September, 2008

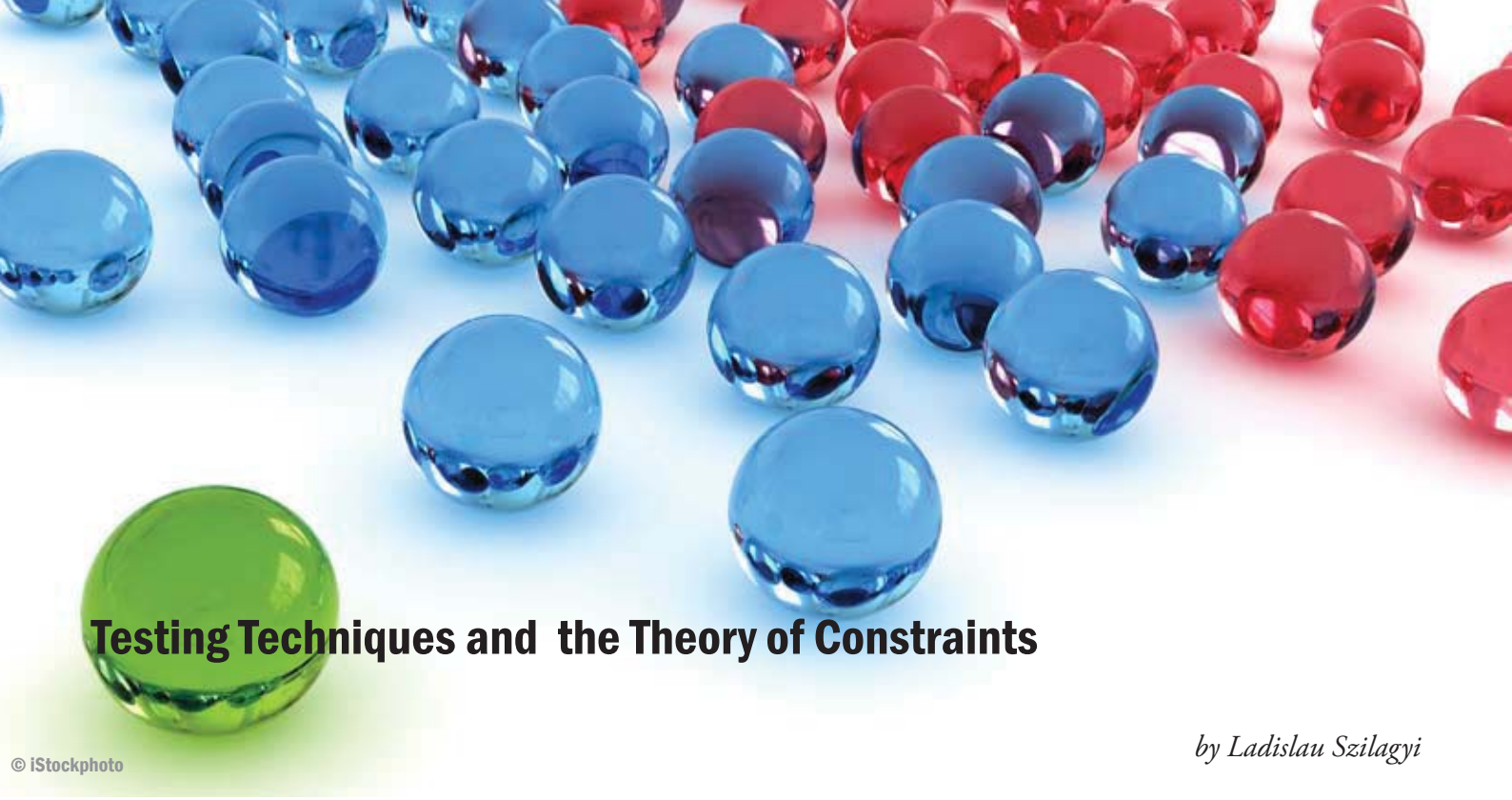
te testing experience

The Magazine for Professional Testers

Test Techniques in practice -

Do they help?
Why do we often test without them?

printed in Germany
free exemplar
www.testingexperience.com
ISSN 1866-5705



Testing Techniques and the Theory of Constraints

© iStockphoto

by Ladislau Szilagyi

Introduction

In the context of newly emerging testing schools and methodologies (see *Derk-Jan de Grood's Test Goal - Result driven testing*[4]), the topics of *Test strategy, tactics, testing mission* are hot and much debated in the worldwide testing community.

Also, in a real software project's day-to-day context, there are always constraints affecting the testing goals' optimal fulfillment.
How do we deal with these constraints? Here, we must use our **testing techniques** practical skills! Appropriate testing techniques must be used as mandatory key elements in this process of dealing with testing constraints, and *Eliyahu Goldratt's* way of thinking in ***Theory of Constraints*** may provide us with a proven methodology to be applied.

What is a testing process constraint?

According to Webster's dictionary, a constraint is "the state of being checked, restricted, or compelled to avoid or perform some action". I prefer to describe a constraint as "the conflict (between two actions), affecting the optimal fulfillment of an intended goal".

An example will help! Consider the following common situation: We are asked to perform *efficient and effective testing*. Apparently, that means we should obtain high test coverage in order to find as many bugs as we can (effective) and we should minimize test time (efficient). In order to obtain high test coverage, we should *execute as many test cases as we can*, whilst at the same time, to minimize test time, *we* should *execute only some of the test cases* (fig.1). That's a testing process constraint; We can't have both!

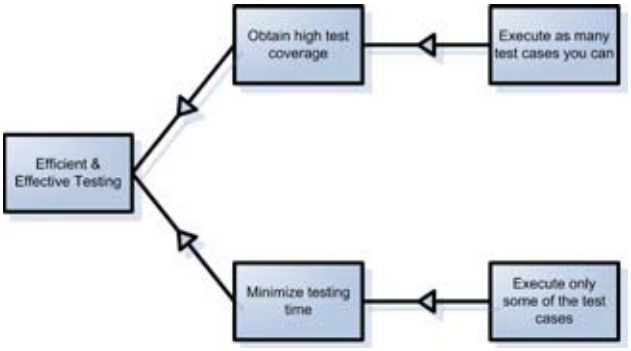


Fig 1. Constraint example

How can we deal with constraints? Here, we must use our **testing techniques** practical skills! It is obvious that the mastery of various testing techniques can help us find a way out of such situations. New testing techniques must be used as mandatory key elements in this process of dealing with testing process constraints. However, without an appropriate methodology, we may randomly jump to using another (possibly unsuitable) testing technique, and precious time will be wasted. I suggest that the thinking mechanism behind ***Theory of Constraints*** may guide us as a proven methodology.

The Theory of Constraints

The Theory of Constraints (TOC) is an overall management philosophy. It is based on the application of scientific principles and logic reasoning to guide human-based organizations. TOC is one of the favorite buzz-words of the current business management strategy school.

Obviously, it is out of the scope of this article to offer a detailed description and explanation of TOC. Instead, we will try to cover only some of the main ideas behind this theory.

In his bestseller "*What is this thing called THEORY OF CONSTRAINTS and how should it be implemented?*"[1], Eliyahu Goldratt, the author of TOC, points out the essence of his theory:

1. **Identify the Constraints**. Remember that to identify the constraints also means to prioritize them according to their impact on the goal.
2. **Decide How to Exploit the Constraints**. Now that we decided, how we are going to manage the constraints, and how should we manage the vast majority of the system's resources which are not constraints? We should manage them so that everything that the constraints are going to consume will be supplied by the non-constraints.
3. **Subordinate Everything Else to the Above Decision**. It's obvious we still have room for much more improvement. Whatever the constraints are, there must be a way to reduce their limiting impact.
4. **Elevate the Constraints**. But, can we stop here? There will be another constraint, but if we continually elevate a constraint there must come a time when we break it. Whatever we have elevated will no longer be limiting the system. Will the system's performance now go to infinity? Certainly not. Another constraint will limit its performance and thus…
5. **Go Back to Step1.**

TOC Thinking Processes

The Thinking Processes emerged as TOC practitioners needed to identify the core con-

straints and how to manage or elevate them. They needed the answers to three simple questions:

- What to change?
- What to change to?
- How to cause the change?

The thinking process steps of TOC uses some exotic terms:

1. Define the system – build the Intermediate Objectives Map
2. Analyze the mismatches - build the Current Reality Tree
3. Create a transformation – applying the "Evaporating Clouds" method
4. Design the future – build the Future Reality Tree
5. Plan the execution – build Prerequisite Trees

The decisive step to obtain the conflict resolution is the so-called "Evaporating Clouds" method (in fact, a method to "evaporate" conflict). Let's try to detail the underlying thinking process,by taking a look under the hood of the TOC Thinking Processes!

Evaporating Clouds

A "Cloud" is based on *Necessary Condition Thinking*: we must "execute as many test cases as we can" (wanted) as a necessary condition to obtain "high test coverage" (needed). We have already seen that when two "wants" appear to be mutually exclusive, there is a conflict. To solve this conflict, the way forward is to recognize that our "wants" are motivated by underlying "needs". We "want" to execute as many test cases as we can because we "need" to obtain high test coverage.

Let's build a graphic representation of our Conflict (fig.2).

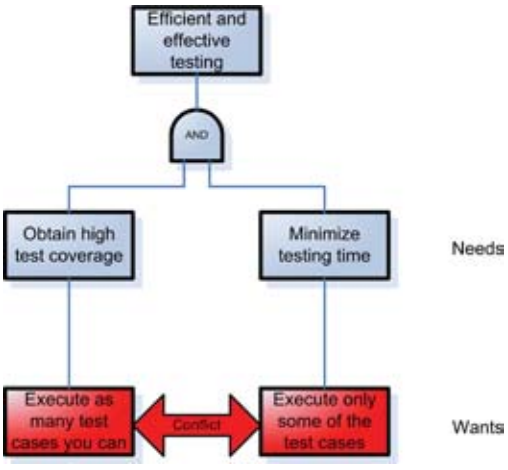


Fig 2. Needs and wants

The steps to follow in order to apply the "Evaporating Clouds" method are:

1. Identify the Wants ("execute as many test cases as you can", "execute only the most important test cases")
2. Identify the Conflict (can't have both!)
3. Identify the underlying Needs ("obtain high test coverage", "minimize testing time")
4. Identify the common Objective ("efficient and effective testing")
5. Identify and validate the Assumptions (in order to satisfy the Need we must obtain our Want because of our Assumptions)
6. Propose Solutions

The assumptions are basically "why" we must obtain our Want, and finding erroneous assumptions is the key to breaking the conflict. Assumptions "hide" underneath the Want-to-Need edges, and the Needs-to-Common Objective edges. There are also assumptions that underlie the Conflict entity itself— why we believe we can't have both Wants simultaneously.

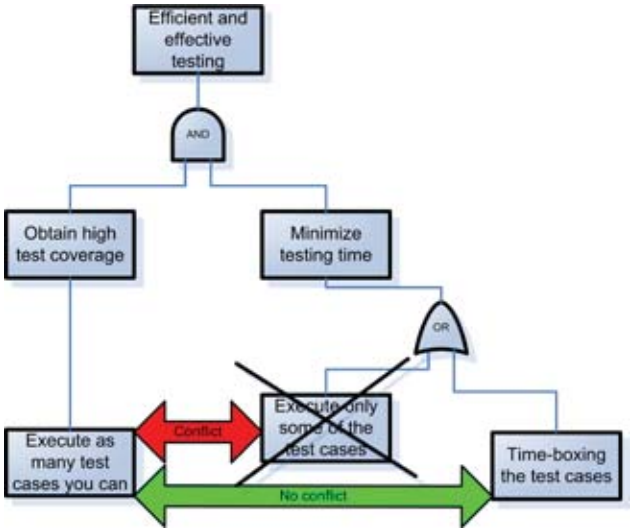


Fig 3. Propose Solution

Let's target the "execute only the most important test cases" Want. We assumed that in order to "minimize testing time" we *must* "execute only the most important test cases". Well, *must* or *may* execute? What if we imagine that perhaps there is another way to minimize testing time? Here, our testing skills shall guide us to search for alternatives…and one alternative is the risk-based testing approach! Yes, we could make a quality risk analysis (see5), prioritize the product features and come up with testing tactics involving various levels of testing (extensive, balanced, according to opportunity) and implying different time slices assigned to the test cases, depending on the risk of the feature being verified ("time-boxing the test cases" fig.3).

Oops, we just got our Solution, it satisfies our need to minimize the testing time (fig.4)!

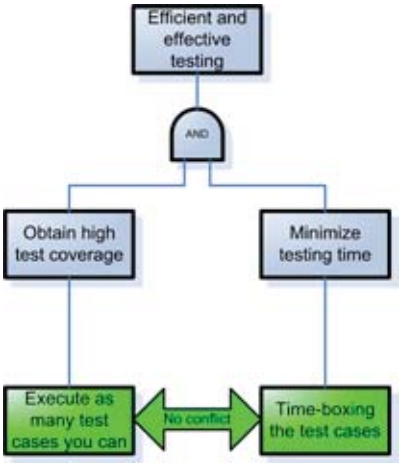


Fig 4. "Evaporated" conflict

Strategy, tactics and the practical use of TOC to remove testing process constraints

When we discuss the testing process, it's best to consider the whole picture. In the test process, the following levels define and apply test activities:

1. Test Plan (scope, approach, resources and schedule of intended test activities)
2. Test Strategy (high-level description of the test levels to be performed and the testing within those levels)
3. Test Tactics (practical actions performed by the tester when testing)

Strategy and tactics are different entities and are planned and performed in distinct time intervals. Strategy is setting the goal. In other words, the strategy sets the "What for?". There is no such thing as a 'unique test goal', but obviously there is a primary goal and there are many secondary testing goals, more exactly represented as a tree of inter-dependent testing goals (fig.5).

Tactics, on the other hand, are supposed to tell us "How we are supposed to reach the objectives." In other words, tactics answers the "How?". For any meaningful testing action we should be able to ask: "Why are we doing this? What is its purpose?". The answer to these questions is the strategic goal. Likewise, for any meaningful testing strategic goal we should be able to answer: "How do we obtain this? What actions are needed in order to achieve it?". The answer to these questions is the tactic action. That means that there must be

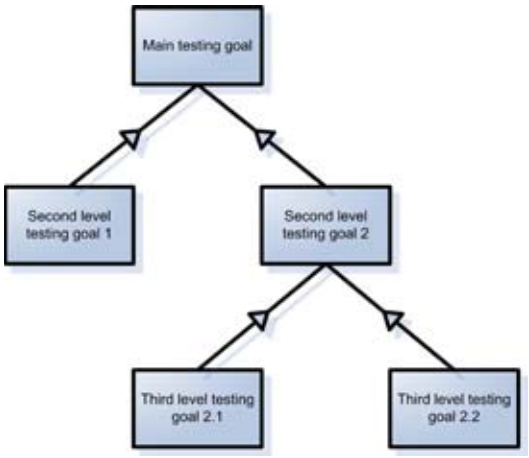


Fig 5. Testing goals tree

a corresponding tactic action behind any strategic goal, (fig.6).

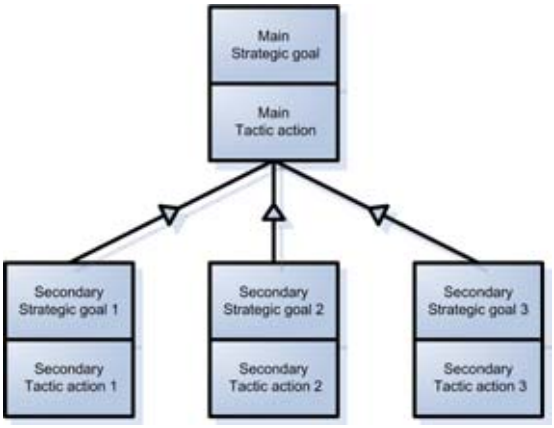


Fig 6. Strategic testing goals and tactic actions tree

Usually we cannot modify the test strategy *during the testing session*. The right time to update it will be *after* the testing session, when we learn the lessons (success or failure) from the testing mission. However, if we are doing *Exploratory testing*, we *may* modify our strategy during the testing session according to what we have learnt up to now. In either of these two cases, at the stage of performing the practical test tactic actions, we will be surely confronted with constraints affecting the testing goal's optimal completion.

Let's look more closely at the various categories of testing process constraints. Among the types of constraints impacting the testing mission's optimal completion, most of the types are related to:

1. Time (too short)
2. Cost (too high)
3. Quality (expectations too high – zero defects!)
4. Resources (lack of, insufficient, not trained)
5. Test environment (incomplete, not set up)
6. Test tools (lack of, not suited, difficult to maintain)
7. Test activities (none or wrong testing technique applied)

The first six categories of constraints must be handled by test managers in their continuous risk mitigation activities throughout the testing process life cycle. This is not always an easy task; you must admit that it's hard for a test manager to:

- request some more testing time (testing cost is implied too)
- criticize a manager for a wrongly imposed 'good enough' level of product quality
- ask for an extra tester to help the current testing mission
- ask for some more hardware to set-up the test environment
- convince the company's top management to buy a new testing tool
- instantly become an expert in using a new testing tool

I will focus here only on the last category, the constraints connected to the *test activities*, where often the constraint's root cause is the wrong option for an inappropriate *testing technique*. Here we are fortunately on solid ground, it depends only on us, the testers, to remove these constraints.

The *ISTQB Foundation & Advanced syllabi*[3] describe a complex taxonomy of testing techniques. These are the possible candidates to be used when searching for alternatives in the "evaporating clouds" method. In our previous example on how to remove a constraint, we saw a 'high-level' conflict situation ("execute as many test cases as you can" versus "execute only some of the test cases").

Functional Testing techniques:

1. Specification based (Black box)
   - Equivalence partitioning
   - Boundary values analysis
   - State transition testing
   - Cause-effect graphing
   - Decision tables testing
   - Syntax testing
   - Classification tree method
   - Orthogonal arrays, All pairs
   - Use Case testing
2. Structure based (White box)
   - Statement testing
   - Decision testing
   - Branch testing
   - Condition testing
   - Multiple condition testing
   - Condition determination testing
   - LCSAJ
   - Path testing
3. Defect based
   - Taxonomy
4. Experience based
   - Error guessing
   - Checklist based
   - Exploratory
   - Attacks
5. Static analysis
6. Dynamic analysis

Non-functional Testing types:

1. Accuracy testing
2. Suitability testing
3. Interoperability testing
4. Functional security testing
5. Usability testing
6. Accessibility testing
7. Technical security testing
8. Reliability testing
9. Efficiency testing
10. Maintainability testing
11. Portability testing

Obviously, at the *tactical* 'low-level' in our testing session, we will encounter different kinds of conflict, much more dependent on the current testing context. Here, we must rely on our ability to choose the suitable testing technique in the process of 'evaporating the clouds" of the constraint's conflict. In any such situation, try to draw a quick "needs-wants" graph and then start to identify the wrong assumptions, leading to the "evaporating cloud" solution candidates, selected from the testing techniques you know. Always try to select a "need" that can be satisfied using some other testing technique than the one currently used.

For instance, we may be confronted with the task of testing a dialog which is used to configure a system, and containing combinations of groups of radio-button choices. Here, the constraint may emerge as the conflict between *too many combinations* to be tested and the *limited testing time available*. As possible "solutions" we may use some testing technique candidates:

- *combinatorial testing* techniques (orthogonal arrays, all-pairs testing, t-way testing[6], including the use of tools like AllPairs, PICT, FireEye)
- *cause-effect graphing* and *decision tables*, to identify particular rules that could be hidden behind some particular groups of combinations
- *classification tree method*, to quickly visualize test ideas and find relevant test values

Keep in mind that the testing context (available time, test environment, tester experience, client expectations and feature criticality) will have a serious influence in our choice.

Well, to be honest, our quest does not stop here. There is usually more than one constraint affecting the completion of the testing mission. Our success will be critically influenced by our ability to pick the most important constraint! Here, too, TOC has developed a methodology to help us identify the critical constraint (but, that's another story to be told…please consult the referenced books on TOC[1,2]). And, finally, remember that the testing constraints removal is a continuous process, like the Deming's PDCA cycle. Once we removed a constraint, we will surely find another constraint that affects our testing mission's optimal completion.

Keep TOC's thinking processes as a valuable asset in your first-aid testing kit!

References

1. Eliyahu Goldratt - Theory of Constraints, North River Press
2. William Dettmer - Strategic Navigation: A Systems Approach to Business Strategy, ASQ Press
3. ISTQB - Foundation & Advanced level syllabus
4. Derk-Jan de Grood - TestGoal, Result driven testing, Springer
5. Rex Black – Critical Test Processes, Addison-Wesley
6. Yu Lei & others - IPOG: A General Strategy for T-Way Software Testing, Engineering of Computer-Based Systems, 2007

Biography

Ladislau Szilagyi holds a BS in Computer Science & Mathematics, 1978, Bucharest University, Romania and has more than 30 years of working experience in IT. He worked until 1991 at the Research Institute for Computer Science, Bucharest, authoring multitasking real-time kernels and process control systems. He has been involved in software testing since 1995 and now works as Test Manager at Totalsoft, Romania.

His main specialized areas of consulting and training are Quality Management, Testing and Software Process Improvement. He contributed several articles to the Carnegie Mellon's Software Engineering Institute Repository (https://seir.sei.cmu.edu/seir/), covering topics such as CMMI, Testing, Metrics and Theory of Constraints. He is ISTQB CTFL, member of ISTQB South East European Testing Board and an active ISTQB trainer.

Index Of Advertisers