Пример разбора:

Case study description:

Initial information

The control units of air conditioner system deploy some portions of compiled software. The design of software architecture requires applying some modifications to reduce some high-to-low level of existing disadvantages. Those disadvantages refer to a) the performance and features of end product, and b) limitations from cost reduction objectives.

The given formulation of software disadvantages and problem tasks are described in QAW-ATAM documents in form of engineering scenarios with their tabled parameters.

Problem identification

By performing the traditional QAW procedure, the following selection of preliminary scenarios are under analyses:

Scenario 1: The system should respond in 0.5 second for user inputs in normal operation time

Scenario 2: The system should respond in 100ms for real time request (interruptions)

Scenario 3: Minimize the usage of RAM size

To handle those scenarios (i.e. solving hidden key problems), their expressions should be reformulated with additional information and dependencies. It is desirable to use a contradicting formulation to include the additional information as shown in **Table 5**

Table 5. Reformulating scenarios

Initial scenario	Modified scenario (contradiction)
The system should respond in 0.5 second for user inputs in normal operation time	The system should respond in 0.5 second for user inputs in normal operation time, to improve (or to provide) user satisfaction. but this will decrease (or damage) system modifiability, and will require higher RAM size.
The system should respond in 100ms for real time request (interruptions)	If the system will provide 100 ms as a limit of time response, the interruption of outdoor compressor will be handled which provide safety (reliability) but the there is a risk of increasing implementation complexity and mismatching other important factors (information)
The usage amount of RAM should not exceed 16 Kbytes	If the usage amount of RAM of the system is under 16 Kbytes, it is possible to meet H/W spec, which should supports both new and old model with same solution. (24Kb in new model, 16Kb in old model) However, there is a risk of increasing implementation complexity(Manage RAM per task) and stack overflow.

This exercise highlights some hidden details and apply “why?” approach when trying to clarify the key reason of an advantage or disadvantage. This approach is usually also used in building the cause-effect chains (or root cause analysis).

On the other hand, the reformulated scenario enables applying TRIZ-based abstracting language, via matching specific system properties with contradiction matrix parameters.

Problem solving

Solving key problems which are highlighted by formulated technical and physical contradictions (in this case study only technical contradictions are reviewed) starts with usage of contradiction matrix, and ends with describing conceptual directions of solving by inventive principles.

The description of problem solving is performed as following 3-step process:

A) Matching parameters: Transform the identified system properties into equivalent parameters of the contradiction matrix. This process is shown in **Table 6**

Table 6. Equivalent TRIZ- based parameters

			Contradiction matrix parameters	
			of improvement	**of worsening**
Contradicting properties of scenario 1 See	Advantages	High operation time User satisfaction	Speed, Duration of action of moving object, Loss of information, Loss of time	-
	Disadvantages	Modifiability RAM size	-	Device complexity, adaptability and versatility of manufacture
Contradicting properties of scenario 2 See	Advantages	Safety	Reliability	-
	Disadvantages	Complexity	-	Device complexity
Contradicting properties of scenario 3 See	Advantages	H/W Spec satisfaction	Adaptability	-
	Disadvantages	Complexity Stack overflow	-	Complexity of control

- *Remark 1:* The used here "contradiction matrix" are in of two editions: a) original Altshuller matrix b) Adapted by Goldfire Innovator™ for software objects
- *Remark 2:* Some of other contradictions might be physical contradictions, which solving way differs from the described here

B) Extracting inventive principles

After the step recognizing parameters, the contradiction matrix points which appropriate inventive principles are recommended to apply. Next **Table 7** describes some of those matched principles.

Table 7. Found inventive principles

	Inventive principles	**General description**
Contradiction 1	• Segmentation	- divide an object into independent parts - make an object easy to disassemble - increase the degree of fragmentation (or segmentation) of an object
	• Asymmetry	- divide an object into independent parts - make an object easy to disassemble - increase the degree of fragmentation (or segmentation) of an object
	• Preliminary action	- perform the required change of an object (either fully or partially) before it is needed - prearrange objects conveniently so that they can come into action quickly, without losing time during delivery

	• Other way around	- invert the actions that are used to solve the problem (for example, instead of cooling an object, heat it) - make movable parts (or the external environment) fixed, and make fixed parts movable - turn the object (or process) 'upside down'
	• Dynamics	- enable (or design) the characteristics of an object, an external environment, or a process to make it optimal or to find an optimal operating condition - divide an object into parts that can be moved relative to each other - if an object (or a process) is rigid or inflexible, make it movable or adaptable
	• Mechanics substitution	- replace a mechanical means with a sensory means (optical, acoustic, taste, or smell) - use electric, magnetic, and electromagnetic fields to interact with the object - change from static fields to movable fields, from unstructured fields to structured fields - use fields in conjunction with field-activated particles (for example, ferromagnetic particles)
	• Discarding and recovering	- make parts of an object that have fulfilled their function go away (discard by dissolving, evaporating, and so on), or modify the parts directly during operation - conversely, restore consumable parts of an object directly during operation
Contradiction 2	• Other way around	See up
	• Parameter changes	- invert the actions that are used to solve the problem (for example, instead of cooling an object, heat it) - make movable parts (or the external environment) fixed, and make fixed parts movable - turn the object (or process) 'upside down'
	• Segmentation	See up
Contradiction 3	• Change the degree of freedom	- change the degree of freedom of an object
	• Segmentation	See up

➢ *Remark:* The "Contradiction N" means the contradiction of scenario N.

C) Applying inventive principles: After selecting the potential inventive principles from the contradiction matrix, analyze the opportunities of their specific applications. The usage of supporting samples to understand how to apply an inventive principle can increase the performance.

Table 8. Inventive principles application

Inventive principles	Sampling references	Explanation of applying
Segmentation	• Client/Server Architecture • Serial batch processing • Paging • Data structures (linked lists) • Database partitioning • Database normalization • RAID array • Parallelism • Modular design	If the responding time depends on a sequence of elements/route, then choose the segment which could be optimized without increasing RAM

Asymmetry	• Compression with data loss • Priority • Ragged arrays • Asymmetric data compression • One-way hash function • 2-3 Trees	If the system has symmetric or asymmetric "action-respond" relationship, then increase the asymmetry level, to enable responding information in a form of frequent "notification events'
Preliminary action	• Compilation vs. interpretation • Presorting • Query optimization • Preallocation of resources • Caching	Perform responding events before it is needed by user. Or pre-arrange objects such that they can come into action from the most convenient place and without losing time for their delivery
Other way around	• ASP delivery model • Inverted index • Garbage collection • Pattern-matching in reverse	Include a part of responding information in the "request" format of action
Dynamics	• Dynamic memory and resource allocation • Software that adapts to hardware configuration • Software that adapts to user interaction • Packet routing • DNS • Load balancing • Associative array • Common Object Request Broker Architecture (CORBA)	Enable responding objects to have flexible movement (decrease dependencies) manner between operating zones
Mechanics substitution (Change type of interaction)	• Entity-Attribute-Value data model • User interface transformations • Data abstraction to capture physical phenomena (RFID) • VARCHAR data type • Artificial neural network	to apply this principle it is vital to find out some equivalent methods of mechanics in our information objects. After enhanced discussion between TRIZ experts, we can identify three methods (continuous movement, leveraging, hitting) – avoiding written explanations it is possible to consider "respond" event as one of those three methods
Discarding and recovering	• Paging • Garbage collection • Linear hashing • Data structures (linked lists) • Using temporary files for archive access	Design a responding entity at user side which gather pieces of required response information from direct object place during operation
Parameter changes	• Pacing data flow to manage the interaction between systems • Just-in-time compilation (JIT) • Transmission buffer packing • Tired data storage • Highlighting keywords and sentences in source documents • Codepage transformation • Variable bit rate in audio and video files	undefined
Change the degree of freedom	• Associative array • Dataflow computing • Load distribution • Paging	If the system supports physical memory device as hard disk, the part of data that is not used by a process is unloaded from RAM to a hard disk.
Segmentation	• Client/Server Architecture • Serial batch processing • Paging • Data structures (linked lists) • Database partitioning • Database normalization • RAID array • Parallelism • Modular design	If there are main module which MUST loaded in RAM, Move it from RAM area to ROM. It is possible to reduce usage amount of RAM without Complexity.

After providing the specific formulation of found inventive principle, the software architect review them for validating feasibility and specifying details of available ones. Sometimes, the architect is able to generate the specific ideas himself when looking on the samples of principles.

It is important to mention here, that the identified principles of QAW^{++} usually cover the suggested ones of traditional QAW.

Handling secondary tasks (Tradeoffs)

Secondary tasks (Tradeoffs) appear sometimes in QAW after suggesting and describing a new software architecture. The QAW++ contains a new feature (opportunity) of handling those secondary tasks which could be a serious barrier of applying a good idea or solution.

The case study includes consideration of two tradeoffs:

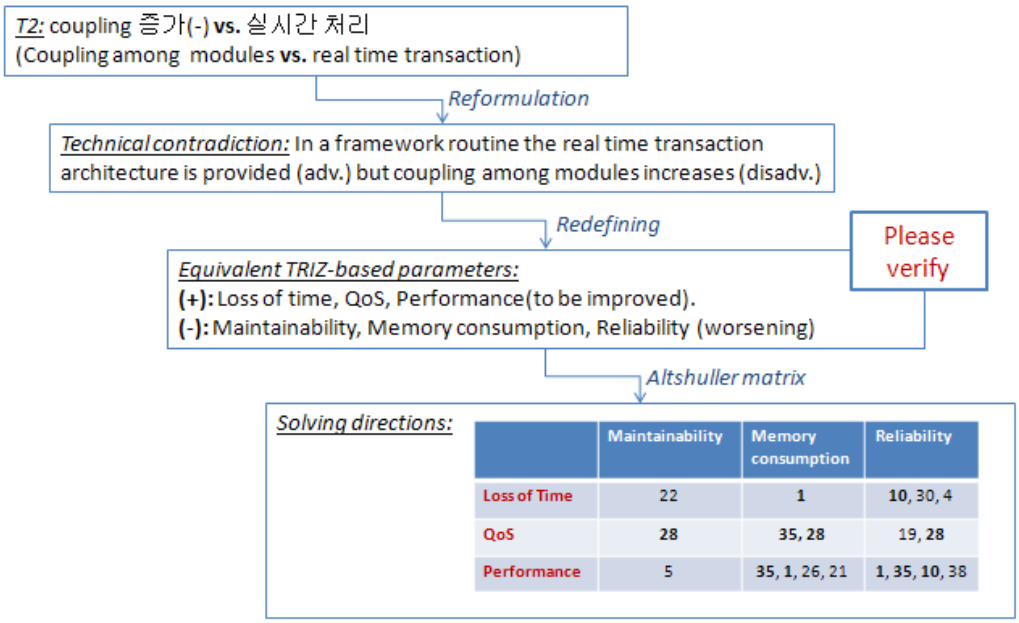
1) Coupling among modules vs. real time transaction
2) Source code volume vs. real time transaction

The procedure of handling those secondary tasks (tradeoffs) is similar to problem solving as it is based on using contradictions approach, which algorithm repeats same steps:

A. Formulating the technical contradiction using contradictive properties of a tradeoff
B. Find out the equivalent parameters from software-oriented contradiction matrix
C. Extract the recommended inventive principles from contradiction matrix
D. Provide explanation of each specific application of those principles
E. Validate the feasibility of generated ideas and conceptual directions of solving

This approach is implemented for mentioned tradeoffs as shown on **Figures 5, 6**:

Figure 5. Handling Tradeoff 1



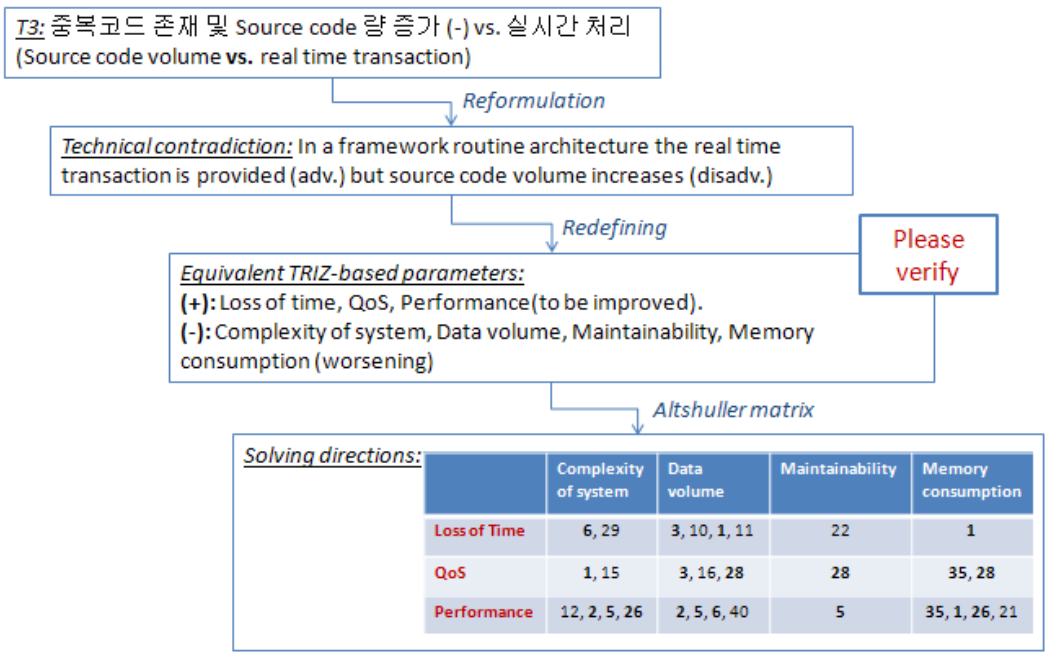
Inventive Principles - Level 1: (1, 10, 28, 35)

- **10: Preliminary action** (samples)
  - perform the required change of an object (either fully or partially) before it is needed
  - prearrange objects conveniently so that they can come into action quickly, without losing time during delivery

Inventive Principles - Level 2: (4, 5, 19, 21, 22, 26, 30, 38)

- **4: Symmetry change** (samples)
  - change the shape of an object from symmetrical to asymmetrical
  - if an object is asymmetrical, increase its degree of asymmetry
- **5: Merging** (samples)
  - bring identical or similar objects closer together; merge identical or similar objects; assemble identical or similar parts to perform parallel operations
  - make operations contiguous or parallel, bring them together in time
- **19: Periodic action** (samples)
  - use periodic actions instead of continuous action
  - if an action is already periodic, change the periodic magnitude or frequency
  - use pauses between actions to perform a different action
- **26: Copying** (samples)
  - instead of using an unavailable, expensive object, use simpler and inexpensive copies
- **30: Flexible shells and isolating layers** (samples)
  - change boundary conditions
  - isolate an object from the environment by means of flexible shells and isolating layers
- **38: Active objects** (samples)
  - use objects that provide more intensive interaction than the existing interaction

Figure 6. Handling Tradeoff 2



Inventive Principles - Level 1: (1, 3, 5, 6, 26, 28, 35)

- **3: Local quality** (samples)
  - change an object's structure from uniform to non-uniform, change an external environment (or external influence) from uniform to non-uniform
  - make each part of an object work in the conditions that are most suitable for its operation
  - make each part of an object fulfill a different and useful function
- **6: Multifunctionality** (samples)
  - make a part or an object perform multiple functions, eliminate the need for other parts

Inventive Principles - Level 2: (10, 11, 12, 15, 16, 21, 22, 29, 40)

- **11: Beforehand compensation** (samples)
  - prepare emergency means beforehand to compensate for the relatively low reliability of an object
- **12: Equipotentiality** (samples)
  - change the behavior of an object so that it will minimize or stabilize the use of energy (information) in the object
- **15: Dynamic parts** (samples)
  - enable (or design) the characteristics of an object, an external environment, or a process to make it optimal or to find an optimal operating condition
  - divide an object into distributed parts
  - if an object (or a process) is rigid or inflexible, make it dynamic or adaptable
- **16: Partial or excessive actions** (samples)
  - if 100 percent of an effect is hard to achieve using a given method, use slightly less or slightly more of the same method to make the problem easier to solve
- **29: Change the degree of freedom** (samples)
  - change from objects with a simple structure to objects with a composite structure
- **40: Composite objects** (samples)
  - change from objects with a simple structure to objects with a composite structure