

The Conflict Resolution Diagram

The Logical Thinking Process

The Logical Thinking Process is a set of Systems Thinking tools that helps us to understand goals and problems and come up with breakthrough solutions. The tools were first presented by Eli Goldratt in his novel “It’s Not Luck”, a sequel to “The Goal”. Several textbooks have been written to explain how to apply these tools. The following is based mainly on “The Logical Thinking Process” by H. William Dettmer. **The Conflict Resolution Diagram (CRD) is used to better understand conflicts between two ideas or two courses of action.** The CRD is sometimes called the “Evaporating Cloud”.

We don’t use the Conflict Resolution Diagram for interpersonal conflicts. The Conflict Resolution Diagram may help to clarify a conflict of ideas which is one of the causes of the interpersonal conflict.

When do I use a Conflict Resolution Diagram?

1. You know what your goal is and which critical success factors and necessary conditions you need to achieve the goal. If not, use the **Intermediate Objectives Map** to clarify your goal.
2. You know that you miss one or more of the necessary conditions to reach your goal. You’ve examined the root causes that cause this undesirable situation. If not, use the **Current Reality Tree** to find those root causes.
3. You see that the root causes of the undesirable situation arise out of a conflict, which can have one of two forms:
 - **Contradiction:** one branch of the reasoning leads to “I must do X”; another branch of the reasoning says “I must not do X”. For example: *the government must spend more money to revive the economy AND must spend less money to decrease the deficit. I need X AND I need not X.*
 - **Exclusive alternatives:** I need to two necessary conditions to reach my goal, but I can’t have both. Typically, we lack the resources (money, time, people...) to pursue both alternatives. For example: *I need to spend time at work delivering value AND I need to spend time at work learning BUT I don’t have time to both. I need X AND Y BUT I can’t have both or I need to choose between X OR Y.*
4. Apply the **Conflict Resolution Diagram** to better understand the contradiction or exclusive alternatives. This can have the following outcomes:
 - We discover that the assumptions behind the conflict are incorrect and there is no conflict, only a misunderstanding.
 - We discover that the conflict can be resolved without compromise if we can have or do something, a new fact called an “injection”. We agree to explore ways of achieving the injection. We can use the next tool, the Future Reality Tree to explore the consequences of implementing the injection.
 - We don’t get any further insight into the conflict, usually because we’re unable or unwilling to uncover all assumptions behind our thinking. We recommend you step away from the conflict and return to

it later with the help of people not involved in the conflict who can bring some fresh questions and perspectives.

- A win-lose solution or compromise (or lose-lose) solution is not acceptable.
5. Once we have ideas to improve the situation, we explore them using the **Future Reality Tree** and **Prerequisite Tree** tools. We can then establish a plan of action using a **Transition Tree**.

Who uses the Conflict Resolution Diagram?

We typically encounter two situations:

- **One person owns the whole conflict:** the conflict owner is the “client” and at least one other person helps the client use the Conflict Resolution Diagram as a “consultant”. The consultant should approach the situation with a completely open mind. The consultant should only ask questions of the following three types:
 1. **Open Questions** invite the interviewee to tell a story. E.g. “Will you tell me about...”, “What happens next?” or “Will you give me an example?”
 2. **Control Questions** ask for the facts in the story. E.g. “How many...”, “How often...”, “Where...”, “Who...”.
 3. **Confirmation Questions** verify that the interviewer has understood correctly. E.g. “If I’ve understood correctly, <explain what you understood>. Is that correct?”
- **Two people each advocate one of the exclusive alternatives.** Each advocate fills in half of the diagram, the side representing the position of the other advocate. Each advocate must at least be willing to ask questions and listen to the other’s point of view, they should act as “consultant” for the other advocate. An independent “consultant” can help to ensure that the rules of the process are followed. If the advocates can’t work together, the consultant can work with each of them independently to clarify their positions and to get some insight into the position of the other advocate, before bringing the two advocates together.

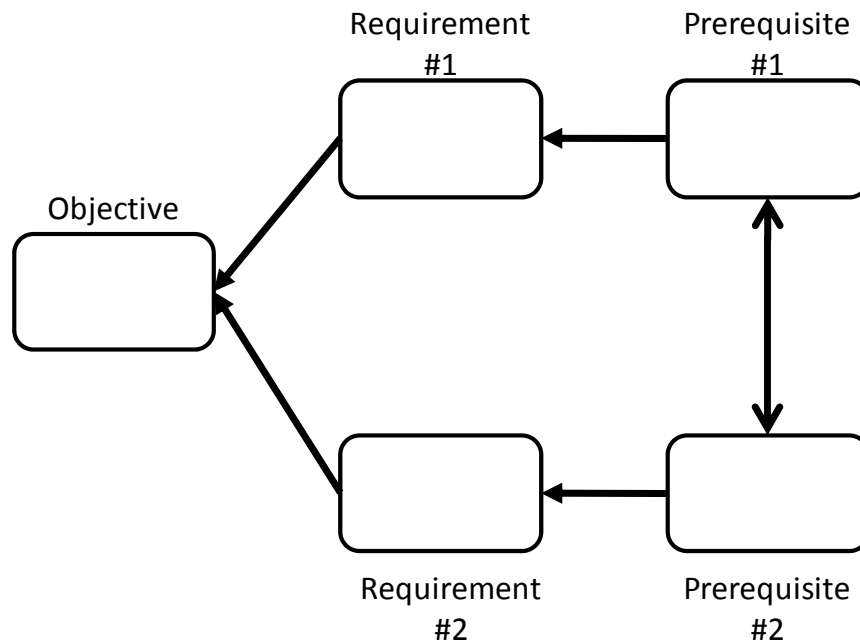
In any case, we must avoid trying to find (or worse even, push) solutions when we build a Conflict Resolution Diagram. The aim of the tool is to better understand the problem, before starting to look at possible solutions. If any participant has difficulty with the questions, they should practice with the “Nine Boxes” technique first.

How do we apply the Conflict Resolution diagram?

We recommend using low tech tools like large sheets of paper, pens and Post-Its. Don't hesitate to replace a Post-It when you can improve the clarity of the diagram!

1. Create a blank Conflict Resolution Diagram

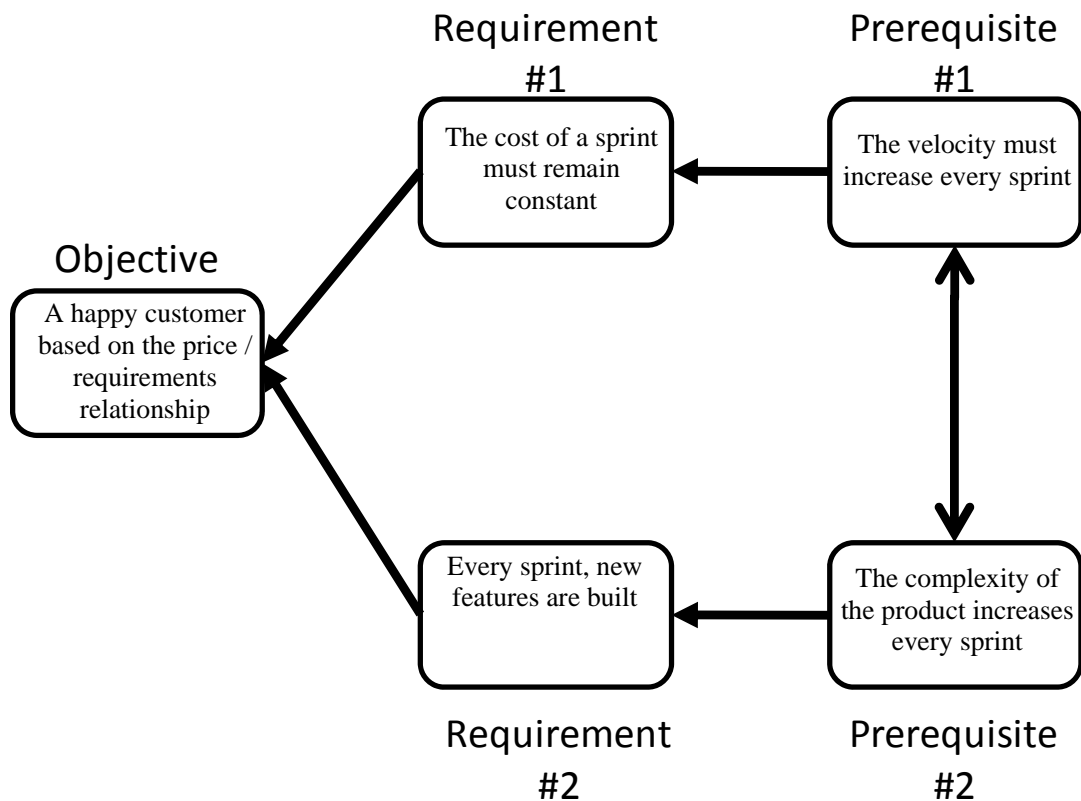
Draw the following diagram on a large sheet of paper. Leave plenty of space around the diagram.



Consultant: ask the client to briefly explain the context of the conflict.

Client: briefly give some background information that is required for participants to understand the case you will present.

Example:



2. Articulate the conflict

Client: fill in Prerequisite 1 and 2 with the two terms from the conflict.

Consultant: ask questions to ensure that the meaning of the two terms is clear.

Example: see CRD above

3. Determine the goal and requirements on each side

Consultant: ask the client why they need each prerequisite. You should end up with a statement like “In order to have <requirement> we must <prerequisite>”. Ask questions to ensure that the requirement is clearly described. Don’t question the logic (yet).

Client: the requirements usually come directly from your Intermediate Objectives Map.

Consultant: find out the common objective of the two requirements. The objective must be concrete and the two requirements must be necessary to achieve the goal. You should end up with a statement like “In order to have <objective> we must <requirement 1> AND <requirement 2>”. Don’t question the logic (yet).

Client: the goal usually comes directly from your Intermediate Objectives Map. Clarify the requirements as necessary.

Example: see CRD above

If we can’t find a common goal for the conflicting prerequisites, that can mean two things:

- There isn’t a conflict because we’re looking at two completely unrelated systems

- We're looking at the wrong level of the system. Say we have the situation: Objective1 requires Requirement1 requires Prerequisite1 and Objective2 requires Requirement2 requires Prerequisite2. Why can't we have both P1 and P2? Because we want both O1 and O2. Why do we need both O1 and O2? Because there's a goal O3 that requires both O1 and O2. O3 is our real objective.

4. Evaluate the entire relationship

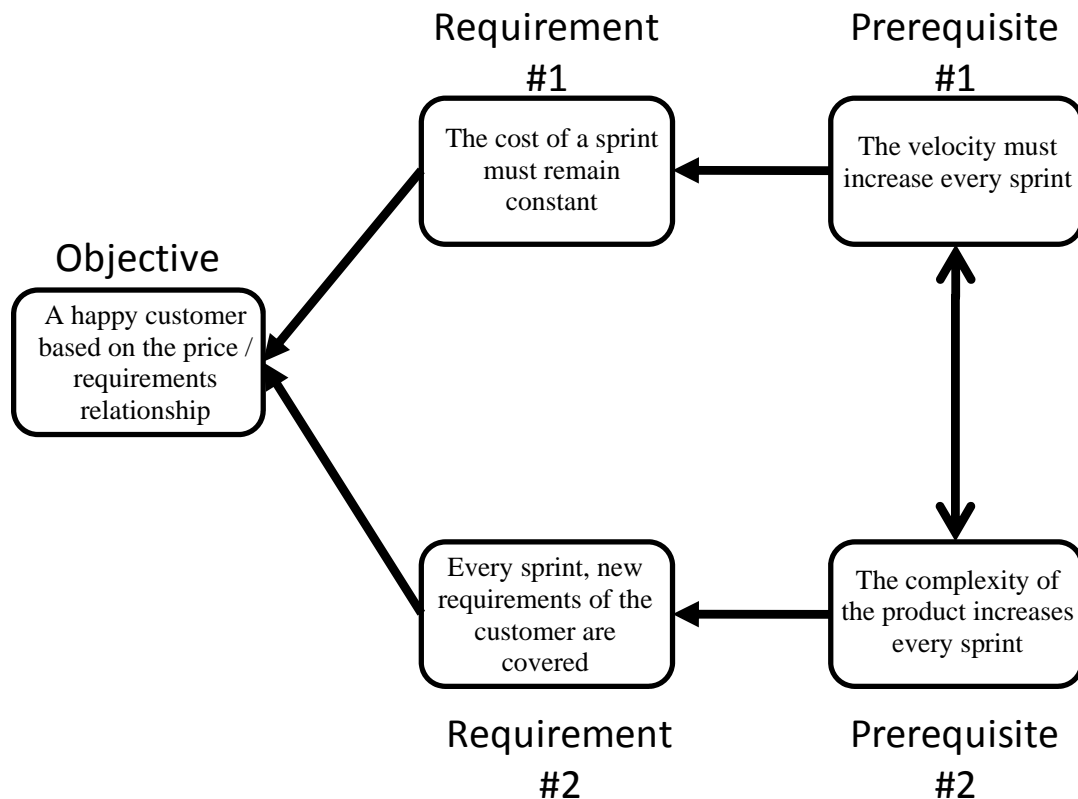
At this point, it's useful to bring in someone who wasn't involved in the construction of the diagram.

Client: explain the logic of the diagram by reading the diagram from left to right:

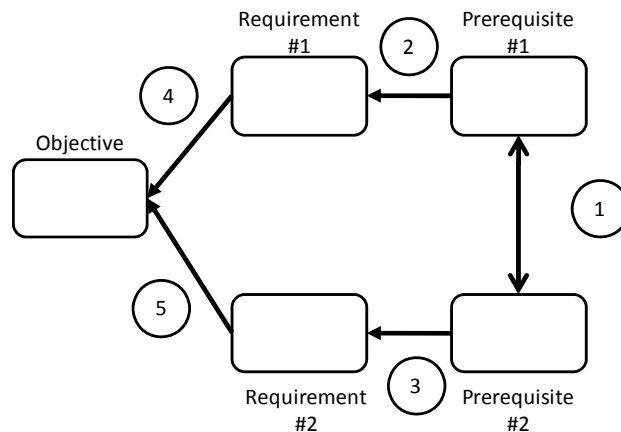
- In order to have <objective> we must <requirement 1> and <requirement 2>
- In order to have <requirement 1> we must <prerequisite 1>
- In order to have <requirement 2> we must <prerequisite 2>
- We can't have <prerequisite 1> AND <prerequisite 2>

Consultant: evaluate the **clarity** of the reasoning. Did we avoid tautologies ("in order to be rich we need lots of money")? Can someone who knows the context immediately understand the diagram? If not, ask questions to clarify the five statements. Don't question the logic (yet).

Example: we reformulated requirement #2 for clarity.



5. Develop underlying assumptions



Consultant:

- Examining each relationship in turn, as indicated on the diagram above, ask **why** this relationship holds. E.g. “Why can’t have <prerequisite 1> AND <prerequisite 2>?” Write down as many assumptions on Post-Its as the client can come up with. Ask questions to clarify the reasoning. Don’t question the logic (yet).
- Put each Post-It next to the concerned relationship.
- If the client runs out of assumptions, propose your own assumptions. Only add those assumptions that the client agrees with.
- Move on to the next relationship when the client can’t find more assumptions.

Client: brainstorm assumptions without censoring. We want to uncover all potential assumptions. Some of these assumptions will turn out to be wrong. Use *extreme wording* where appropriate. If you don’t end up with at least five assumptions per relationship, you’re probably holding back or censoring your ideas.

Example:

1.
 - More covered requirements in the product always means more user documentation has to be written
 - More covered requirements always means more knowledge transfer and communication in the team
 - The more requirements are covered, the more testing and regression testing are needed → testing of the GUI can not be automated
 - The more requirements are covered, the more code is produced, which increases complexity
 - The more code, the more performance tuning is needed
2.
 - It is possible to increase the velocity every sprint
 - The customer pays per sprint
 - Increasing the velocity is the only way to keep a fixed price for a sprint
3.
 - The project lasts forever
 - The budget is endless
 - It is impossible to cover new requirements with existing features
 - The customer gets a kick out of the number of covered requirements

- Deliver more and new features is the only way to cover the customer's requirements

4

- The customer's idea about pricing is fixed and cannot be changed
- A customer will always be unhappy when the price changes
- The customer wants to pay per sprint
- There is another parameter that is changing and affecting the price so that the velocity must increase (eg salary)

5

- The customer knows all its requirements
- The customer has an endless amount of requirements
- Every requirement that is covered by the product delivers the customer the same amount of new business / income (the business value of every requirement is the same)

"Extreme Wording"

When writing down the assumptions we can use *extreme* words like *only*, *best*, *always*, *everywhere*, *all the time* or *everybody* to exaggerate our claims. E.g. "In order to make people happy at the party (requirement) we need to serve ice cream (prerequisite) because *everybody* loves ice cream (assumption 1) and ice cream *always* cheers people up (assumption 2) and I love eating ice cream *all the time* (assumption 3).

These statements invite us to reply "Oh yeah? Well...", to challenge the assumption. The next step allows us to (finally) challenge the assumptions and see if the reasoning is valid.

6. Evaluate the assumptions

Consultant: looking at each relationship in the same order as when we developed the assumptions, differentiate between valid and invalid assumptions. Valid assumptions are those where you can't find any counter-arguments or counter examples. Clarify the assumptions as needed, together with the client. If an assumption is invalid, write down the reason. Mark assumptions that can be challenged.

Client: evaluate each of your assumptions critically. Explain your reasoning and provide examples of the valid assumptions.

If a relationship ends up without valid assumptions, the diagram is invalidated.

If there are no valid assumptions behind the conflict between the two prerequisites, we don't have a conflict, we have a misunderstanding. If we don't actually need one of the prerequisites or requirements to achieve the goal, nothing stands in our way to achieve the goal. Clarity is very important for a CRD. Often we can resolve conflicts when we discover that we used the same term for two different concepts.

7. Create injections

Usually, the conflict doesn't just "evaporate". We need to use our creativity to create options ("injections") to solve the conflict in a win-win way. Some useful techniques are

- **Question the wording of contradictions.** If we have a contradiction of the type "do X AND NOT do X" are the two X's really the same? For example: *in one company there was a longstanding conflict between Sales and Production. Sales needed to offer **customised products** to match the specific needs of*

customers; Production needed to produce **standardised products (or not customise products)** to provide products at a competitive price. Customers want their needs to be met at a competitive price. A product can't be both customised and standardised. The conflict was resolved when they realised that they used the word "product" for different concepts. They could offer customised "sales products" built out of standardised "production products".

- **Challenge extreme assumptions** by finding counter-examples and restricting the applicability of the assumption. E.g. An assumption like "Because it always works..." can be restricted by adding "... *except in these specific circumstances*". If the assumption isn't true in those circumstances, the conflict may not exist in those circumstances. We may have made the problem smaller.
- **Find alternatives.** Assumptions of the type "X is the *only* way to..." practically invite us to provide alternative prerequisites or requirements to get to the desired objective. If we have an alternative which doesn't conflict, we've solved the problem. Even if we're not sure the alternative will work, we can agree to test or research the alternative. For example: "*TDD is the only way to improve the quality of the code!*" *Oh yeah? And what about reviews, pair programming, design by contract, code analysis tools, design training, more testers, better analysis and a thousand other quality-improving techniques?*
- **Find the assumptions behind the assumptions** by asking further "why" questions. For example "*We need two months to put a new release into production*". *Why?* "*Because we need six weeks to integration test the new release and two weeks to prepare the production environment.*" *Why test for six weeks?* "*Because we need to manually test every use case*". *Why manually?* "*Because we can't automate those types of test*". *Why every case?* "*Because we don't know what's changed in the release*". *Why test?* "*Because developers put bugs into the software.*" *Why do developers put bugs into the software?* "*Because they're stupid? I don't know!*" *Why don't we ask them?*
- **Try "What If?" scenarios.** Take a crucial valid assumption. What if that assumption were invalid? Imagine a world in which the assumption is invalid. Would that allow us to resolve the conflict? If yes, what would it take to make that assumption invalid? The real breakthrough solutions usually come from invalidating those assumptions that everybody holds self-evident. To be able to do that, we must first make those self-evident assumptions visible, which is why we need to avoid censoring and examining the validity of the reasoning until the end. For example: "*What if we could automate 50% of the tests? What if we could automate 80% of the tests? What if you knew what had changed? What if developers didn't put bugs in the software?*"

Example:


We defined the following injections:

1.
 - Lower the cost of the team
 - Automate (part of) the testing
- 4
 - Have the customer pay per business value instead of per requirement or sprint

See also:

"The Goal", "It's Not Luck", "Necessary but not sufficient" – Eliyahu Goldratt

“The Logical Thinking Processes” – Bill Dettmer

More Systems Thinking, Lean, Agile and Theory of Constraints resources available on http://www.agilecoach.net			
Pascal Van Cauwenberghe	Jef Cumps	Portia Tung	
pascal@agilecoach.net	Jef.cumps@ilean.be	portia@agilecoach.net	
Nayima	iLean	Emergn	
http://www.nayima.be	http://www.ilean.be	http://www.portiatung.org	