July, 2006

The Revised 40 Principles for Software Inventions

Umakant Mishra
The Revised 40 Principles for Software Inventions*

By- Umakant Mishra, Bangalore, India
http://umakantm.blogspot.in

Introduction

Applying “40 Inventive Principles” is one of the earliest and most fundamental technique of TRIZ. There may be controversies on the application of other TRIZ tools, but all TRIZ schools and professionals are anonymous on the usefulness and application of Inventive Principles.

The beauty of Inventive Principles is that they can be independently applied. They are easy to learn and easy to apply, but can solve difficult problems. Many TRIZ students start leaning with Inventive Principles. My learning on TRIZ also started with Inventive Principles.

Background problem

- Although the 40 principles are fundamentally sound there is some difficulty in applying them in areas like computer software. As they were originally developed for mechanical or technical problems, the meanings of the words in many cases do not go well with software environment.

- There are a few principles such as “Phase Transition”, “Thermal Expansion”, “Strong Oxidizers” etc. which, although have very clear meaning in physical world, have no relevance in a software environment.

- Some principles like “Parameter Change” are clearly applicable to software problems. But the description of their applications such as “change physical state”, “change temperature” etc. are confusing in software environment. This is because the “parameters” in software context are different from parameters used in physical objects.

- Many researchers have found that 40 principles are very relevant and useful for software environment. Some have tried to present software examples for the principles. But all of them face difficulties to explain the specific principles and applications mentioned above.

* This paper was first published in Jul 2006 issue of TRIZsite Journal
- The other problem faced is that there are many documents on Inventive principles for engineering field and each of them has some incompatibility in software context. There is no document on “software principles” which can be used as single point of reference by the software inventors.

There is a need to review the 40 Principles in the context of software industry and rephrase the application of some principles to avoid confusion while applying them in a software environment. There is a need to build a single reference document to be followed by software inventors.

**Objectives of this work**

- One of the objectives is to review all technical principles in the context of software industry in order to find any inconsistency applying therein.

- Another objective to modify the principles that do not make sense in software context.

- Yet another objective is to modify or rewrite the application of the principles in the context of software environment.

- Another objective is to prepare a single reference document for “software principles”.

- The most important objective is to achieve all the above without changing the concept, structure and approach of original 40 principles developed by G. Altshuller.

**Methodology**

- The work is based on the existing documents on 40 principles. As I don’t understand Russian sources, I took the English sources of 40 principles as the base. The three main sources I referred are the following.


  2. “TRIZ: The Right Solution at the Right Time”, by Yuri Salamatov, Chapter on “List of Inventive Principles”, pp 222-225; Published by Insyte B.V, 1999

  3. “Hands on Systematic Innovation” by Darrell Mann, Chapter on “Inventive Principles”, pp 203-218; Published by CREAX press, 2002

- I studied more than a thousand software patents during last few years and tested the relevance of each of the Inventive principles with regard to software inventions.
- Tried to find out the Principles and their applications that do not make sense in a software context and tried to rephrase them without changing their approach.

- Tried to apply the 40 principles in the context of software inventions in several articles (in TRIZsite Journal and the forthcoming book “TRIZ Principles for Information Technology”)

- Differences in “Proper Names” of Principles are ignored as long as they are not misleading in a software environment. For example, Principle-2 is known as “Extraction” and “Taking out”. Similarly Principle-13 is called “Reversing” and also “Other way round”. All these alternative names are considered fine and accepted.

The changes made in Principles for software:

As mentioned above, the objective of this work is NOT to write a fresh set of Principles for software inventions, NOR to defy the applicability of existing 40 principles to software inventions. Rather the objective is to tune the words of existing 40 principles in the context of software industry so that it can be used as a reference by future researchers working on TRIZ for software. The summary of changes made to the 40 principles are as follows.

- The names of 3 principles are changed. Principle-36 is changed from “Phase Transition” to “Conversion and Migration”, Principle-37 is changed from “Thermal Expansion” to “Expansion”, and Principle-38 changed from “Strong Oxidants” to “Improve Quality”.

- In some cases (e.g., principle-28: Mechanics Substitution, Principle-31: Porous materials etc.) the applications (descriptions) widely vary in different reference documents. In those cases the applications (descriptions) are tailored to be meaningful in software context without changing the original concept behind the principle.

- In few cases, the narrations of applications are changed. The applications using terms like “aerodynamic”, “hydrodynamic”, “piezoelectric”, “ultrasonic”, “infrared”, “ultraviolet”, “temperature”, “liquid”, “gas” etc. are either removed or altered to use parallel terms used in software industry.

- In many cases the term “object” is replaced with “object or system” to make the meaning comfortable in software context as the term “object” has a very specific meaning in software.

- Besides the above, some minor changes are made in the descriptions to aggregate the differences found between the reference documents and to keep the meaning suitable in a software context.
The revised 40 Principles for Software Inventions

The revised 40 principles for software inventions are as follows:

1. Principle-1: Segmentation (or Division)
   1.1 Divide a system or object into separate independent parts or sections
   1.2 Make a system easy to put together and take apart
   1.3 Increase the degree of fragmentation or segmentation

   Note: No changes made to this principle or its applications.

2. Principle-2: Taking out (or Taking away, or Extraction)
   2.1 Take out an undesired part or function of the object.
   2.2 Take out the cause or carrier of an undesired property or function

   Note: No changes made to this principle or its applications.

3. Principle-3: Local Quality (or Localized characteristics)
   3.1 Change the structure of an object or system from uniform to non-uniform.
   3.2 Change the external environment of the system from uniform to non-uniform.
   3.3 If an object or system has multiple functions, change specific parts of the object to achieve efficiency in the corresponding function.
   3.4 If an object or system has multiple functions, change specific parts of the object or system to make suitable for specific local conditions.

   Note: No changes made to this principle. The applications are rephrased to clearly distinguish the four different application environments.

4. Principle-4: Asymmetry
   4.1 If an object or system is symmetrical make it asymmetrical, or introduce lines of asymmetries.
   4.2 If an object is already asymmetrical, increase the degree of asymmetry.
   4.3 Change the shape of an object or system to suite external asymmetries (e.g. ergonomic features).

   Note: No changes made to this principle or its applications.
5. **Principle-5: Merging** (or combining, or consolidating)

5.1 Merge in space- Physically join or merge identical or related objects or operations to work together in space.

5.2 Merge in time- Join or merge objects, operations or functions so that they act contiguously or parallel in time.

5.3 Merge both in space and time- Join or merge objects or functions to act together in both space and time.

Note: No changes made to this principle. Only the last application is modified to include “merging in both space and time”.

6. **Principle-6: Universality**

6.1 Make a new object or system that performs multiple functions, thereby eliminating the need for multiple existing systems.

Note: No changes made to this principle or its applications.

7. **Principle-7: Nesting** (or Nested Doll)

7.1 Put one object or system inside another

7.2 Allow one object or system to pass through an appropriate hole in another

7.3 Increase the number of nesting, or Use a stacking or cascading arrangement

Note: No changes made to this principle or its applications.

8. **Principle-8: Counterweight** (or Anti-weight)

8.1 Where the weight of an object or system causes problems, combine it with something that provides lift

8.2 Where the weight of an object or system causes problems, make it interact with the environment to get lift

Note: No changes made to this principle or applications in essence. However, the description of the applications are modified to eliminate the terms like “aerodynamic”, “hydrodynamic” etc. to avoid confusions in software environment.

9. **Principle-9: Prior Counteraction** (or Preliminary anti-action)

9.1 Where the action causes stresses or harmful effects, introduce anti-actions and beforehand stresses to control the harmful effects

Note: No changes made to this principle or its applications.
10. Principle-10: Prior Action (or Preliminary Action)

10.1 Perform a useful action (either fully or partially) before it is needed.
10.2 Pre-arrange objects or systems such that they can come into action at the most convenient time and place.

Note: No changes made to this principle or its applications.

11. Principle-11: Cushioning

11.1 Introduce emergency backups to compensate for the potentially low reliability of an object.

Note: No changes made to this principle or its applications.

12. Principle-12: Equipotentiality

12.1 If an object or system needs to be lowered or raised, redesign the object's environment to bring balance and compensate that need.

Note: No changes made to this principle or its applications.

13. Principle-13: Other Way Round (or Reverse)

13.1 Use an opposite or inverse action to solve the problem.
13.2 Make movable objects fixed, and fixed objects movable.
13.3 Turn the object, system or process 'upside down'.

Note: No changes made to this principle or its applications.

14. Principle-14: Curvature (or Spheroidality)

14.1 Change linear parts to curved parts, flat surfaces to spherical surfaces.
14.2 Use rollers, balls, spirals, and domes.
14.3 Change linear motion to rotary motion.

Note: No changes made to this principle or its applications.

15. Principle-15: Dynamize (or Dynamics or Dynamicity)

15.1 If an object or system is rigid or inflexible make it movable or flexible.
15.2 Split an object or system into parts those are capable of moving relative to each other.
15.3 Make a system or object to be adaptive (by changing its functionality) to achieve optimal performance under different operational conditions.

Note: No changes made to this principle or its applications.
16. Principle-16: Partial or Excessive action

16.1 If it is not possible to do the precise action or achieve the precise result, try to achieve 'slightly less' or 'slightly more' of the action or result.

16.2 If an action is crucial and partial action may lead to problems, better do more or excessive.

16.3 If an action is difficult or expensive, and can be easily done in future, better do it less or partial.

Note: No changes made to this principle. Application 2 and 3 are modified a bit.

17. Principle-17: Another Dimension

17.1 If an object moves in a straight line or plane, consider moving outside the line or plane.

17.2 Use "another side" of a given object or system

17.3 Use “another orientation” or “layout” of a given object or system

Note: No changes made to this principle or its applications.

18. Principle-18: Vibration (or Mechanical Vibration)

18.1 Cause an object to oscillate or vibrate

18.2 Increase or change the frequency of vibration

18.3 Make use of an object or system's resonant frequency

Note: No changes made to this principle. Applications 1 to 3 are in tact. Application 4 and 5 are eliminated as the terms "piezoelectric", "ultrasonic" etc. are confusing with software.


19.1 Replace continuous actions with periodic or pulsating actions

19.2 If an action is already periodic, change the frequency to suit external conditions.

19.3 Use the gaps between actions to perform other useful actions

Note: No changes made to this principle or its applications.

20. Principle-20: Continuity of Useful Action

20.1 Make all parts of an object or system work continuously with full efficiency

Note: No changes made to this principle. Only application 2 ("eliminate idle running") is removed.
21. **Principle-21: Skipping** (or Hurry or Rush through)

21.1 Conduct an action at very high speed to eliminate harmful side effects.

21.2 Skip unwanted steps to gain speed.

Note: No changes made to this principle. Only application-2 added, as there are many such cases in software development and business processes.

22. **Principle-22: Blessings in Disguise** (*Turn Lemons into Lemonade*, or *Turn harm into benefit*)

22.1 Use harmful factors or objects or actions to deliver positive effect.

Note: No changes made to this principle. Only application-2 removed.

23. **Principle-23: Feedback**

23.1 Introduce feedback to improve a process or action.

23.2 If feedback is already used, make it adaptable to variations in operating conditions.

Note: No changes made to this principle or its applications.

24. **Principle-24: Intermediary** (or Mediator)

24.1 Introduce an intermediary between two objects, systems or actions.

24.2 Introduce a temporary intermediary that disappears (or can be easily removed) after it has completed its function.

Note: No changes made to this principle or its applications.

25. **Principle-25: Self Service**

25.1 Enable an object or system to serve itself by performing installation, configuration, repair and maintenance operations itself.

25.2 Make use of waste materials or unused resources.

Note: No changes made to this principle or its applications.

26. **Principle-26: Copying**

26.1 Use simple and inexpensive copies in place of unavailable, expensive, complicated, dangerous or possibly vulnerable objects and systems.

26.2 Replace an object, or action with an optical copy, infrared or ultraviolet copies.

26.3 Use virtual copies or simulations of an object or system or function.

Note: No changes made to this principle. Only application 2 and 3 are combined, as there is no relevance of infrared and ultraviolet copies in software. A new application “virtual copies or simulations” is added instead which is very significant in software environment.
27. **Principle-27: Cheap short-living objects** (or Cheap and disposable, or Cheap short-life instead of costly long-life)

27.1 Replace an expensive object or system with a multitude of inexpensive, short-living objects

Note: No changes made to this principle or its applications.

28. **Principle-28: Mechanics Substitution** (or Another sense, or Replacement of Mechanical system)

28.1 Replace a mechanical means with a sensory (optical, acoustic, taste, touch or smell) or other (magnetic, electromagnetic, thermal etc) means.

28.2 Change the internal mechanism, structure, format or code.

Note: No changes made to this principle. However the applications are found to be widely different in the reference documents. Well, they are revised to suite software environments.

29. **Principle-29: Pneumatics and Hydraulics**

29.1 Use gases and liquids instead of solid parts or systems

Note: No changes made to this principle or its applications. There may not be any relevance of physical gas or liquid in software, but they have relevance with computer hardware. Besides liquid can be taken in the meaning of lightweight or free-flowing.

30. **Principle-30: Thin and Flexible** (or Flexible shells and thin films)

30.1 Incorporate flexible shells and thin films instead of solid structures.

30.2 Use flexible and think films to isolate an object or system from its environment.

Note: No changes made to this principle or its applications.

31. **Principle-31: Hole** (or Porous materials)

31.1 Make an object porous or add porous elements

31.2 If an object is already porous, add something useful into the pores

Note: No changes made to this principle or its applications.

32. **Principle-32: Color Changes** (or Changing color)

32.1 Change the color of an object, its part or its external environment, which can change the visibility of the object.

32.2 Change the transparency of an object or its surroundings

Note: No changes made to this principle or its applications.
33. **Principle-33: Homogeneity** (or Compatibility)

33.1 Make interacting objects of the same material (or material with matching properties)

Note: No changes made to this principle or its applications.

34. **Principle-34: Discard and Recover** (or Reject and regenerate parts)

34.1 Discard the elements of an object or system which have fulfilled their functions

34.2 Restore exhausted or consumable parts of an object or system during operation

Note: No changes made to this principle or its applications.

35. **Principle-35: Parameter Change**

35.1 Change parameters an Object or System

Note: No changes made to this principle. All the applications dealing with different technical parameters (such as temperature etc.) are put together into one application. This avoids confusion, as software parameters are different from physical object parameters.

36. **Principle-36: Conversion** (or Conversion and Migration)

36.1 Convert the old incompatible data, file, format or technology to a new compatible data, file, format or technology.

36.2 Migrate from old incompatible data, format or technology to compatible data, format or technology.

Note: The principle “Phase Transition” is changed to “Conversion and Migration”. This is because the phases like “liquid”, “gas” etc. has no meaning in software. On the other hand “conversion and migration” is a widely used activity in software. Conversion (say from one format to another) in software environment has conceptual similarity with phase transition (say from liquid to gas) in physical environment.

37. **Principle-37: Expansion**

37.1 Use expansion (or compression) of an object or component to achieve useful effect

Note: This principle is changed from “thermal expansion” to “expansion”, as the term “thermal” has no real significance in software. The term “expansion” alone is meaningful and significant in software context.
38. **Principle-38: Improve Quality**

38.1 Replace an object or system with an updated or advanced object or system

38.2 Add features or functions to improve or update an object or system

Note: The principle “Strong Oxidants” is changed to “Improve Quality”, as the terms “air”, “oxygen” etc. has no relevance to software. However, the internal meaning of the principle, i.e., “replacing x with enriched x”, remains in tact.

39. **Principle-39: Calm** (or Inert Atmosphere)

39.1 Replace a normal environment with an inert one

39.2 Add neutral parts, or inert elements to an object or system

Note: No changes made to this principle or its applications.

40. **Principle-40: Composite** (or Composite Materials)

40.1 Use composite (multiple) materials instead of uniform ones.

Note: No changes made to this principle or its applications.

**Reference:**


3. Darrell Mann, “Hands on Systematic Innovation”, Published by CREAX press, 2002


About the author  
After working for more than 18 years in various fields of Information Technology Umakant is currently doing independent research on TRIZ and IT since 2004. He last worked as Director and Chief Technology Officer (2000-2004) in CREAX Information Technologies (Bangalore). Before that he worked as IS/IT manager (1996-2000) for ActionAid India (Bangalore).

Umakant is a Master in Philosophy (MA), Master in Business Administration (MBA), Bachelor in Law and Logic (LLB), Microsoft Certified Systems Engineer (MCSE+I), Certified Novel Engineer (CNE), Master Certified Novell Engineer (MCNE), Certified Intranet Manager (CIM), Certified Internet Professional (CIP), Certified Software Test Manager (CSTM) and holds many other global IT certifications.