Bonus! Scalability

Who’s Scale Measures Scalability?

Since 2000 I have had the great pleasure of sitting in as well as presenting several hundred (closer to a thousand) software demos. For those of you who do not know this, a software sales engineer or sales person is required by the unwritten law of software to use the word scalability in every meeting, pitch, demo or phone call. If they don't they will be black balled from the world of software sales. When I was demonstrating software I would get to the scalability point and say, "Ok, here it comes – the one word that every consultant has to say." I would pause to make sure the air was thick with anticipation. Then I would say… "It's scalable!" There would be groans and a little bit of a letdown. Then I would explain why it is scalable. However, in this world of software and software sales, I think sales representatives don't know what it really means and why it is important. Even if the sales person knows what scalability is, they take it for granted that the prospective client knows the term or knows the true impact of the term.

In this section you will find out the what, why and the how of scalability and what you need to know before you weigh in on its impact before you automate you accounts payable process.

Here is how the dictionary defines scalability:

"The ability of something, especially a computer system, to adapt to increased demands[1]."

That's a great working definition; the dictionary went on to explain:

For example, a central server of some kind with ten clients may perform adequately but with a thousand clients it might fail to meet response time requirements. In this case, the average response time probably scales linearly with the number of clients, we say it has a complexity of O(N) ("order N") but there are problems with other complexities. E.g. if we want N nodes in a network to be able to communicate with each other, we could connect each one to a central exchange, requiring O(N) wires or we could provide a direct connection between each pair, requiring O(N^2) wires (the exact number or formula is not usually so important as the highest power of N involved).[2]

My head hurt just a little from writing that definition. I imagine yours may be from reading it. However, the definition explains that the more data (of strain) you put on a system without the appropriate capacity the system will not work to capacity. (Furthermore) The problem with scalability as it pertains to an automated AP environment is the above definition is only one-third of the overall scope, which is "network" scalability. I have created a way to analyze each of the three aspects, which we will call "disciplines" as it relates to AP Automation. Later in this section I will give you a workable test you can administer to ensure that each discipline will not be a negative factor in your world.

[1] http://dictionary.reference.com/browse/scalability
[2] http://dictionary.reference.com/browse/scalability

Discipline 1: Network Scalability

For starters, let's go easy. As explained in the Dictionary definition, network scalability is pretty easy to understand. To make the definition somewhat entertaining as well as informative, I'm going to use a highway and cars as an illustration.

In Figure 2 the highway is open. Cars are moving at an appropriate speed, they can exit and enter and traffic crosses without a problem. This is just like a network. The road is the infrastructure and the cars are the data.

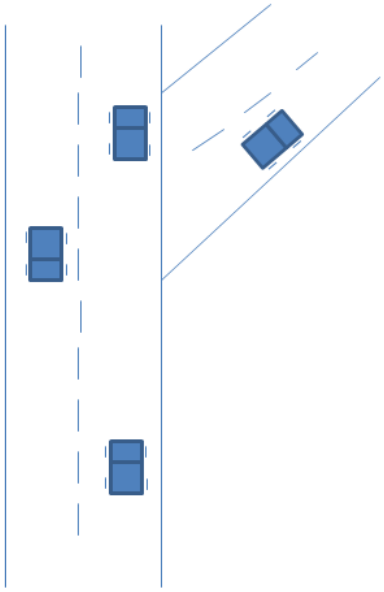

Figure 1

If the road stays the same and the traffic increases the cars (data) will slow down (Fig. 3). If the cars (data) continue to increase and the roads (network) do not, the cars (data) will slow down and sometimes come to a complete stop. (Right? – People in Atlanta or LA.)

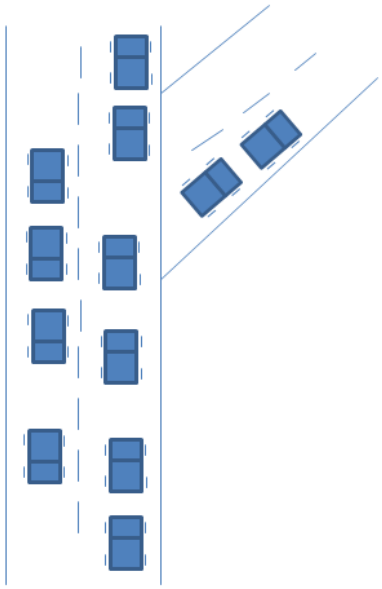

Figure 2

The problem is easily corrected when the road (network) is widened or more lanes are added (more bandwidth). This allows the cars (data) to travel without delay (Fig. 4).

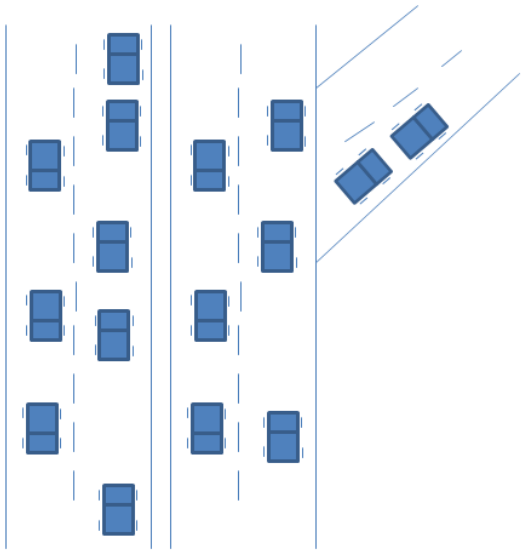

Figure 3

The widening of the road (bandwidth) is the addition of servers (RAM, Memory, Power, etc.) and a better connection to the internet (T1, T3 connection).

Discipline 2 – Code Scalability

With discipline 1, the situation becomes very frustrating when you add data, people, or load onto your network, and realize the network is not scalable. Add to the network (your servers or upgrade connection) just to find out that the speed is no better. This is where the second discipline and the most frustrating/hardest to correct comes into effect: Code Scalability, referring to the code within the software.

Using the same analogy of the road and the cars, the code is the engine that drives the car. The problem is twofold. First, the cars can break down causing a need to be fixed. A skilled mechanic (software developer) can trouble shoot and have the broken cars up and running. The second, and most difficult, problem to correct is cars driving in the wrong direction at the wrong time (Fig. 5).

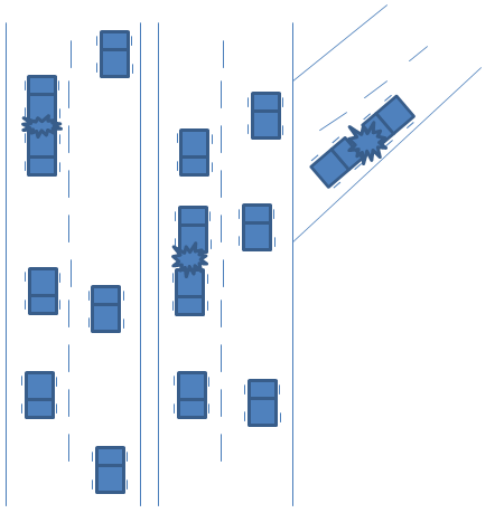

Figure 4

Fig. 1.4

Guess what will happen if cars on the freeway start heading northbound in a southbound lane. When you interact with a piece of software, it performs thousands of requests that create several actions per second. If the software is not programmed properly a request will get in the way of another request causing both requests to either wait on another request to sort the knot out or create an error that will not allow you to move on. If this happens, no network hardware or infrastructure additions are going to help code scalability.

Code scalability problems occur when a developer has no idea what they are doing or when a team is led by a person with little to no experience. To put it in more of a positive way; to eliminate code scalability problems the developer or leader needs to be very skilled on the platform in which they are building the software (road). It is critical for the developer to fully understand the goal of each process within the software. In the modern world, the developer has to have the ability to talk to people and interpret the problems they are trying to solve using software. Not wanting to put all the blame of communication on

the developer, the person articulating the problem needs to have an understanding of the realm of possibilities within the software.

Developer(s) as well as the problem fixer(s) need(s) to be in concert on possibilities as well as delivery. If one or both are out of bounds on either the goal/outcome possibilities or developers abilities; serious problems will occur. Continuing with the idea of the road, let's say you have a talented construction worker that is skilled at pouring a driveway, not building a road using asphalt. That person is approached by a developer that wants to put in a new road to access another town in a more direct route; however that route is on a 90 degree grade. What do you think the outcome will be? It will not be good. Same with solving a process problem, if the goal is unrealistic and the builder (developer) is not skilled and experienced, the outcome will be an untraveled road with lots of bumps.

Discipline 3: Process Scalability

This is a difficult one too, because, unlike the two previous disciplines scalability is 100% made by humans.

If the parts of the process are not outlined in the proper manner your highways will start creating circles

Sticking with the road, but expanding the idea, the cars are the software modules. A full software solution is made up of smaller pieces of software to manage particular processes and procedures. These smaller pieces of software are called modules (An example is within your email client you have the ability to search. The search function is a module). So, the modules are the cars, the road is the network and the process is the highway system. As you are aware our highway system is designed to take us from point A to point B in the quickest, safest, and most efficient manner. The same is true with the process that is outlined to automate accounts payable. Experience plays a key role in constructing the correct highway system. However, with process scalability the experience comes from (generally) non-technical sources, and the problems start because the sources can be (and usually are) numerous. The old saying, "Too many cooks in the kitchen" rings truer than ever. Here is a list of influences on the accounts payable process:

- Auditor
  - Internal
  - External
  - Customer
  - Investor
- Approvals
  - Budget (over or under)
  - Accuracy of the work
  - Product Delivered
  - Contractual
  - Against Purchase Order
- Coding

- Control
    - Owner
    - Department
- Accuracy of the Billing
- Vendor
- Risk
    - Duplicate
    - Non – Vendor
    - Scam – Fraud

This is not a complete list; however it starts to make the point about the horrors of process scalability. If the parts of the process are not outlined in the proper manner your highways will start creating circles. Figure 6 illustrates the correct highway system, with exits and sections created in a constant motion pattern.

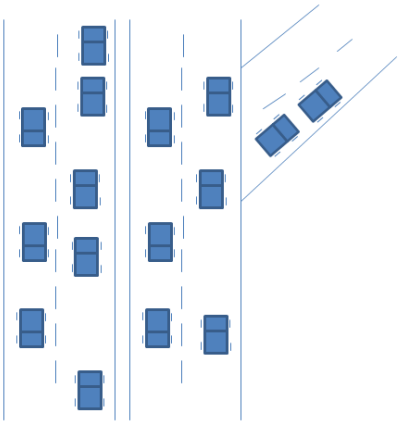

Figure 5

Figure 7 illustrates the dreaded principle of looping**.** Looping occurs when the same person needs to touch the same invoice multiple times in different steps of the process. Looping will kill the efficiencies of your process as well as eat up very large chucks of your processing time. This will lead to late payments, fire drills, and you can forget the ability to capture a vendor discount.

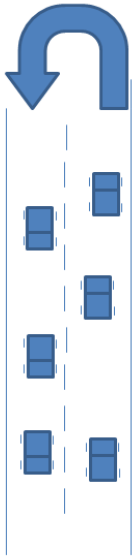

Figure 6

Understand that if there are loops in your process, then the process is, by definition, not scalable. Here is an example:

Approver 1, Greg, is an accounting person who will receive the invoice and code it against budget. Approver 2, Peter, is the approver of the invoice against receiving and the contract. Approver 3, Janet, is to approve against Sarbanes Oxley (SOX) compliance rules. After these three approvals the invoice needs to go back to Greg to be entered into the accounting system, then to Peter again for cash management. This process may work well when the invoice volume is low, but if you add additional volume the process will overwhelm due to loops and re-work (re-work is having to repeat the same work more than once. The difference between re-work and looping is re-work can be done by different people but it is the same work); therefore the process is not scalable.

So…now that you understand the disciplines of scalability: Network, Code and Process... why does this make a difference? The one word answer is money. The largest cost of an automation implementation is trial and error. The best way to combat trial and error is to create a system in the beginning that will grow with flexibility and not to bring you to a certain point and then require you to do something completely different.

Now it is time to tip the scale in your favor. Use the questions below to determine how scalable your (current or potential) automation process is.

Questions	Yes/No
1. Is your internet download and upload speed higher than 1000 kb? If you do not know, go to http://www.speakeasy.net/speedtest/ and find out.	

2. **Did the developer(s) or company who created your automation software have more than six years' experience with creating AP Automation? Based on experience, three years is the break point that understanding of AP Automation because less theoretical.** 3. **Is the maintenance of your network done by a person where that is their sole job?**
4. **Do the developer(s) or company who created your automation software do AP Automation exclusively?**
5. **Did a process designer help you develop your process?**
6. **Was your process designed from scratch instead of making your paper environment digital?**

Score: For every yes give yourself 1 point

Score	Scalability
6	Very Scalable – No Problems
4 or 5	Mostly Scalable with long term problems (over 10 years)
2 or 3	Somewhat Scalable with midterm problems (5 to 10 years)
0 or 1	Will not scale – Immediate Problems

The next time you have the privilege of participating in a software demonstration, go ahead and ask the sales person/engineer to step on the scale. It is perfectly acceptable to use the above disciplines to find out if their solution is going to help, hurt, or cost you in the long run.