

developer.*

The Independent Magazine for Software Professionals

How Did This Happen?

by Donald E. Gray

It was Saturday afternoon when the house phone rang. “Don, this is John. I know we haven’t talked in 10 years, but I have a client who has a problem.” In 20 years I’ve never had a client call and say “Don, things are going great! Why don’t we pay you to do nothing? Just send us an invoice.” So I’m not surprised when the first words I hear are “Something’s wrong, can you come and take a look?” What did surprise me was Saturday and at home.

George, the client, works for a major defense contractor. Recently George inherited a system he didn’t know much about. The system had suddenly started producing about 25% defects. At \$300 to \$ 25,000 per part, upper management was screaming to find the problem and solve it.

Prelude to a Problem

We can use a general systems thinking drawing called a “Causal Loop Diagram” (a.k.a. “Diagram of Effects”) to see the situation’s dynamics.

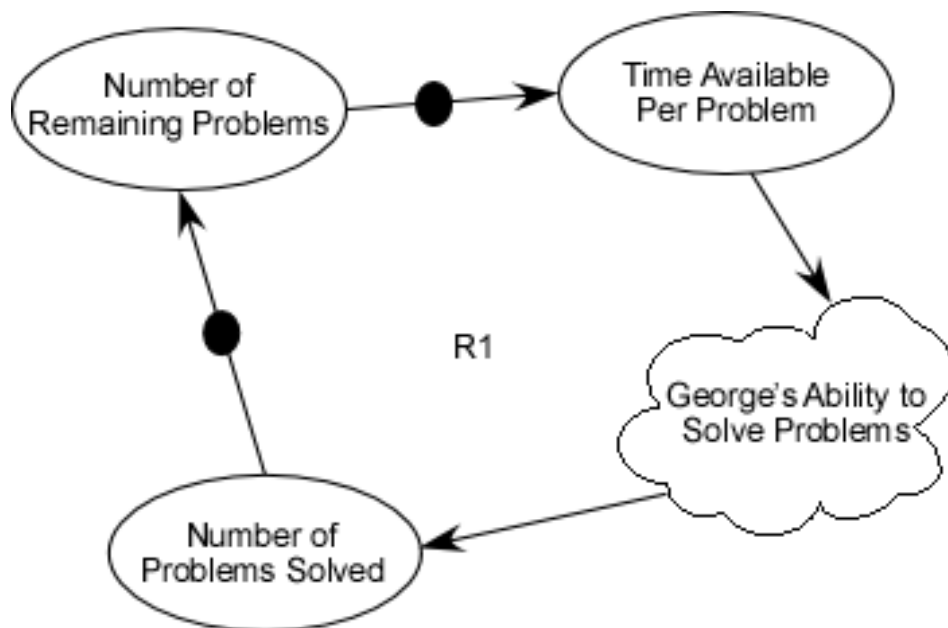


Figure 1: George’s Problem Solving Ability

You can start reading this diagram at any node. Cloud-shaped objects represent subjective quantities, values that defy quantifying, or are too expensive to quantify. “George’s Ability” doesn’t quantify easily, but you can experience it anyway. Ellipse-shaped objects represent objective quantities. Counting the number of problems and problems solved is easy to do. The arrows connect the variables and show the direction of influence.¹

As George’s ability to solve problems goes up, the number of problems solved goes up. Black dots indicate an inverse action, so as the number of problems solved goes up, the number of remaining problems go down. As the number of remaining problems goes down, the time available per problem goes up. As the time available per problem goes up, George’s ability to solve the problems goes up. And around and around it goes.

Figure 1 contains “fuzziness.” George’s ability to solve any given problem will vary. Has he seen a similar problem before? What is his energy level? Will solving a given problem fit in the time available or will George have to “come back to it?” Even so, if we know George, we have an idea what kind of problem solving ability he has, and it’s a useful approximation.

Another important fact about Figure 1 is that the loop reinforces itself. This means if things are going well, they tend to get even better. Conversely, when things are going poorly, they’ll tend to get worse. We indicate this reinforcing loop with an “R1” in the loop’s middle. When we add another reinforcing loop, we’ll label it “R2,” and so on. If things keep going well, eventually George will solve all the problems. I don’t think that will happen in George’s world, but theoretically it could. More likely, the QA department will want new information collected during production, clients will want different reports, the product engineers will decide to run the process differently. Or, George’s problem-solving ability will be recognized by management, and he’ll be “awarded” the department’s “problem project.”

Houston, We Have A Problem

When George inherits the problem project, suddenly, the number of problems he needs to solve skyrockets. Since this loop reinforces itself, as the number of problems shoots up, fewer problems get solved (or at least the problem-solving rate goes down). Customers become unhappy. QA rejects more parts. When management finally gets involved, the “BIG MANAGER” flies down to take hands on control until “the problem gets solved.” Dynamically, this looks like Figure 2.

¹ For more information, please read the *developer.* Magazine* article by Don Gray called “The Diagram of Effects” at www.developerdotstar.com/mag/articles/gray_diagram_of_effects.html.)

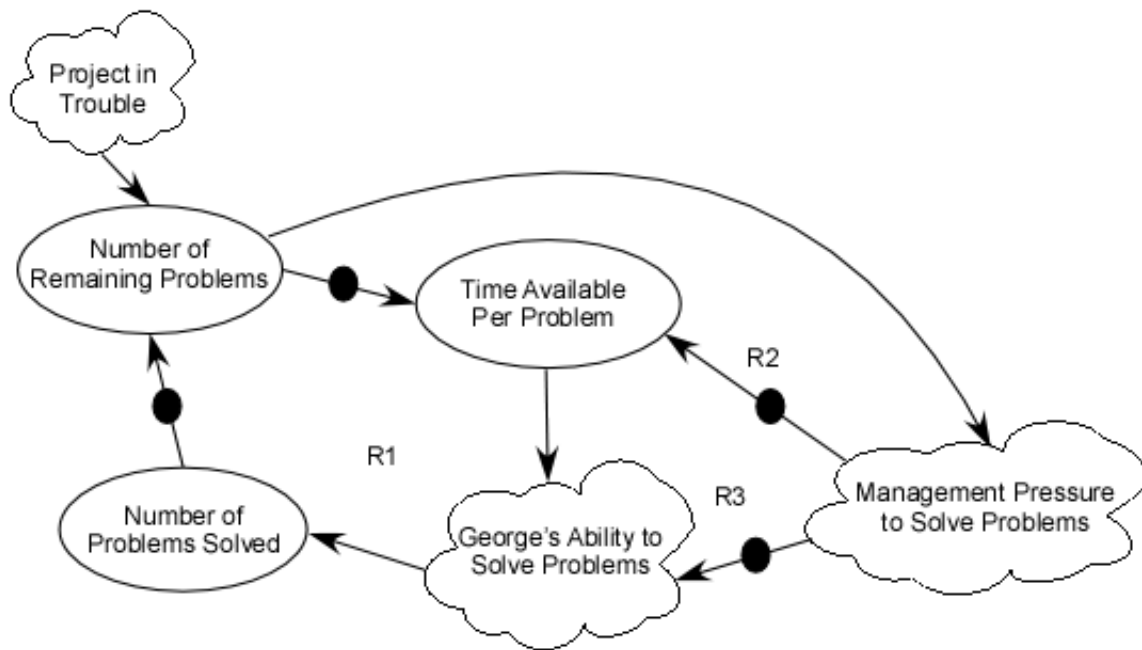


Figure 2: From Bad to Worse

Note that adding management pressure creates a boomerang affect. The management’s goal is to help solve the problem quicker, but now George has less time to solve problems since he’s required to attend the daily meeting from 8 – 9 AM explaining what’s wrong, and what’s being done to fix it. The BIG MANAGER also moved into George’s office space. Now George gets to overhear the phone conversations about the project. This creates an additional distraction that reduces George’s effectiveness. Management pressure actually creates two new reinforcing loops that amplify the effects in the original reinforcing loop.

Reinforcing loops are “engines” in systems thinking. They cause either growth or decay until something happens that stops the cycle. In the positive cycle, George would eventually solve all the problems. In the negative cycle the problems get worse until something collapses. George could quit or get sick. Clients could take their business elsewhere. To prevent collapse, management needs to intervene to balance the system’s dynamics. This led to the Saturday phone call.

When I became involved, the system structure changed again. The result looks like:

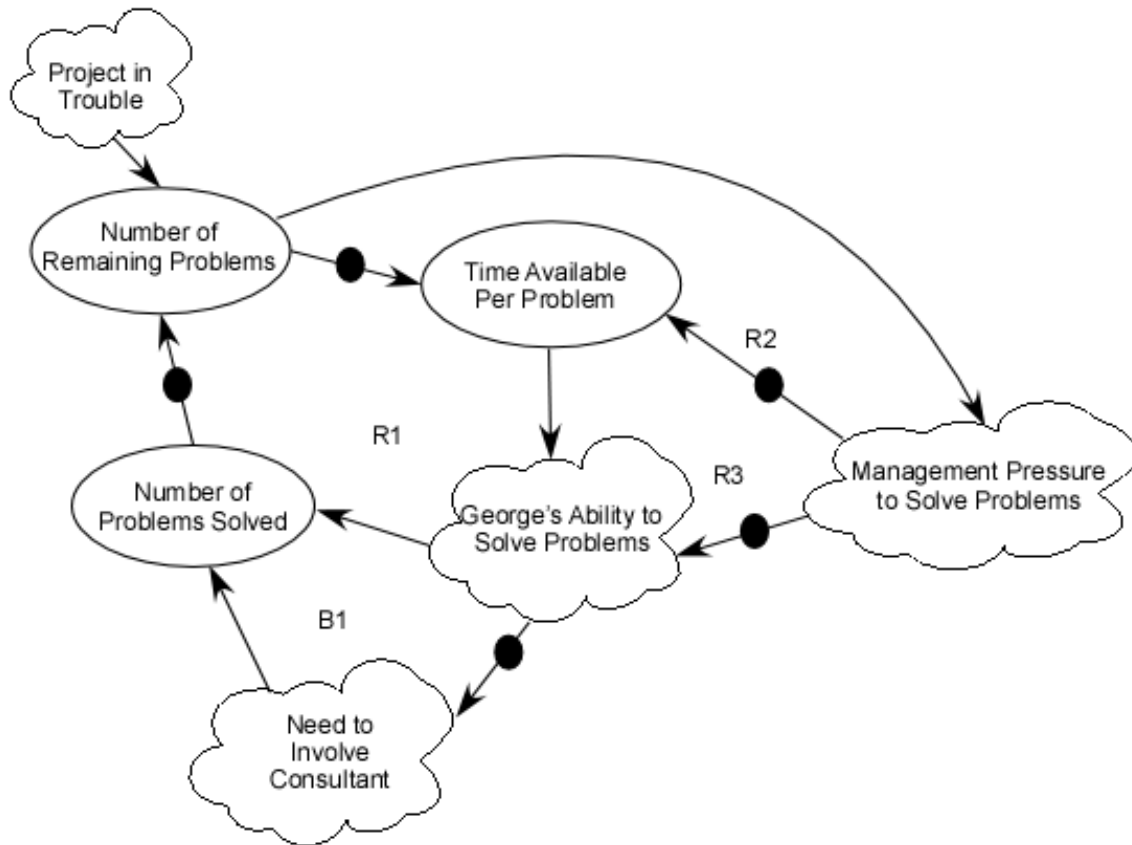


Figure 3: Starting to Balance

As George’s ability goes down, the need for George to obtain help goes up. As George gets more help (becomes more efficient), the number of problems solved goes up. As the number of problems solved goes up, the number of remaining problems goes down. As the number of remaining problems goes down, the time available per problem goes up. As the time available per problem goes up, George’s ability to solve the problems goes up. Eventually, the need for my involvement should be come less, as the system recovers its balance. My involvement creates a “balancing loop” (B1). Balancing loops add stability to systems and organizations.

Creating Causal Loop Diagrams

Causal Loop Diagrams help reveal system dynamics. Creating the diagrams involves more work than reading them, but can be done by anyone willing to take time to think things through and look for relationships. For example, what problems might arise by involving help? Is it possible that things will get worse before they get better? And why would that be?

The Buddy System

The first step in creating CLDs: find a buddy, friend or coworker with whom to share the process. When I started working with CLDs, I didn't pay attention to this. I've since noticed that I add something to every diagram that comes my way, and every diagram I send out comes back with meaningful corrections or contributions. This happens because everyone's world-view is slightly different. Most world views overlap enough that we can understand each other, but are different enough that other people will think of things you don't. If you'd like, you can send me your drawings for review and comment.

If another view-point is necessary, two may be better. But be aware that at some point adding people will create more "drag" on progress than contributing "thrust" towards completeness. In a work environment, completeness counts more than speed, so make sure you consider all viewpoints when developing CLDs. For independent work, I prefer "mostly complete" and quickness.

When sharing with your buddy, consider how you're going to present the CLD. I worked with a friend on the phone one time and tried to draw the CLD as he described his drawing. Unless you enjoy lessons in miscommunication, skip this and at least send some sort of a drawing (JPEG, GIF, PNG) they can view, print and mark up. I've found sending the actual drawing works best. For Windows users, I typically use Visio. I have a template for CLDs (using Jerry Weinberg's notation) that my friend Brian Pioreck created. When I'm working with Mac users, I use Canvas to create the CLDs. In both cases, we can change the CLD and share the results via email.

Creating a CLD usually follows these steps:

1. Bump headfirst, sometimes painfully, into a puzzling question.
2. Think about the question.
3. List some variables related to the question.
4. Connect the variables showing how one influences others.
5. Check the CLD by reading the story.
6. Repeat steps 2 - 5 as necessary and appropriate.

What was that?

Here are some questions that usually start my systems thinking mode. “What is really going on here?” “Why is this happening again?” “You mean adding help might make things worse?” “What possible negative impacts could there be?” “How could that happen?”

Thinking

Sometimes when I think about a problem, I can see the answer immediately. Other times I have to commit the problem to my subconscious and do something else until good ideas surface. I have favorite activities that involve my body, but leave my mind free (like bush hogging). Other activities (like Aikido, hitting golf balls) occupy all of me. Typically the more perplexing the problem, the more I need to focus on something else for a while.

Thinking about the question hopefully leads to a deeper understanding of the situation. If it doesn't perhaps the question is too small or too big. Perhaps the question isn't clearly stated. Try changing the question and see how it changes your thinking.

As I thought about the question, I remembered how sometimes the changes we make to help actually result in decreased results before things get better. This leads to the “worse before better dynamic”.

Variables

Now it's time to list the variables you've discovered in your thinking. Choose things that can change. "Number of Problems Solved" changes and influences other variables. I like to use variable names that work with "As [insert name] goes up (or down), then [downstream name] goes down (or up)." This helps me prevent putting constants in the model. Let's consider the following variables:

- Number of New People
- George's Training Load
- Number of Problems Solved
- Relative Progress

In my experience, getting all the variables in the first pass doesn't happen often. The point here is to look for hidden structures that create the observed dynamics. As the drawing takes shape, new variables appear, and old variables may not be needed. It depends on your point of view.

Which Comes First?

Since all the variables are related, and you can start reading a CLD at any point, it doesn't matter which variable gets put in the drawing first. Pick one and get started telling the story. Once in a great while I do a hand sketch. Generally I head straight to the drawing package where I can cut, paste, drag and drop. So far, the results of getting someone to help George looks like:

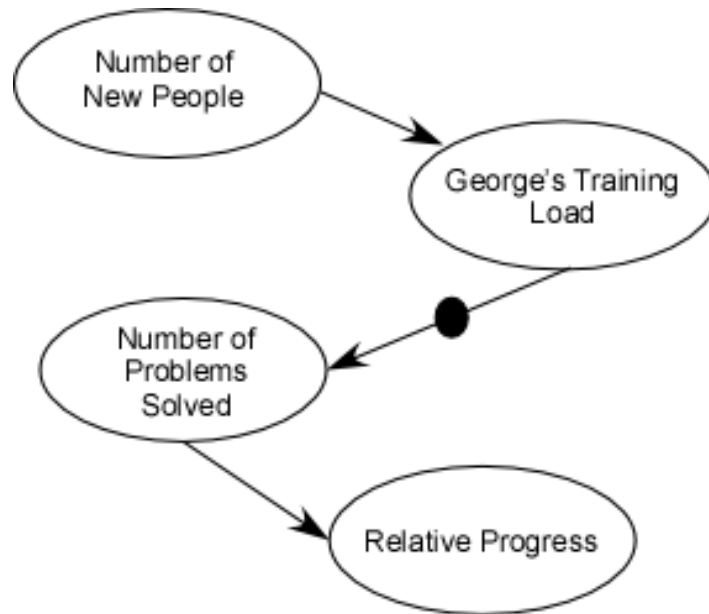


Figure 4: Adding Help

This means as more people are added, George will actually solve fewer problems in the immediate future so the relative progress goes down.

The Rest of the Story

As I draw the CLD, I find myself rethinking assumptions and conclusions. This restarts me iterating through the steps. Don't be surprised if it takes several iterations to become comfortable with the CLD. For example, Figure 4 doesn't include the results of George's training load. Adding Figure 1 and Figure 4 together we get:

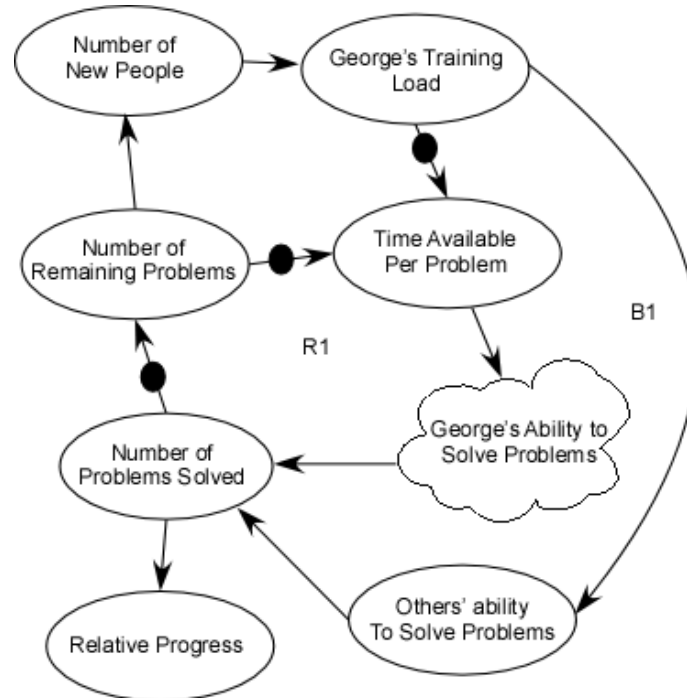


Figure 5: The Solution

This happened in this situation. The additional skilled effort increased the number of problems solved to the point where the defect rate fell below 1%. This put George back in the original position of being able to solve the problems as they occurred.

The Never Ending Cycle

If two people take the same variables and each create a CLD they may not produce exactly the same drawing as I do. That's OK. That's also why it's good to have other people review your drawings.

Causal Loop Diagrams give us a method for revealing hidden system structure. Being able to read and create CLDs provides a way to clarify our thinking and for sharing our current understanding.

The author thanks Johanna Rothman and Stuart Scott for their contributions to this article.

###

About the Author: For a quarter century, Don Gray has worked in applied cybernetics, starting in machine and process automation and migrating to organizational systems and change. His major areas of interest are cognition, modeling, and understanding system change. His work focuses on integrating people, projects, and processes. He is a host for the Amplifying Your Effectiveness conference. Visit his website at www.donaldegray.com.