increase the quality in that area. Optimum functions can be achieved by varying interactions over time for various characteristics of the system. Improving local quality can improve efficiency. Such improvement can result from the transition of a homogeneous to a heterogeneous structure for the object or its environment. This may result in the use of a gradient as that being created to produce multistate memory [38] which can replace binary memory for computer systems. Another recent transition from a homogeneous structure to a heterogeneous one is in the development of distributed computing platforms such as the cloud.

Local quality can also be achieved by having different parts of an object carry out different functions. Also referred to as specialization, this is observed in service-oriented architecture (SOA). In the mechanical world, this type of specialization may be exemplified by a pencil with an eraser at one end. To achieve optimum function, each part must be placed in conditions that favor it. Systems engineering is working to achieve local quality of systems through localization techniques. One such system that has automated this principle is self-positioning solar panels that can track the movement of the sun and reposition to optimize their energy producing ability.

Asymmetry

Asymmetry is the process of transitioning from isotropic to anisotropic or vice-versa. That is, a property invariant with respect to direction of measurement will transition to a property that differs according to the direction of measurement. Since engineers seem to have an unconscious bias towards symmetrical design [4], changing symmetry will often result in new innovation.

Making something asymmetrical can be challenging, but it is accomplished in many solutions such as dynamic load balancing and resource allocation. In manufacturing, asymmetry in parts reduces errors in assembly [39]. This exemplifies changing the shape of the object or system to suit external asymmetries. In computer and systems science, customization could result in this type of change.

Consolidation

Consolidation is the bringing together of functions, characteristics, parts of a system, or even separate systems into a relationship. Consolidation can create desirable, new or unique properties or outcomes. By consolidating homogeneous objects, processes, or objects destined for contiguous operations, a more precise workflow can result. These principles can be extended into data flows and other material and energy flows within a system. While consolidation implies colocation, it can also be done by making time homogeneous or making operations contiguous. This is demonstrated in the world of computer systems through synchronized processes, parallelization, and integration.

Universality

Universality addresses making a system more uniform and more comprehensive in its function. This universality can refer to a feature, an action or a condition of the system over space and/or time as well as the use of an object for different purposes. Additionally, universality can refer to the same requirement or use of different objects for the same purpose or result. To achieve this result, a system can be made more dynamic, with interchangeable elements, or the system can use adjustable features.

Patterns are derived by the recognition of universality. Allowing customization, patterns form the foundation for reuse. Objects can be adjusted by removing system elements, thereby allowing a single

object to perform several different functions. In systems engineering, this is manifest in localization and multifunctional resources.

Matrioshka (Nested dolls)

Nesting is the quality of being made to fit closely, to fit together, or to fit one inside of another. Nesting optimizes space utilization and can be used to protect objects. Subassemblies can exhibit this quality. In computer science, this is most commonly referred to as encapsulation, in which data and functions are protected from “outside” influence. Wrappers are another common artifact of computer data processing systems. These wrappers allow ease of movement of information through a system that might reject the native form. Such wrappers are common in digital communications. In computer systems, nesting of objects is manifest in programming through multiple layers of abstraction and inheritance as well as programming in layered computer network architecture such as the OSI model. Modulation, where a message signal is conveyed inside another signal that can be physically transmitted, is a type of nesting.

Another perspective of nesting is that of an object passing through a void in another object, such as is common in a camera’s zoom lens. In computer network communication, the use of tunneling is an example of this type of nesting. Steganography, the hiding of information within digital images, can also be considered a special form of matrioshka.

Counterbalance (Counter-Weight)

The act of counterbalancing is compensation, or opposition, to bring into balance an environmental force. This can be achieved through a more uniform distribution, or it can be achieved by use of a contrary action or force to alleviate an effect. Floating is enabled by weight being countered by hydrodynamic forces; flying is enabled by aerodynamic lift. The encouragement with this principle is to examine the environment to find existing opposing forces that can be applied.

In systems engineering, this principle is manifest in distributed architectures where a more uniform distribution of resources is evident. Other environmental forces that require balancing in digital systems include power, space, cooling, bandwidth constraints, and network latency.

Preliminary Counteraction

Preliminary counteraction involves determining what might go wrong in advance and taking a counter action to eliminate, reduce, or prevent the occurrence of the undesirable effect. Counteraction can apply equally to functions, events, and conditions.

In systems engineering, determining what might go wrong is the foundation for risk management and for potential failure modes analysis. These analyses examine known failures, invent failures to expose weak points in a design, expose potentially dangerous conditions, and show the effect of (unanticipated) human interaction and exposure of the system to different sources of energy.

Several solutions seen in risk mitigation are applications of this principle. Throttling is a mechanism that can control a flow which could overwhelm the receiving system. RAID storage divides and duplicates information stored on a computer disk because of the assumption that there will be failures on the disk. Error-handling routines are programmed into systems anticipating user and system errors. Fail-overs are designed into systems as protection from component failure. These steps are all examples of preliminary counteraction.

Preliminary Action

This principle should not be confused with preliminary counteraction that requires an opposing action. The principle of preliminary action refers to the performance of an action earlier in time than the occurrence of another event. Preliminary action can be used to increase performance and safety, as well as to support correct actions and outcomes.

One way to implement this principle is by performing required changes to an object completely or partially in advance. This is an anticipatory application; as such, it is preventative maintenance.

In computer systems, preliminary action can be implemented as preprocessing, prefetching, and caching of information. Another means of implementation is to place an object in a position that is in advance of the object's need, so it can go into action from the most convenient location. Preliminary action is a critical part of lean manufacturing and process improvement and applies to any process that uses materials or information.

Cushion in Advance

Critical to this principal is the recognition that every system has some capacity for unreliability which cannot be eliminated. Cushioning the system in advance is a means to deal with types of unreliability that cannot be eliminated. In digital systems that includes uninterruptible power supplies to cushion against loss of power in critical systems, scheduled backups to safeguard data, timed releases to prevent blocking in resource access, as well as input handlers that eliminate error [40].

Equipotentiality

There are three concepts that make up this principle and they can be used in combination or separately:

- create uniform potential to achieve a system benefit.
- create associations with the system to support equal potential.
- establish contiguous relationships and associations.

The goal of this principal is to conduct the process spending a minimum amount of energy on auxiliary actions or functions. This is an integral part of lean manufacturing: eliminating undesirable or wasteful action. The environment, the structure of the object and the system can all be used in implementing this principle. Ultimately, the condition of the work is changed so that it will not require unnecessary movement such as lifting and lowering.

Digital systems have a few analogies to this principle. Networks are established to enable different pathways between nodes which then help to establish a uniform potential. Load balancing can help enforce that uniformity. Within a computer, memory swapping helps create uniformity in the access of information.

Inverse Action (Do It in Reverse)

This principle is focused on thinking of the opposite. In place of a direct action, the opposite action is implemented. For example, a moveable part may be made stationary or vice versa; or the object can be turned upside down or inside out. It is important to understand that there are two possibilities for reverse—one where there is nothing, and one where the opposite is taken. For example, action vs. no action is not the same as action vs. anti-action. A common manifestation of this principle is a treadmill where the walking person does not advance but instead the “ground” does so. In computer systems, sorting is approached in this fashion in databases where the indices can be inverted for faster search time.

Spherical-Shape (Spheroidality)

Spheroidality is achieved by replacing linear attributes with curved or spherical ones. This can be a change from linear to nonlinear, or it can be a change in a coordinate system (Cartesian to polar). Feedback can create a spherical path in a system.

Data input to computers has followed this principle and has moved from cards and linear tape to circular disks. Basic structuring of networks and information also show a shift from more linear arrangements (e.g., hierarchical) to more circular ones (e.g., graphs). Computer program processing has also gone from linear (procedural) code to nonlinear (event oriented). Circular buffering also demonstrates this principle.

Dynamization

This principle focuses on change and adaptability. A system, state, or property is made ephemeral, temporary, moveable, adaptive, flexible, or changeable. It is also possible to make similar changes in the environment as opposed to the object. The focus is to change characteristics of the object or environment to achieve optimal performance or action. One means of implementing this principle is to make the stationary and moveable dimensions interchangeable. It is also possible to divide the object into parts that can change their position relative to one another.

Computers have enabled a great deal of dynamism by moving print to digital media. Most recently, programs utilize dynamic load libraries and have dynamic cache, dynamic menus, and dynamic scaling. Dynamic scaling is one property of the principle of timesharing found in cloud computing.

Partial or Excessive Effect or Action

This principle deals with cases where 100% action is difficult or impossible to perform. Partial action results when fewer actions or less of an action is taken than would be required, and the results are dealt with. Excessive action results when more actions than are required are taken and the results are dealt with. The use of a different form, or a combination, of energy can be used in this principle. Approximations and estimations are examples of partial action. Perturbation analysis utilizes excessive action to change the state of a system.

Transition into Another Dimension

All dimensions are considered by this principle. Vertical to horizontal, straight or perpendicular to diagonal, two dimensional to three dimensional or to four dimensional are all transitions into new dimensions. These transitions can be applied multiple times and in either (or all) directions.

Computer memory caching can transition from a permanent (disk) to a temporary (RAM) dimension. Format changes, such as binary to hex, can also be considered dimensional changes. Disruptive technologies, developed outside of the normal technology strategy, can be in a different dimension. This change in dimension allows their undetected maturation. Computer-based visualization allows transition from two dimensions to three or four and beyond into augmented and virtual reality.

Mechanical Oscillations

This principle is concerned with the use of vibration or oscillation to involve a regular, periodic variation in value about a mean. Mechanical vibration requires understanding that a stable system is not always best and recognizes that there is a place for controlled instability or variation. Scheduling, queue handling, and randomness in systems demonstrate this principle. Full and complete utilization of the electromagnetic spectrum also utilizes this principle.

Periodic Action

Continuous actions can be made periodic by changing how the action is done. Variation can be introduced by changing the frequency or amplitude of the periodic action. Additionally, both uniform and random patterns can be applied to the frequency and amplitude to alter the periodicity of the action.

Some major examples of this principle are: burst-mode transmissions, batch processing, and image projection. In computer screens, the refresh frequency can be altered to suit the viewer's requirements. In transmission of images, layers of the image are transmitted through bursts of data transmission providing an increasingly sharper image.

Uninterrupted Useful Action

Having a constant, continuous flow can eliminate system idleness and sometimes intermediate actions. This principle avoids starting and stopping an action as well as the elimination of nondynamic parts of a flow. It leads to balanced utilization and is also a part of lean manufacturing. In computer science, continuity of useful action can be extrapolated to the adage "don't fix what isn't broken" with

regards to upgrading systems. A system with continuous, useful action should remain that way after being updated. On the other hand, the continuous action might not be of the system, but of the user of the system which would be the case with software. Hardware implementations provide continuous action by avoiding system restarts and by employing continuous refreshing and failover.

Rushing Through

This principle is useful when overcoming something that goes wrong at one given speed. The speed of the action can be increased to reduce or eliminate the bad effects. This is why running over hot coals does less damage to the soles of the feet than walking over them would.

There is no analogy to this in the digital systems domain.

Transform Damage into Use

The conversion of harm into benefit for a system requires a change in attitude. The system must be examined for wasted materials, energy, information, functions, space, time, etc. to find the harmful factors. Once identified, they can be dealt with. Combining two harmful factors can eliminate both; increasing the degree of harm can sometimes eliminate it completely. Often the waste can be converted into something that can be of use to another part of the object or system [41]. The conversion of waste heat to power using a turbine is an example of this principle.

Feedback

Feedback is some output of a system that is used as a control mechanism for that system. This output can be a useful or a harmful output. If a system is already using feedback, the frequency or amplitude of the feedback can be altered, or new sources of feedback can be sought. A common application of this principle is a gas pump nozzle which senses the increasing pressure inside the tank to stop the flow of gasoline. In systems engineering, feedback is used as an improvement driver and incremental development of systems relies heavily on feedback.

Mediator

A mediator negotiates a temporary link between incompatible parties, functions, or conditions. It uses an intermediate carrier, a blocker or a process, and any of these can easily be removed once the link is no longer needed. Mediators exist at interfaces, and they can manifest themselves in many ways. In system and software engineering, interface engineering accomplishes mediation using wrappers, mappings, and middleware. Dedicated devices such as print servers also serve as mediators between computers and a printer and routers serve as mediators on a network.

Self Service

This principle refers to the ability of an object or system to process related functions in conjunction with its primary function. The object services itself, repairs itself, or carries out supplementary useful actions. One common application of self-service is a self-healing cutting surface. The self-service can be the result of physical, chemical, geometric, spatial, or temporal arrangement of actions or objects. It can overlap with feedback for timing and often makes use of excess or wasted material and energy.

A common computer example is the automated spelling and grammar checking (in a word processor) which is executed while the user is typing. Automatic detection and treatment for viruses and worms is also a type of self-service performed by a computer. Programs self-service through autoupdating, and clocks can self-synchronize. Computer viruses and worms are a special case of self-service through self-replication.

Copying

A copy is a replica of something that is used when the original itself is too valuable to be used. Utilizing a copy can allow adjustments in scale and material. Copies can be abstractions, images, dimensional models, or simulations. Computer emulators implement this principle by providing a

copy of an operating system or environment. For example, approximately 10% of the Vatican library holdings have been digitized so that scholars may “examine” minute details on computer generated images of the originals. (According to the Vatican librarian, the ability to magnify any area of the digitized image has revealed an amazing quantity of information for scholars who would not otherwise be allowed to handle the original materials.) Computer viruses create multiple copies to achieve a broader distribution than would be possible without it.

Inexpensive Short-Life Object (Disposable)

Cheaper, simpler, or more disposable objects can be used to reduce cost, to increase convenience and to improve the service life of an object. Disposability often requires a change from complication to simplicity or the replacement of a high cost material with something of lower cost. Implementing this principle requires a tradeoff of one or more properties. A common example is that of temporary staffing or computer timesharing. System and software engineering uses this principle in the development of prototypes that can help improve the actual system being created.

Replacement of Mechanical System

This principle deals with the substitution of interactions, devices, mechanisms, and systems with physical fields or other forms, actions, and states. These substitutions change or replace the principle of operation. In systems where there is a human in the loop, the mechanical system can be replaced with a sensory system. A good example of an implementation of this principle is the development of voice recognition capabilities that are replacing the keyboard as an interface to a computer. The replacement of mechanical systems with digital systems, such as the replacement of the punched paper tape reader with a computer file system, has been taking place for many years.

Pneumatic or Hydraulic Construction

System properties that can be considered for replacement include compressibility, flow, turbulence, springiness, and energy absorption. This principle replaces a solid part of an object with gas or liquid (air or water commonly). Packing materials utilize this principle in the form of adjustable air bags and moldable forms. Data and information can be compressed for more rapid transmission. The energy absorption property of water is superior to that of air, and water has replaced air to cool microelectronics especially in large computers and data centers [10].

Flexible Covers or Thin Films

Flexible membranes, covers, or thin films can replace traditional materials and isolate components, objects, or systems from their environment. These membranes can be layered to achieve the protection desired. In computer systems, middleware can be considered a thin film that separates the algorithms and display from the databases. Wrappers (for functions or data) can also be considered as thin films or flexible membranes. Graphical user interfaces are another example where the separation is between the “guts” of a program and the end user.

Porous Material

Materials (gasses, solids, and liquids) can be made more porous by creating voids. Such voids change the condition of the materials and increase their surface area. This principle can be utilized to increase air flow or achieve filtering. It should not be limited just to materials as it can be applied equally as well to space, time, fields, functions, and information. In digital transmissions, protocols often contain “reserved” fields that are unused (or empty) space. In the physical construction of a system, empty space is left for room to expand the system.

Changing the Color

A change in color of a system can enhance value or make it easier to detect change. It can also be used to improve measurement or improve visibility. In nature at least, the ability to change color

can insure survival [8]. As well, objects (or the environment) can be made translucent with this principle. State change or status can easily be represented by alteration of color, translucency, and or size. Graphical user interfaces for computers make excellent use of this principle.

Homogeneity

Homogeneity states that if two or more objects or substances interact, they should consist of the same material, energy, or information. These interactions can be through action, features, functions or objects. Creating homogeneity eliminates translation and transposition. Marginal homogeneity can also be employed where there are inconsequential differences between the objects or substances. In the digital domain, this principle can be implemented by combining functions that access the same data into a single application. In information science, this principle can be implemented through abstraction, and can also refer to the creation of a unified data model. In systems design, homogeneity is achieved by having all the components provided by the same source thereby eliminating the need for integration.

Rejecting and Regeneration (Discard and Renewal) of Parts

Rejecting is removing, discarding, dissolving, or evaporating, a part of an object or system that is no longer needed. Regeneration is the repair or restoration of a part so that it might be reused. An example of rejecting is a biodegradable packaging material. In computer programming, the process of garbage collection is a regeneration of computer memory. In databases, a transaction rollback implements both parts of this principle. In network communications, packet headers are handled according to this principle.

Transformation of Properties

Transforming or changing a property can accommodate useful benefits. The transformation can be to a physical state, a change in concentration or density, or a change in the degree of flexibility. Additionally, changes in geometry or configuration can be transformations. The use of parameters to dynamically customize a computer program supports this principle. In analytics, transformations such as the Fourier transform are used to decompose functions into other functions.

Phase Transition

Materials that change phase (gas to liquid, liquid to solid, solid to gas) can be used to implement an effect on an object or system. Phase transitions are accommodated by alteration of other properties of the material, alterations which include volume changes and altered heat capacity which can be captured for benefit to the system. A change from physical to digital can also be considered a phase transition and is exemplified by the transition from postal mail to electronic mail.

Thermal Expansion

This principle uses the conversion of heat energy into mechanical energy. In systems where there is a heat-induced change, the heat can be used to benefit the system, or it can be negated. The principle can be extended to other phenomena such as gravity, light, and pressure.

Strong Oxidants (Accelerated Oxidation)

The various states of oxygen can be leveraged as a catalyst for change, improving an action or function. Catalysts can be used to accelerate or decelerate an action and support optimization of functions. Application to the digital domain requires focusing on the catalytic function. An example is the format in which information is held. The format can obfuscate or facilitate use and understanding of the information. Visualization is another catalyst for understanding. Optimization of functions can be achieved in several ways in the digital environment.

Inert Media (Environment)

An inert environment is a neutral atmosphere or environment that supports a desired function. Implementation of this principle requires first the determination of what it is which is preventing the desired function and then requires the creation of the inert environment to protect or enable that function. A clean room is a good example of an inert environment. A test environment (harness) for system testing and a virtual sandbox are both examples of inert environments. In operational environments, this would be the removal of extraneous electromagnetic emissions from the environment of the system. This is often achieved through shielding.

Composite Materials

This principle is the opposite of homogeneity. It is the replacement of homogeneous materials with composites. Such a replacement often results in layering. The emerging process of mash-ups in the digital domain is an implementation of this principle. In systems engineering, the use of commodity hardware from a variety of sources to create a system creates a composite.

4. Discussion

This work proposes extensions to the work originally presented by Altshuller in 1946. Altshuller's work addressed an approach for inventive reasoning for mechanical/physical systems. The extensions presented here extend his work to incorporate characteristics that are applicable to modern day digital systems.

The TRIZ innovation methodology can be useful when confronted with any contradiction or problem by providing a systematic way to explore a solution space. Applied to trade studies, TRIZ can result in more robust compromise solutions [1]. As has been shown by engineers in many companies, including SC Johnson, Proctor and Gamble, Motorola, Hewlett Packard, GE, Unilever, and Dow Chemical, application of TRIZ principles to problem spaces results in a more effective solution [2,15,16,42]. At the very least, the effort the system engineer applies to describing and characterizing the contradiction and operational zone of the system will provide significant insight and, in turn, improve any system design.

Systems engineers, using these digital system characteristics, can leverage the TRIZ methodology to:

1. Help better define the problem;
2. Clarify resources and constraints;
3. Allow reduction of the contradiction and solution into a system of interactions;
4. Help better understand the system of interactions;
5. Balance tradeoffs that will be necessary;
6. Help delineate the benefits of the solution;
7. Refine the contradictions to ensure the right problem is being solved; and
8. Refine the solution (which itself presents new contradictions).

With the extensions proposed herein to consider distinguishing features for digital systems, TRIZ could become an invaluable tool for the digital systems engineer.

Supplementary Materials: TRIZ contradiction matrix is available online at <http://www.mdpi.com/2079-8954/7/3/39/s1>.

Author Contributions: Conceptualization, K.L.; Formal analysis, K.L.; Methodology, K.L.; Software, K.L.; Supervision, R.C.; Writing—original draft, K.L.; Writing—review & editing, K.L. and R.C.

Funding: This research received no external funding.

Conflicts of Interest: The authors declare no conflict of interest.

Appendix A. Definitions of the Original System Characteristics (Features)

Productivity

The output of a system per unit time or per unit of input. The effectiveness of productive effort.

Universality, adaptability

The extent to which a system/object is appropriate for all situations or positively responds to external changes.

Level of automation

The extent to which a system or object performs its functions without human assistance. At the highest level of operation, the system or object can determine what is needed, adapt to that need, and monitor its function.

Reliability

A system's ability to perform its intended functions in consistent, predictable ways.

Precision of manufacture

The extent to which the actual characteristics of the system or object match the specified or required characteristics.

Precision of measurements

The degree to which the result of a measurement of a property of a system conforms to the correct value or a standard.

Complexity of construction

The number and diversity of elements and element interrelationships comprising a system. Human interaction increases complexity.

Complexity of inspection and measurements

Demonstrated by measuring or monitoring systems that are complex, costly, require much time and labor to set up and use, or that have complex relationships between components or components that interfere with each other. Can also be indicated by the increasing cost of measuring to a satisfactory error.

Ease of manufacture

The degree of facility or effortlessness in manufacturing or fabricating the object/system.

Ease of use

Simple, easy to understand and perform or utilize; accessible.

Ease of repair

The degree of facility or effortlessness in the mending or fixing of the system. Encompasses convenience as well as time to repair faults, failures, or defects.

Loss of information

Partial or complete, permanent or temporary, loss of data or access to data in or by a system. Frequently includes sensory data such as aroma, texture, etc.

External damaging factors

Susceptibility of a system to externally generated (harmful) effects. A harmful effect is one that reduces the efficiency or quality of the functioning of the object or system.

Internal damaging factors

Harmful effects are generated by the object or system, as part of its operation.

Length of the stationary object

Any one linear dimension of the fixed or stationary object, not necessarily the longest.

Length of the moving object

Any one linear dimension of the moving object, not necessarily the longest.

Surface of the stationary object

The outermost or uppermost layer of the stationary object.

Surface of the moving object

The outermost or uppermost layer of the moving object.

Volume of the stationary object

The cubic measure of space occupied by the stationary object.

Volume of the moving object

The cubic measure of space occupied by the moving object.

Shape

The external contours, form, or appearance of a system.

Speed

The rate of a process or action, or the rate of movement of an object.

Functional time of the stationary object

The time that the stationary object or system can correctly perform its action. Also referred to as service life or durability.

Functional time of the moving object

The time that the moving object or system can correctly perform its action. Also referred to as service life or durability.

Loss of time

Time taken to perform the activity or function of the system or object. Improving the loss of time means reducing the time taken for the activity.

Quantity of material

The number or amount of a system's materials, substances, parts, or subsystems which are altered (permanently or temporarily) or consumed (fully or partially) during normal functioning.

Loss of material

Waste of a system's materials, substances, parts, or subsystems during normal functioning.

Strength

The extent to which the object can resist changing in response to force. Resistance to breaking.

Stability of the object's structure

The standalone nature and immobility of the system or object.

Force

Force measures the interaction between systems and is measured as the strength or energy of physical action or movement. It is any interaction that changes an object's condition or environment.

Tension, pressure

Technically, force per unit area but can also be a balance between, and interplay of, opposing objects or systems.

Weight of the moving object

The mass of the object, in a gravitational field. The force that the body exerts on its support or suspension.

Weight of the stationary object

The mass of the object, in a gravitational field. The force that the body exerts on the surface on which it rests.

Temperature

The thermal condition of the object or system. Includes other thermal parameters, such as heat capacity, that affect the rate of change of temperature.

Brightness of the lighting

Illumination characteristics of the system such as brightness, intensity, and color.

Power

Energy that is produced by mechanical, electrical, or other means. This characteristic refers to production not consumption.

Energy use of the moving object

Consumption of energy resource(s) in order to perform the action or function of the moving object.

Energy use of the fixed object

Consumption of energy resource(s) in order to perform the action or function of the stationary object.

Loss of Energy

The waste of energy resource(s) through the function or movement of the object or system.

References

1. Blackburn, T.D.; Mazzuchi, T.A.; Sarkani, S. Using a triz framework for systems engineering trade studies systems engineering. *Syst. Eng.* **2012**, *15*, 355–367. [CrossRef]
2. Kasravi, K. Applications of TRIZ to IT: Cases and Lessons Learned. In *TRIZCON 2010*; The Altshuller Institute for TRIZ Studies: Dayton, OH, USA, 2010.
3. Altshuller, G.S. Algorithm of Inventive Problem Solving (ARIZ85c). Available online: <http://www.evolutus.com/Textbooks/ariz85c.pdf> (accessed on 8 August 2019).
4. Altshuller, G. *40 Principles Extended Edition: TRIZ Keys to Technical Innovation*; Technical Innovation Center Inc.: Worcester, MA, USA, 2005.
5. Orloff, M.A. *Inventive Thinking Through TRIZ: A Practical Guide*, 2nd ed.; Springer: Heidelberg, Germany, 2006; p. 351.
6. Czinki, A.; Hentschel, C. Solving Complex Problems and TRIZ. In *TRIZ Future 2015 (TFC 2015)*; Elsevier: Berlin, Germany, 2016; pp. 27–32.

7. Yang, K.; El-Haik, B. *Design for Six Sigma: A Roadmap for Product Development*; McGraw-Hill Professional: New York, NY, USA, 2003; p. 624.
8. Polovina, S. An Introduction to Conceptual Graphs. In *Proceedings of the 15th International Conference on Conceptual Structures (ICCS 2007)*; Springer: Sheffield, UK, 2007.
9. Phadke, M.S. Introduction to Robust Design (Taguchi Method). Available online: iSixSigma.com (accessed on 11 March 2002).
10. Chan, L.-K.; Wu, M.-L. Quality function deployment: A literature review. *Eur. J. Oper. Res.* **2002**, *143*, 463–497. [[CrossRef](#)]
11. Akao, Y. Development History of Quality Function Deployment. In *The Customer Driven Approach to Quality Planning and Deployment*; Asian Productivity Organization: Tokyo, Japan, 1994.
12. Jack, H. *Engineer on a Disk*; 2001. Available online: <http://engineeronadisk.com/V3/index.html> (accessed on 8 August 2019).
13. Eppinger, S.D.; Browning, T.R. *Design Structure Matrix: Methods and Applications*; MIT Press: Cambridge, MA, USA, 2012.
14. Gordon, W.J.J. *Synectics: The Development of Creative Capacity*; MacMillan Publishing Company: New York, NY, USA, 1968.
15. Spreafico, C.; Russon, D. TRIZ industrial case studies: A critical survey. In *TRIZ Future 2015 (TFC 2015)*; Elsevier: Berlin, Germany, 2016; pp. 51–56.
16. Shaughnessy, H.A. *What Makes Samsung Such an Innovative Company?* Forbes: Jersey City, NJ, USA, 2013.
17. Mann, D. Emergent Contradictions: A Synthesis of Triz and Complex Systems Theory. In *2ND IFIP TC-5 Working Conference on CAI*; Springer: Brighton, MI, USA, 2007.
18. Rea, K.C. TRIZ and Software-40 Principles Analogies, Part I. *TRIZ J.* **2001**, *9*, 1–8.
19. Rea, K.C. TRIZ and Software-40 Principles Analogies, Part II. *TRIZ J.* **2001**, *11*, 1–7.
20. Fullbright, R. TRIZ and Software Fini. *TRIZ J.* **2004**. Available online: <https://triz-journal.com/triz-software-fini/> (accessed on 8 August 2019).
21. Mishra, U. The Revised 40 principles for Software Inventions. *SSRN Electron. J.* **2005**. [[CrossRef](#)]
22. Van den Tillaart, R. TRIZ and Software-40 Principle Analogies, a sequel. *TRIZ J.* **2006**. Available online: <https://triz-journal.com/triz-software-40-principle-analogies-sequel/> (accessed on 8 August 2019).
23. Ahmed, S.S. Rapid Innovation with TRIZ—Part 2. *TRIZ J.* **2005**. Available online: <https://triz-journal.com/rapid-innovation-triz-part-2-using-triz-symbols-schematics-continuous-innovation/> (accessed on 8 August 2019).
24. Bonnema, G.M. Insight, innovation, and the big picture in system design. *Syst. Eng.* **2011**, *14*, 223–238. [[CrossRef](#)]
25. Filmore, P. Teaching TRIZ: Breaking Mindsets. *TRIZ J.* **2008**. Available online: <https://triz-journal.com/key-to-teaching-triz-breaking-mindsets/> (accessed on 8 August 2019).
26. Boardman, J.; Sauser, B. *Systems Thinking: Coping with 21st Century Problems*; Industrial Innovation Series; Badiru, A.B., Ed.; Taylor and Francis: Abingdon, Oxford, UK, 2008.
27. Andersen, B.; Fagerhaug, T. *Root Cause Analysis: Simplified Tools and Techniques*, 2nd ed.; American Society for Quality, Quality Press: Milwaukee, WI, USA, 2006.
28. Domb, E.; Miller, J.; MacGran, E.; Slocum, M. The 39 Features of Altshuller's Contradiction Matrix. 2007. Available online: <http://www.eng.uwaterloo.ca/~ljzelek/teaching/syde361/TRIZ39.pdf> (accessed on 9 January 2019).
29. Domb, E. TRIZ Matrix/40 Principles/TRIZ Contradictions Table. 2007. Available online: <http://triz40.com/> (accessed on 9 May 2008).
30. Sowa, J.F. *Knowledge Representation: Logical, Philosophical, and Computational Foundations*; Brooks/Cole: New York, NY, USA, 2000.
31. Alexander, C.; Ishikawa, S.; Silverstein, M. *A Pattern Language: Towns, Buildings, Construction*; Oxford University Press: New York, NY, USA, 1977; Volume 2, p. 1171.
32. Lakoff, G.; Johnson, M. *Metaphors We Live by*; University of Chicago Press: Chicago, IL, USA, 1980.
33. Lakoff, G. *Women, Fire, and Dangerous Things*; University of Chicago Press: Chicago, IL, USA, 1990.
34. Pylyshyn, Z.W. *Things and Places: How the Mind Connects with the World*; Massachusetts Institute of Technology Press: Cambridge, MA, USA, 2007.
35. Jeffery, K.G. GRIDS and Ambient Computing. In *NODE 2002 Web-and Database-Related Workshops*; Springer: Erfurt, Germany, 2002.

36. Venere, E. Nanowick at Heart of New System to Cool 'Power Electronics'. 2010. Available online: <http://www.sciencedaily.com/releases/2010/07/100722153634.htm> (accessed on 26 July 2010).
37. Womack, J.P.; Jones, D.T. *Lean Thinking: Banish Waste and Create Wealth in Your Corporation*, 2nd ed.; Free Press: Glencoe, IL, USA, 2003.
38. Mullner, P.; Knowlton, W.B. *Multi-State Memory and Multi-Functional Devices Comprising Magnetoplastic or Magnetoelastic Materials*; U.S.P. Office, Ed.; Boise State University: Boise, ID, USA, 2007; p. 26.
39. Creveling, C.M.; Slutsky, J.; Antis, D. *Design for Six Sigma: In Technology Product Development*; Pearson Education Inc.: Upper Saddle Rive, NJ, USA, 2003.
40. Saenz, A. BlindType, Virtual Keyboard of the Future. Available online: <http://singularityhub.com/2010/07/29/exclusive-i-used-blindtype-virtual-keyboard-of-the-future/> (accessed on 3 August 2010).
41. Cichanowicz, E.J.S. Simultaneously Providing Renewable Power and Cooling for Data Center Operation. U.S. Patent 20100024445A1, 4 Fenurary 2010.
42. Nagappan, A.; Shahrul, K.; Ishak, A.A.; Yeoh, T.S. Evolution of Triz Application for Manufacturing Industries. *Appl. Mech. Mater.* **2013**, *330*, 760–767. [[CrossRef](#)]



© 2019 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license (<http://creativecommons.org/licenses/by/4.0/>).