A Method of Resolving Differences Based on the Concepts of Functions and Process Objects Part 2

Takahara Toshio ( )

Abstract

This article improves the framework of the method of "resolving differences" which consists of problem solving, making new function and idealization based on the concepts of "function" and "process object". These three types of "resolving differences" have a unified input-output relation. We set the purpose in terms of Object. Then we find out input of object to get this purpose of object for every kind of logical types of changing Objects.

1. Introduction

To recognize something in the real world is simply to recognize elements of something and the relations between them. To change something in the real world is simply to change elements of something and/or the relation between them.
The requirements on the ideal theory of recognition and changing of the world are to handle every "object" to be recognized and to operate it in every possible types of changing "object" in every applying area including technology and institution.

After showing some basic concepts around object and the outline of the previous paper, this paper shows that problem solving, making new function and idealization have a unified input-output relation.

2. Object [1]

2.1 Object

Object is something to be recognized which itself coincide with common sense. My opinion is that Object consist of System Object and Process Object corresponding to "being" and "movement" (or "action") respectively. The target to be reached which is to be recognized and controlled is the subset of this Object. Operations on these objects to change the real world should cover all types of operation.
I can recognize being that is matter and "idea" and movement.

1) **Matter (Being):** System Object
2) **"Idea" (Being):** System Object
 21) Information of individual or common notion which is born by physical entity
 e.g.: Information on document
 22) My idea
3) **Movement or Action：** Process Object

System Object consists of system sub-objects and process sub-objects and in the same way Process Object consists of system sub-objects and process sub-objects.

2.2 Granularity, Attributes and Structure of Object

Attributes is content of Object with specific description. **Attributes of upper level object is total attributes and structure of the object**. Object has two types of total attribute. One is "**attribute**" in narrow sense which is not easy to change and "**state**" which is easy to change.

One of the most important attributes of Object is its granularity. Roughly speaking **granularity** is scope or sphere in space and time and degree of abstraction (Density). Granularity specifies the area in space and the range in time domain. Designating proper granularity is always one of the most difficult problems in every issue.
Element of lower level object can become Object of next hierarchy or granularity as in the next Figure.

For example,
1) A shape of bathtub is an attribute and temperature of the bathtub is a state.
2) The maximum speed of a car is one of the attributes of System Object of the car.
3) On the contrary the speed of this running car is a state of Process Object of car's running.
Process Object has also attributes and states because Process Object is Object.

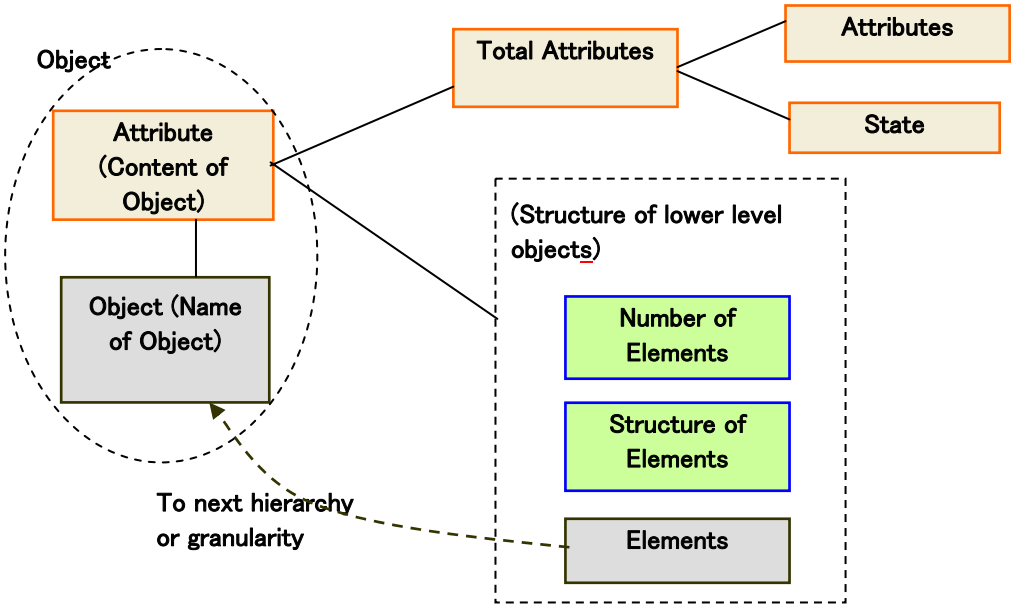

Figure 1. Structure of Object

2.3 Differences and Function

We have differences

- between attributes of a Object and that of another Object,
- between attributes of a Object and absence of another Object which are the attributes themselves or

- between an Object and its absence which is Object itself.

Or we have differences

- between attributes of a Object at a time and that of another time,
- between attributes of a Object at a time and absence of another time which are the attributes themselves or
- between an Object at a time and its absence of another time which is Object itself.

Function is fundamentally positive meaning of Process Object. [2]
This function is divided in two types. One is **fundamental function** or **process function** which is positive meaning of Process Object itself. And the other is **subsidiary function** or **attribute function** which is positive meaning of an attribute or a state of Object which consist of System Object and Process Object. And of course making a System to help realize these functions contribute to a kind of function.
Functions form hierarchy toward upper level function.

2.4 How to Express Cause- Effect Diagram

Object and their relations make Object world. In this paper many figures and tables are used to express Object world. In the case of causal relation we use cause- effect diagram. One of typical formats of cause- effect diagram is shown as follows.

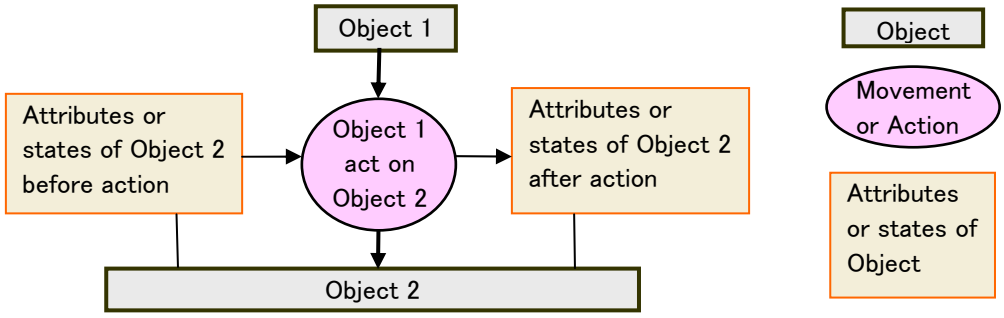

Figure 2. One of Typical Formats of Cause- Effect Diagram

Following Figure 3. 4. 5 are some examples of cause- effect diagram using this format.

In next chapter, Figure 6. 7. 8. are written neglecting their attributes or states. Figure 9 and also Table.3 which have the same contents as Figure 9 in next chapter is the accumulation of cause- effect diagrams expressing every possibility of the cause- effects. It is important that Object 1 and Object 2 are not necessarily restricted to System Object in these Figures and Tables.

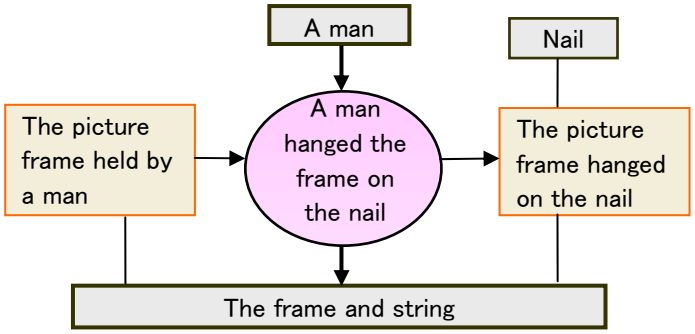

Figure 3. An Example of Cause- Effect- Diagram: Picture Hanging

Some rules in cause- effect diagram with links and an example of expressing cause- effect diagram with links are as follows.

Some rules in cause- effect diagram with links:
1) Two cause- effect diagrams can be linked via same Object, same attributes or same states.
2) An Object, attribute or state should appear in a cause- effect diagram only once.

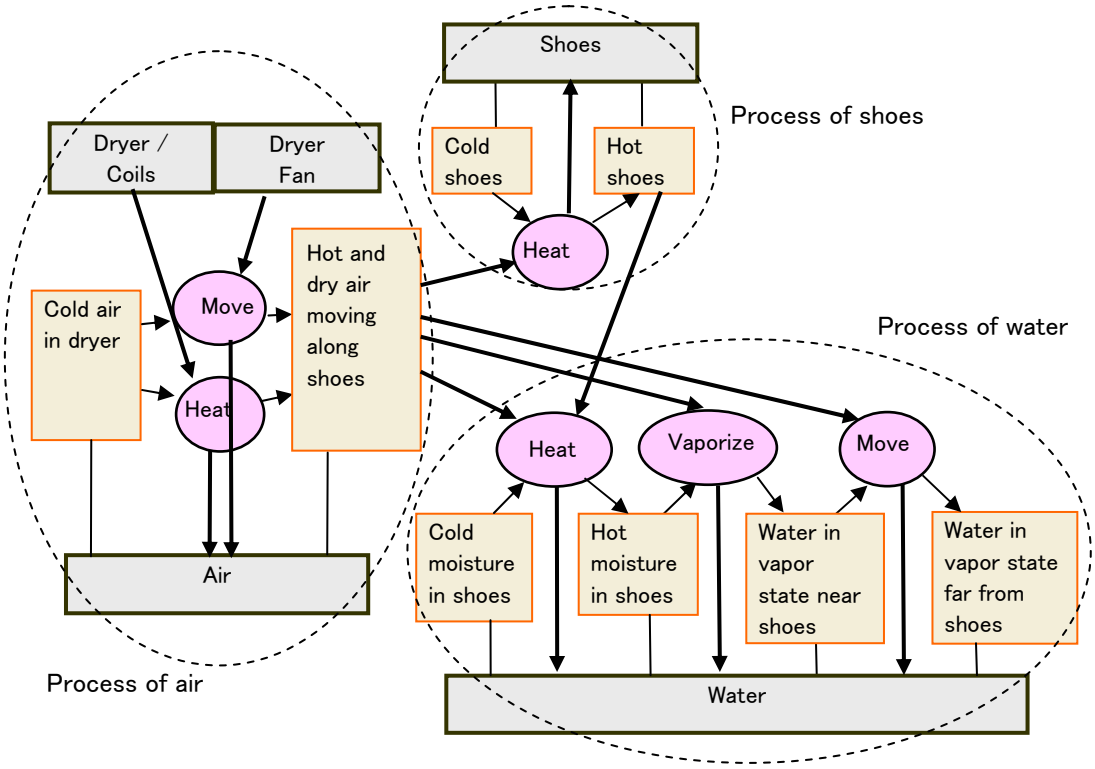

Figure 4. An Example of Cause- Effect Diagram: Shoes Dryer in 'Hierarchical TRIZ Algorithms' (HTA), p.D4 [3][1]

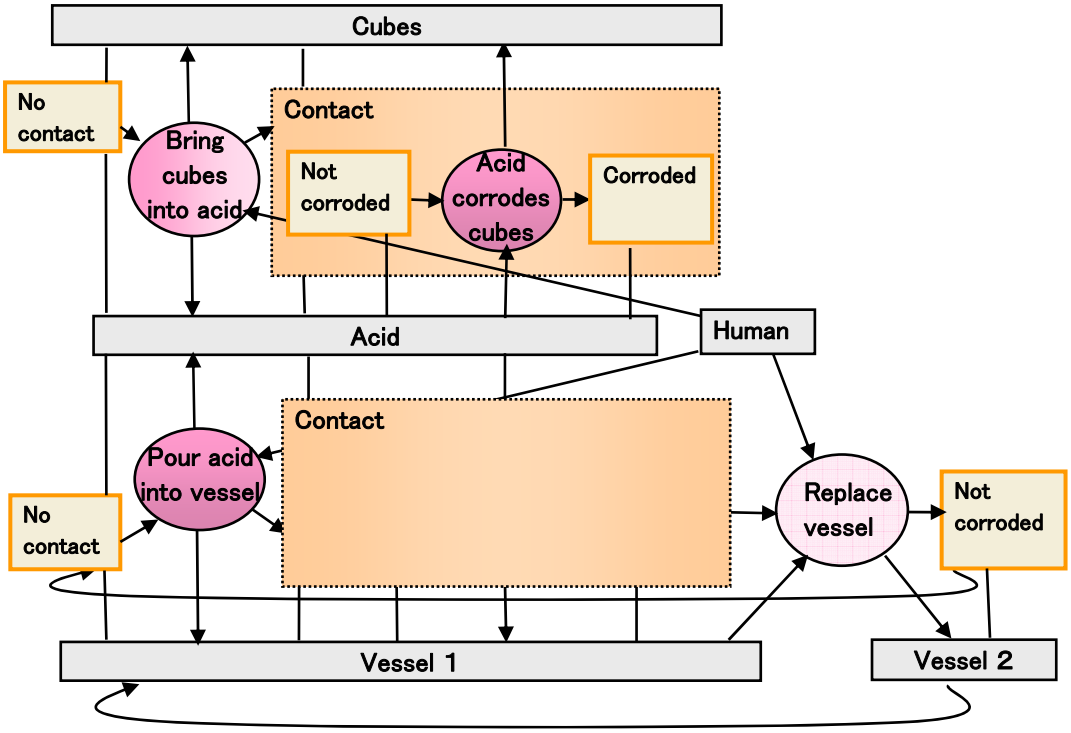

Figure 5. An Example of Cause- Effect Diagram: Corrosion in HTA, p.G6, H6 [3][1]

3. Changing Objects

In this section I deal with the structure of changing objects from formal point of view. The structure of changing objects will be useful if it can be obtained corresponding to the situation based on the model in recognizing the real world. Our recognition is based on "**Object 1- Process Object- Object 2 model"**, so I enumerate the way of changing Object in a structural way around this model in following way.

Figure 6. Object 1 and Process Object change Object 2

"Object 1- Process Object- Object 2 model" is a simplified one that extracted from m- to- n bilateral relation in the real world to show one to one causal relation, which usually means Object 1 and Process Object change Object 2. I recognize the real world and use the system as this model says. In this recognition various granularities are available so both Object 1 and Object 2 are not restricted to System Object. We can see the world at various granularities. A social activity acts on another social activity. A physical action acts on another physical action to change the direction of its action.

We have types of "phase" or "mode" on the relation between human being and technical systems or institutional systems. In **"make mode"** we actively make or change technical systems or institutional systems. In **"use mode"** we simply make use of existing technical systems or institutional systems. [4]
This change as in "Use mode" is treated as **Principle U**.
Principle U of changing Object: Object 1 and Process Object makes Object 2
If only Object 1 and Process Object makes Object 2, Principle U of changing Object is available not only in "use mode" but also in "make mode".

But we can not use "Object 1- Process Object- Object 2 Model" to change the attributes of Object 1, Process Object or Object 2, because each attributes of this running model is our very target to change.
In use mode" in the real world we have little chance to change the object. Important thing is that the model is effective only in the real world at a granularity to change Object 2 for human being.

I change "Object 1- Process Object- Object 2 model" to the original bilateral relation maintaining one to one relation as in next Figure.

Figure 7. "Object 1- Process Object- Object 2 Model" with Bilateral Relation

This Figure shows the possibility that in some cases Object 1 and Object 2 make Process Object. This is shown in next Figure.

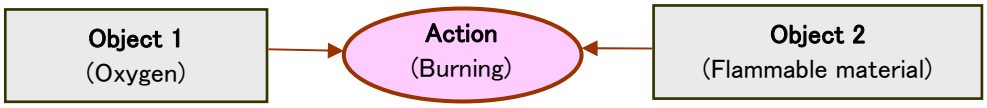

Figure 8. Change Process Object
Using "Object 1- Process Object- Object 2 Model"

Principle P of changing Object: We can make Object 1 and Object 2 in "Object 1- Process Object- Object 2 Model" change Process Object. To do so both Object 1 and Object 2 must be restricted to System Object.
It is important that it is not true that Object 1 and Object 2 must be always restricted to System Object. This could narrow our viewpoint.

Besides these Principles next Principle is a general one according to dialectic law.
Principle D of changing Object: Changing total attributes or structure of Object can generate a new Object or delete itself according to Dialectic law.

Contents of Object or **attributes** can change Object itself. Attributes of Object is total attributes and structure of lower level objects. Therefore change of total attributes or structure of lower level objects can change the contents of Object or Object itself. In dialectic relation

quantitative change of total attributes or structural change in lower level objects which consists of number of elements, structure of elements and elements themselves can cause qualitative change of object to become a new Object different from the existing one. This means to generate a new Object or delete the existing Object. This Principle includes the law of the mutual transformation of quantitative and qualitative changes in usual dialectics.

Principle U or Principle P can act on Principle D.

First type of operating Object is a very ordinary one but important.
Type A of operating Object: We can Act on each Object in "Object 1- Process Object- Object 2 Model". Both Object 1 and Object 2 must be restricted to System Object.

This Type is divided into two Types according to granularity in space.
Type AM of operating Object: We can act on Object 1 or Object 2 in "Object 1- Process Object- Object 2 Model" to change them directly by person. To do so both Object 1 and Object 2 must be restricted to System Object.

In Type AM we restrict the granularity of dealing with direct operation of "Object 1- Process Object- Object 2 Model" by each individual.
From outside of the Model we obtain next type to be able to operate every Object via other object.

Type AO of operating Object: We can act on each Object in "Object 1- Process Object- Object 2 Model" from Outside of the model via other object. Both Object 1 and Object 2 must be restricted to System Object.
In Type AO we can act not only on Object 1 and Object 2 but also on Process Object different from in Type AM.

The other type of operating Object is rather simple. In "use mode" Object can be generated, deleted or changed in the real world according to "Object 1- Process Object- Object 2 Model". On the other hands in "make mode" we can intervene in this real time process seemingly freely without any restriction. In this intervention we can generate, delete or change Object in "make mode". The only restriction we have here is that "make mode" can transit to "use mode" smoothly. That is to say it is the only restriction that "Object 1- Process Object- Object 2 Model" that is to be made in "make mode" can be realized in the world. This means that in this type we can bring in, bring out or replace Object in "make mode" freely regardless of existing Object.
Type R of operating Object: We can bring in, bring out or Replace each Object of "Object 1- Process Object- Object 2 Model", whole model itself or even a series of the model freely regardless of existing Object.

This Type is divided into two Types according to granularity in time.
Type RM of operating Object: We can bring out, bring in or replace each Object of "Object 1- Process Object- Object 2 Model" or whole model itself freely regardless of existing Object.
Type RE of operating Object: We can bring out, bring in or replace a series of "Object 1- Process Object- Object 2 Model" freely regardless of existing Object.

Thus we get Types of operating Object by person and Principles of transforming Object in processes of real world shown in next tables.

Table 1. Type of Operating Object

Types of operating Object by person		To be changed	Transformed by	Output
Type AM	We can change Object 1 or Object 2 in Model* directly. (Object 1 and Object 2 must be restricted to System Object)	System Object	Change on System Object by person	System Object
Type AO	We can change each Object in Model* from Outside of the model. (Object 1 and Object 2 must be restricted to System Object)	System Object, Process Object	Act on Object by person	System Object, Process Object
Type RM	We can bring out, bring in or Replace each Object of Model* or whole Model* itself freely regardless of existing Object.	System Object, Process Object, Model*	Bring out, bring in or Replace System Object, Process Object, Model*	System Object, Process Object Model*
Type RE	We can bring out, bring in or Replace a series of Model* freely regardless of existing Object.	Series of Model*	Bring out, bring in or Replace series of Model*	Series of Model*

Model*: "Object 1- Process Object- Object 2 Model"

Table 2. Principle of Object Transformation

Transformation Principle in the real world		Input	Transformed by	Changed
Principle U	Model*run as in the real world.	System Object, Process Object	Model*	System Object
Principle P	We can make Object 1 or Object 2 in Model* to change Process Object.	System Object (Object 1 and Object 2 must be restricted to System Object)	Model*	Process Object
Principle D	Changing total attributes or structure of Object can generate a new Object or delete itself according to Dialectic law.	Change attribute of System Object, Process Object by person	Autonomous movement according to dialectic law	System Object, Process Object

Model*: "Object 1- Process Object- Object 2 Model"

Changing Object is usually done in two steps in which each Types of operating Object by person and each Principles of changing Object using Type of operating Object in the real world combines to make up a whole of changing Object including Process Object. The outline of whole structure of changing Object physically is shown in Figure 9 and table 3. I can **enumerate all kinds of changing Object using "Object 1- Process Object- Object 2**

Model”. We use these figure and table to find out input of Object to get the target of Object for every logical types of changing Object such as deleting Process Object.

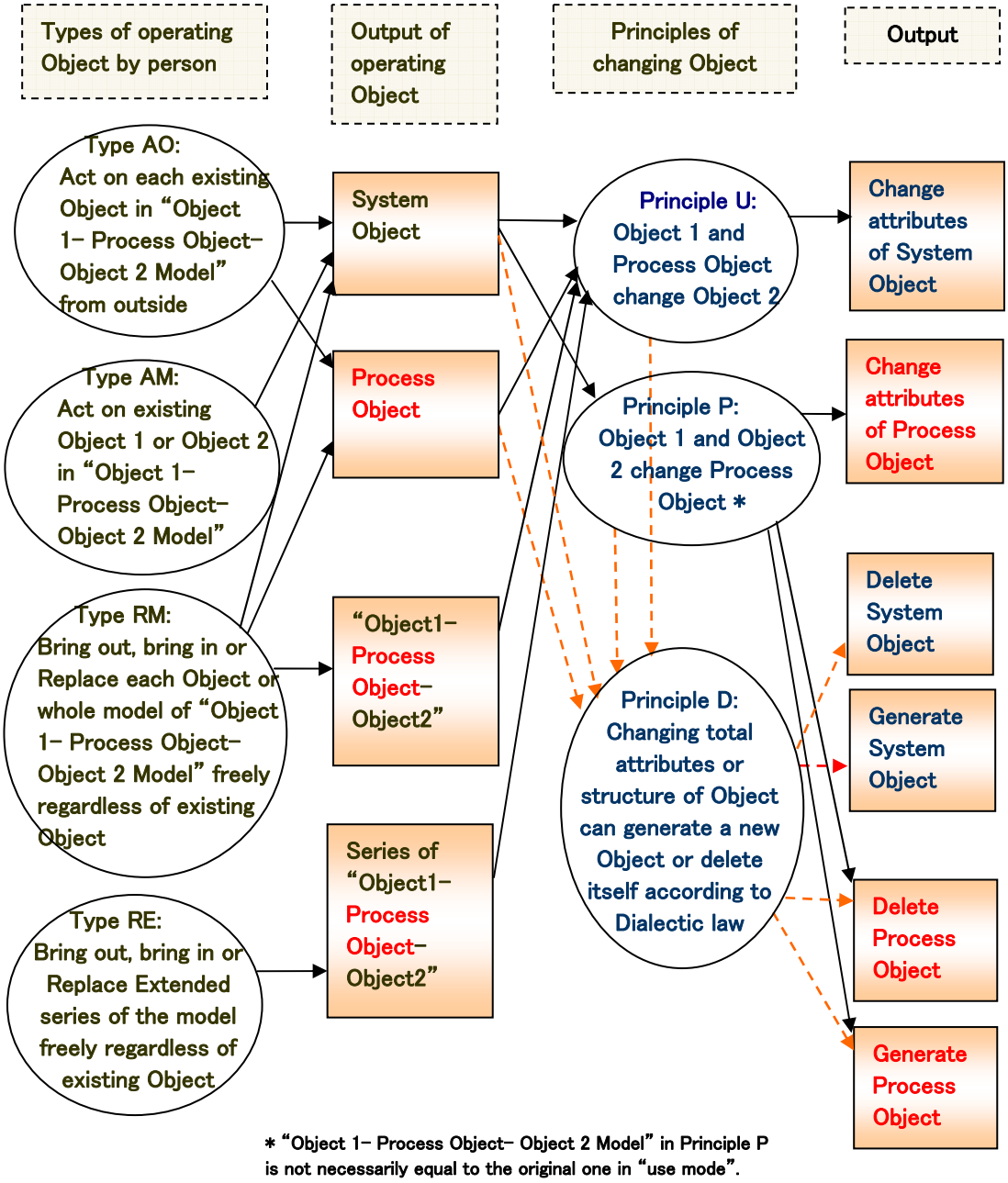

Figure 9. Structure of Operation and Transformation of Object

Table 3. Table of Operation and Transformation of Object

Granularity	Type	Object to be Operated	Principle	Output					
				Delete Process Object	Generate Process Object	Change Process Object	Delete System Object	Generate System Object	Change System Object
From Outside	AO	Object 1 Object 2	U						X
			U-D*				X	X	
			P	X	X	X			
			P-D*	X	X				
			D				X	X	
		Process Object	U						X
			U-D*				X	X	
			D	X	X				
Model	AM	Object 1 Object 2	U						X
			U-D*				X	X	
			P	X	X	X			
			P-D*	X	X				
			D				X	X	
Model	RM	Object 1 Object 2	U						X
			U-D*				X	X	
			P	X	X	X			
			P-D*	X	X				
			D				X	X	
		Process Object	U						X
			U-D*				X	X	
			D	X	X				
Model	RM	Model*	U						X
Extended Model	RE	Series of Model*	U						X

*: Principle U or Principle P act on Principle D. **: "Object 1- Process Object- Object 2 Model"

4. Resolving Differences

4.1 Three Logical Types of Changing Objects and Three Types of Purposes

We can change Object or resolve differences by changing attributes of Object, generating Object or deleting Object on condition that we don't deal with generating or deleting attributes of Object for the present. Among these we have **three logical types of changing Objects** contributing function which consist of **generating Process Object and deleting Process Object** contributing to fundamental function or process function **and changing attributes of Object** contributing to subsidiary function or attribute function.
In short **three logical types of changing Objects are generating Process Object, deleting Process Object and changing attributes of Object**.

Now we see changes from a point-view of contents or purpose of function. To begin with fundamentally we don't intervene autonomous change in the nature and we don't deal with an action from outside.

So the first case is that we feel need to make entirely new function in "use mode". We treat this case as the type of "**making new function**". The second case is that we notice a need to delete or change function in "use mode". We treat this case as the type of "**problem solving**". **In "use mode" we will need only these two types of direct purpose.**

We decide to treat the other types as "**idealizing**". **Idealizing is the third type of purpose directly for production side in Resolving Differences**.

We have two compatible aspects of idealizing. In the first aspect idealizing has some relation with making new function and problem solving. In these cases making new function and problem solving become a trigger of idealization. In "use mode" we notice a need only to make new function, solve problem in changing Object. The purpose of making new function and problem solving are clear and we have a need to do so. In other words all we have to do are to make a new function or solve problems. If I went too far in my argument I am not concerned with the performance of the newly made function or the result of problem solving. So we may add an action to idealize or improve these actions. To have a good solution we need to idealize the newly made function or the result of problem solving and the actions to make them even at the granularity of changing only one Object. Larry Ball deal with idealization in making new function at p.D3 in 'Hierarchical TRIZ Algorithms (HTA)'. [3] Simplifying or "idealizing", "use of resources" or "trimming" in TRIZ is an example in this aspect.

This is also done independently from making new function and problem solving.

In the second aspect we idealize the system or Object world making use of the law of technological or institutional system such as the law of trends in system innovation in TRIZ.

Anyway **we have three types of purposes in Resolving Differences which consist of making new function, problem solving and idealizing.**

Let us consider the situation around Resolving Differences.
First, all these three types of purpose can be in existing System. But we can only generate Process Object in making new system.

Secondly the relation between the three types of purpose and the three logical types of changing Objects contributing function are as follows.

- We can make new function by generating Process Object.
- We can solve problems by deleting Process Object or changing attributes of Object.
- We can idealize system also by deleting Process Object or changing attributes of Object.

4.2 Structure of Resolving Differences

We resolve differences in three steps shown in Figure 10.
1) In the first step of **deciding the purpose** we set the purpose in terms of Object as an output of Figure 9 or Table 3 of "Operation and transformation of Object" to be obtained. We recognize the differences depending on the situation to have one of the three logical types of changing Objects contributing to function consisting of generating Process Object, deleting Process Object or changing attributes of Object. To decide the purpose, **it is necessary to**

recognize not only the specific contents of purpose which depends entirely on the situation but also the form of logical type on which the purpose based. For example to solve a problem of acid attack we aim to minimize cost of replacement of vessel or eliminate action to pour acid to vessel which is to change attributes of Object or delete the Process Object.

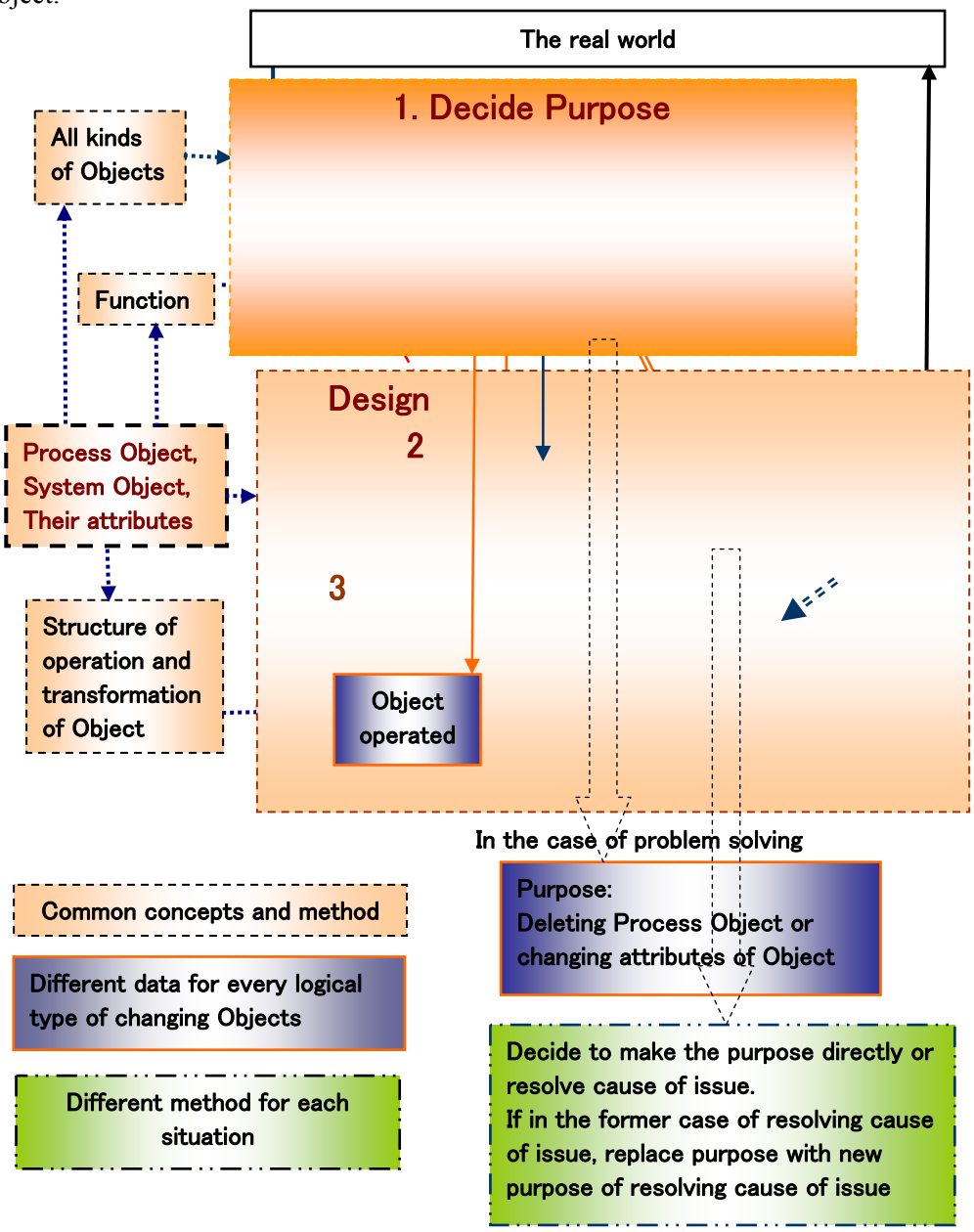

Figure 10. Structure of Resolving Differences at the Granularity of Changing One Object

2) The third step is **designing** to operate Object to get the purpose of output of Figure 9 or Table 3 of "Operation and transformation of Object". We use these figure and table to find

out input of Object to get the target of Object for every logical types of changing Object such as deleting Process Object.
We can do so by grasping Object as every thing we recognize and every way to handle Object including Process Object based on logical types of changing Object at adequate granularity of formality. We can treat every issue by language but at inadequate granularity of formality.
3) Between these two steps almost always we need the second step. The first part of the second step is **designing** to obtain knowledge about how to operate Object depending on the situation. The second part of this **designing** step is necessary if we must replace the purpose set in the first step with the new means to realize the purpose. This could occur in the case of taking the prior means to realize the purpose in making new function or to resolve the cause of problem in problem solving.
Three types of purposes in Resolving Differences which consist of making new function, problem solving and idealizing decide the way of the second step. For example in the case of problem solving we must decide to make the purpose directly as in ASIT [5] or resolve cause of issue as in usual TRIZ. If in the former case to resolve cause of issue, we replace set purpose with this new purpose of resolving cause of issue.

4.3 Three Types of Resolving Differences

We have three types of Resolving Differences depending on three types of purposes at the granularity of changing one Object as follows shown in Figure 11.

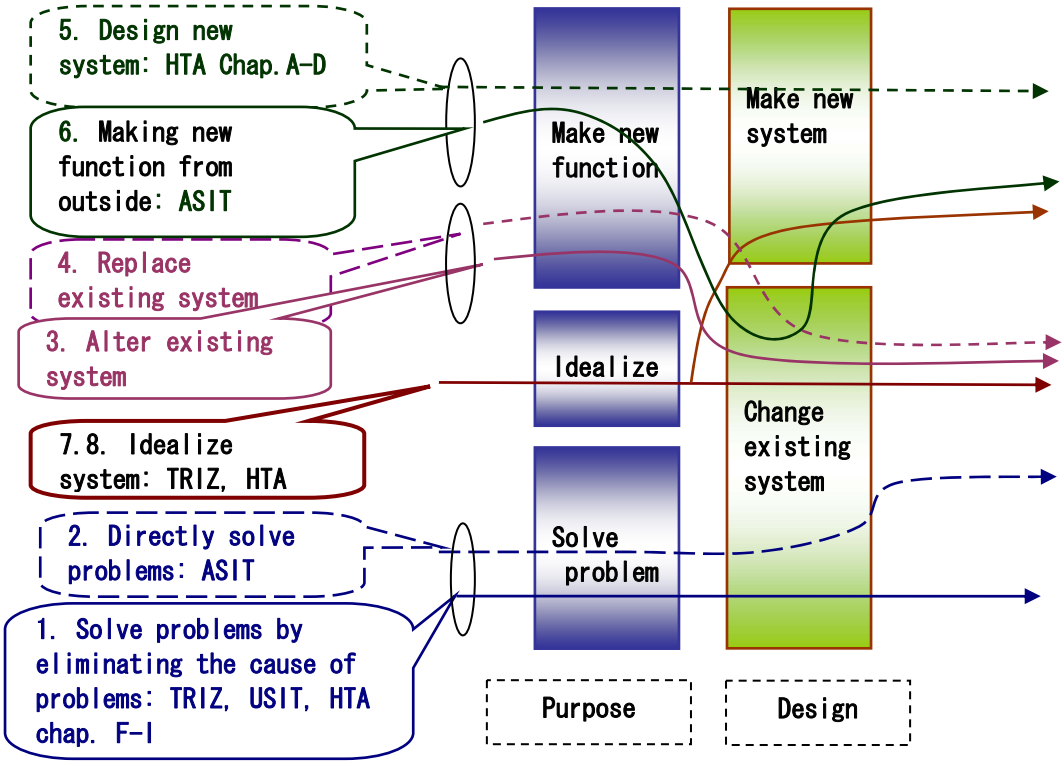

Figure 11. Types of Resolving Differences

[Solving problems]

1) Solve problems by eliminating the cause of issues: TRIZ, USIT, HTA chap. F to I [3]
2) Directly solve problems: ASIT [5]
We can treat solving problem directly in ASIT [5] as a type of making a new function not to cause problem by altering existing system
[Making new function]
3) Making new function on the occasion of replacing existing system
4) Alter existing system
5) Design new System: HTA Chap. A to D [3]
6) Making new function from outside: ASIT [6]
[Idealizing]
7) Idealize using the law of the inside system: TRIZ
8) Improve or idealize system after solving problem or making new function: TRIZ, HTA Chap. B to E [3]

5. On classical TRIZ and "Hierarchical TRIZ Algorithms" [1]

This paper is a study on TRIZ from a formal point of view not on contents of TRIZ. I don't intend to touch existing contents of TRIZ which is, I believe, very useful as it is. But I think it would be more useful if it expanded to deal with Process Object explicitly and combined with this method.

To tell the truth, the title of my previous article in Japanese [1] is 'A Method of Resolving Differences Based on the Concepts of Functions and Process Objects――Or a Comment on "Hierarchical TRIZ Algorithms"――'. In this previous paper I introduce HTA and made some comments on it including its strong point. It said also about the differences between his method and my opinion. [1] Some of them were as follows.

1) How to grasp Object. *Note 1, Note 2*
This paper shows outline of grasping Object. We should see Object or Object world at more various granularity. Only when using Principle P of changing Object in "Object 1- Process Object- Object 2 Model" both Object 1 and Object 2 must be restricted to System Object.
2) How to express cause- effect diagram.
This paper shows some examples.
3) Method of Resolving Differences
This paper shows the possibility of unified way to deal with various kinds of Resolving Differences which consist of idealizing, making new function and problem solving.
4) Dealing with contradiction *Note 3*
In HTA as same as in TRIZ opponents or opposites which cause contradiction are restricted to attributes of System Object. [3] On the contrary from what I said about function and Object opposites of technical contradiction or functional contradiction are Process Object or attributes of Object. In the same way opposites of physical contradiction or causal contradiction are Object or attributes of Object. These mean that both opposites are not restricted to attributes of System Object. [1]

Note 1: 1) How to grasp Object. [1]
In the example of corrosion in chap. G and H of HTA (Figure. 5) [3], I grasp Object and the purpose as follows.

***System Objects**:*
Cubes, Acid, Vessel(Attribute: cost, its value: C)
***Process Object**:*
Test of Cubes(State: operating time, its value: t),
Vessel Corrosion(State: operating time, its value: t, Attribute: Rate of corrosion, its value: replace n times in t), In p.G6 [3] hours of use is practically treated as state of Process Object.
Replace Vessel(Attribute: Cost of vessel, its value: C, State: It takes time to replace vessel, its value: t_r)
***Purpose**: From figure 5 and above analysis the purposes of this problem can be:*
Eliminating vessel,
Eliminating Process Objects that pour acid into vessel, acid corrodes vessel or replace vessel,
Or minimize cost of replacement per operating time nC / t or nC / (t + ntr).

Note 2: 3) Operation of Object. [3] [1]
In chap. K practically Process Object is treated excellently in idealizing but in chap. L it isn't in problem solving.

Note 3: 4) Dealing with contradiction [1]
41) In the above example, if I choose a candidate of solution to eliminate Process Object pouring acid to vessel, undesirable side effect may take place to cause a contradiction.

411) Then in this case opposites of the fully formed contradiction (HTA p.I3) is "not pouring acid into vessel not to make contact each other" and "pouring acid into vessel to make acid contact with cubes".
Latter clause, "pouring acid into vessel to make acid contact with cubes" means that "pouring acid into vessel" cause "making acid contact with cubes" in the condition that cubes are in the vessel.
First clause, "not pouring acid into vessel not to make contact each other" means that "not pouring acid into vessel" cause "not making acid contact with vessel" and at the same time "not making acid contact with cubes" on the condition that cubes are in the vessel.
Thus the situation that cubes are in the vessel make the contradiction between "not pouring acid into vessel" and "pouring acid into vessel".

412) First I consider changing relations between two objects which are both ends of process object. The condition that cubes are in the vessel means the stable relation of cubes and vessel in space. I can change this relation to put cubes and vessel in parallel or put vessel into cube.
If I change to put cubes and vessel in parallel, "pouring acid into vessel" including "pouring acid into cubes" are separated into "pouring acid into vessel" and "pouring acid into cubes". So the fully formed contradiction, "not pouring acid into vessel not to make contact each other" and "pouring acid into vessel to make acid contact with cubes" is resolved by pouring acid into cubes.
This solution furthermore makes the side effect that acid after pouring into cubes make contact with vessel. To avoid this side effect, one of the solutions is to accumulate acid in cubes.

We can obtain this solution and the measure to deal with side effect on the condition that cubes are in the vessel, after we deal consciously with cubes and vessel separately.
We can find no solution if we changed to put vessel into cube.

413) Secondly I consider changing the Object that Process Object operates. I pour acid into vessel to make acid contact with cubes because cubes are in vessel. Now I change to pour acid in cubes. I can directly obtain the solution to generate a Process Object to pour acid to cubes not to make contact with vessel. This solution furthermore makes the side effect that acid after pouring into cubes make contact with vessel. To avoid this side effect one of the solution is to accumulate acid in cubes.

42) In the candidate of solution to eliminate System Object of acid or vessel, "opposites" of the fully formed contradiction is "eliminating acid or vessel not to make contact each other" and "not eliminating acid and container to keep in acid". The solution is "eliminating vessel and making cubes to keep in acid".

6. Conclusion

We resolve differences by deciding the purpose in terms of Object to make an input get to this purpose in the figure or the table of "Operation and transformation of Object".
In this step we use logical types of changing Objects to obtain the unified way to deal with various kinds of Resolving Differences. The granularity of the logical types of changing Objects coarser than language is suited to obtain the formality of this method.

7. Acknowledgments

I express my deep thanks to Mr. Larry BALL whose work encouraged me to write the papers and Prof. NAKAGAWA Toru who recommended my reading his book.

References

[1] TAKAHARA Toshio: 'A Method of Resolving Differences Based on the Concepts of Function and Process Object', 2nd Symposium of TRIZ, Osaka, Japan, Sept.2006. (Paper: Japanese. Slide: English)
[2] TAKAHARA Toshio: 'How Function is Realized in Problem Solving', The TRIZ journal, Nov.2003.
[3] Larry BALL: 'Hierarchical TRIZ Algorithms', The TRIZ journal. 2005- 2006, http://www.3mpub.com/TRIZ/
[4] TAKAHARA Toshio: 'How People Interact with Objects using TRIZ and ASIT', The TRIZ journal, Aug.2003.
[5] Roni HOROWITZ: 'ASIT's Five Thinking Tools with Examples', The TRIZ journal, Sept.2001.
[6] Roni HOROWITZ: 'Using ASIT to develop new products', The TRIZ journal, Nov.2001.