Developing Toward an SLA: Understanding Data Complexity

Tom Wilson

Abstract

A *Service Level Agreement* (SLA) is a business operations concept that is sometimes inappropriately applied to engineering development. When an SLA is not converted to engineering requirements, development occurs blindly and testing is probably done against some of the values specified in the SLA. The result is higher risk and higher cost. This paper illustrates some concepts concerning data complexity that are relevant to the evaluation of the SLA and the performance test. Here, the complexity of the data is focused more on the relationships between data elements and less on the implementation of the system (i.e., the application and/or database subsystem). Obviously, an efficient system will have an organization of data that supports the real relationships.

1 Introduction

An SLA is an agreement between a customer and a service provider. The SLA specifies one or more service levels. Meeting a service level results in payments, or failing to meet a service level results in penalties, depending on how the SLA is defined. An SLA often applies to availability, performance, or some other quality of the system or service. Multiple SLAs can exist.

An SLA defines how the system must perform when in use, but does not directly define how a system should be designed. An SLA is a risk-vs.-reward concept. The idea is that the absolute (i.e., worst case) performance is intentionally compromised in order to lower total cost. Designing a system that meets some worst case performance requirements results in a system that is unnecessarily expensive because the system is not stressed most of the time. The compromise comes from lowering the performance goal so that performance is sufficient most of the time. The risk-vs.-reward aspect associates higher payments (or penalties) with a higher service level. Failure to meet the SLA is allowed, but is penalized.

When an SLA computes an average of measurements over a period of time, it is important to understand the behavior of those measurements. If the data can have a relatively large standard deviation (or some other measure of spread), then we might be less concerned with large individual measurements as long as other measurements can compensate. When this insight is absent, no tolerance exists for such variation. Over-engineering may occur, resulting in higher cost.

The previous installments of this series looked at two facets of evaluating an SLA. In [Wil10b], we looked at the variation of the response times across transaction types; this was a notional analysis. In [Wil10a]), we looked at the variation in average response time because of load; we also took a real transaction and looked at its response time distribution. In this paper, we want to look at the variation in response times because of the complexity of the data involved.

To understand the impact of data, let's define the factors that impact a transaction's response time. A transaction's response time is dictated by (1) the function that it performs, (2) the data on which the function operates, and (3) contention for resources. To understand factor #1, consider the difference between summing a list of numbers and sorting the same list. More work is involved in sorting compared to summing. To understand factor #2, consider the difference between sorting 100 numbers and sorting 1000 numbers. More work is usually required to process the bigger list. To understand factor #2 further, consider the difference between sorting 100 numbers that are already sorted and sorting 100 numbers that are not. The latter instance has more work to do. Not only is size a factor, but the actual content dictates performance. Factor #3 comes into play as resources are competed for. When insufficient resources exist, "waiting" is introduced and response times increase. We will be mostly focused on factor #2 and some aspects of factor #3, particularly when the waiting is a result of the data and not resources such as the CPU. Factor #2 is *complex* when there are many relationships between data elements. For example, a user may which to retrieve all work orders assigned to certain personnel (perhaps those requiring specific training) involving vehicles in a list of organizations that are deploying in a specified time period. We are concerned with replicating what users really do.

2 Evaluating an SLA

The system providing sample data supports logistics and maintenance for military equipment. For the purpose of anonymity, some general details are provided, and some specific details have been changed. This proprietary system has a response time SLA defined for it. For our purpose, the SLA is:

> The system shall respond to user requests with an average response time of 5 seconds for 98% of the transactions during the monthly charge period.[1] The database shall contain upto 5,000,000 items of equipment.

Other details exist in the SLA that concern the number of concurrent users and the sizes of other parameters in the database. These impact performance and will be discussed without explicitly defining them here.

What is missing from the SLA is a description of the growth during the operations lifecycle; only the final goal is defined. Nonetheless, since most of the risk lies with the contractor, understanding this growth is critical to a cost-effective system maintenance plan. The system is in a military environment, not a commercial one. Users will not be showing up when they hear how great the system is or when some interesting function becomes available.[2] New users will be introduced over time as their organizations are planned to be added. The users must be trained to use the system, and the organizations' data must be uploaded. A system maintenance plan should account for this growth behavior. The hardware used in the early part of the lifecycle may not need to handle the final load. Similarly, testing during development should understand the near-term goals. In other words, design requirements may not reflect the final system goals. Since the system should be scalable, the design should address the final goals.

In [Wil10a]), we discussed the relationship between a short performance test interval and the SLA's monthly charge period. In this paper, we also consider the short performance test interval as we focus on the data used in the test.

2.1 System Data

The reference system may not be typical of commercial systems, but it provides some interesting complexities that may have a counterpart in those systems. We need to discuss how the system users and data are organized. This is an important aspect of understanding the performance related to data.

People (both users and non-users) and equipment are assets, and they belong to organizations called *units*. All organizations form a hierarchy of organizations. This is basically a tree structure. Table 1 lists organizations that could exist in a hierarchy. The number of levels and the names are not important. The key is that there is a hierarchy defining relationships between the levels.

Table 1: Organizational Levels

Level	Other Names or Examples
Service	Air Force, Army, Navy
Fleet	Theater, Region
Corps	Group
Division	
Brigade	
Regiment	Battalion, Wing
Unit	Squadron, Company, Battery, Platoon, Section

Unfortunately, the term *unit* is going to be overloaded in this discussion. A *unit* is a general term for an organization. It is also used to describe a level in the hierarchy. So, one unit can exist at the unit level and another at the regiment level. Higher level organizations may or may not have assets in them. They will certainly contain other organizations. For example, a regiment consists of a few units, but the regiment itself may or may not contain people and equipment.

Figure 1 illustrates a notional brigade. The brigade is made up of 30 units. The units shaded purple are at the brigade level and have visibility into all units within the brigade. For each regiment (Regt 1-4), the units shaded yellow are at the regiment level and can see all of the units within the regiment, but not units in other regiments. Units shaded brown have no visibility outside of their units. A unit's visibility is equivalent to its organizational level.

[1] This means that after the top 2% of response times are eliminated, the remaining 98% must have an average that is less than or equal to 5 seconds.

[2] Having no competition has both advantages and disadvantages.

Brigade A					
HQ	**Eng Corps**	**Regt 1**	**Regt 2**	**Regt 3**	**Regt 4**
Brig A HQ	Eng Corps 1 HQ	Regt 1 HQ	Regt 2 HQ	Regt 3 HQ	Regt 4 HQ
	Eng Corps 1 CS Sqn A	Regt 1 Eng Sqn	Regt 2 Eng Sqn	Regt 3 Eng Sqn	Regt 4 Eng Sqn
	Eng Corps 1 CS Sqn B	Regt 1 Sqn A	Regt 2 Sqn A	Regt 3 Cpy A	Regt 4 Bty A
	Eng Corps 1 GS Sqn A	Regt 1 Sqn B	Regt 2 Sqn B	Regt 3 Cpy B	Regt 4 Bty B
	Eng Corps 1 GS Sqn B	Regt 1 Sqn C	Regt 2 Sqn C	Regt 3 Cpy C	Regt 4 Bty C
		Regt 1 Sqn D	Regt 2 Sqn D	Regt 3 Cpy D	Regt 4 Bty D

Figure 1: This figure illustrates a notional brigade, named Brigade A, consisting of 30 units. Purple units denote visibility into all 30 units. Each of the 4 regiments consists of 6 units. Yellow units denote visibility into all 6 units within a regiment, but not visibility into the other regiments. Brown units denote units with no visibility into any other unit. Abbreviations in this figure are defined in Table 4 at the end of this paper.

The impact of the hierarchy is the amount of data that can be referenced by the users in the system. When a user at the brigade level looks at the status of the equipment in the brigade, he gets a much longer list than a user at the squadron level that is looking at his squadron's equipment. He also competes for the same data as other users who have access to the subordinate units. However, just because a user has visibility to all of the assets does not mean that he always references them. For example, a user in a regiment may look at the equipment in a specific squadron.

Table 2 shows a breakdown of the organizations and vehicles for some existing data. The first thing to note is that almost half of the organizations have no vehicles. There are probably not that many in reality. It is more likely that the data are not yet present or the vehicles are being transferred. We are going to have a hard time performing a thorough analysis if the data do not make sense. The "Veh. per Org." field shows the average only for organizations with vehicles. While we did not mine people counts, they are likely to be closely related to the vehicle counts. Similarly, there may be a relationship between non-vehicle equipment and vehicles.

Table 2: Number of Organizations and Vehicles by Organizational Level

Org. Level	**# Orgs.**	**# Veh.**	**# Orgs. w/Veh.**		**Veh. per Org.**
			$= 0$	> 0	
Fleet	2	10	1	1	10.0
Corps	3	0	3	0	0.0
Division	24	5,492	14	10	549.2
Brigade	101	13,117	49	52	252.2
Regiment	367	23,935	104	263	91.0
Unit	1,410	40,579	613	797	50.9

We can conclude several general things from Table 2. There are a few higher-level organizations. When they have equipment, they sometimes have a lot. There are many lower-level organizations. When those organizations have equipment, they do not have a lot. Nonetheless, the total amount of equipment at lower-levels is significant. When one understands the organization of armed forces, this all makes sense.

For the subsequent discussion, organizations with 0 vehicles were removed. Figure 2a shows a boxplot for each organizational level. Figure 2b shows the same data, but the y-axis is limited so that the largest box fills the plot. Figure 2c zooms in even further to show the smaller boxes in more detail. The general conclusion is that the distribution of equipment is not uniform. We have the same conclusions that we made when discussing Table 2.

Figure 3 shows the distribution of equipment for each organizational level using histograms. Only the brigade, regiment, and unit levels are shown. For levels above regiment, there is insufficient data to conclude shape. However, histograms for the regiments and units suggest an exponential distribution. Because we have a skewed distribution, we should not use the averages shown in Table 2 to summarize the data. The performance of a transaction may vary

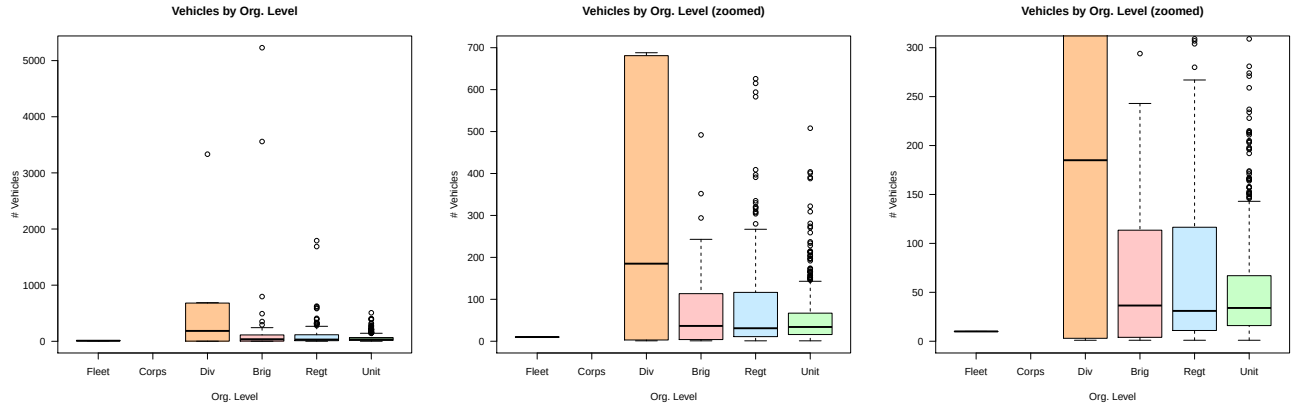

Figure 2: This figure shows the number of vehicles in each unit broken down by organizational level. Charts (a), (b), and (c) show the same data, but limit the y-axis to different maximums.

greatly depending on the organization referenced.

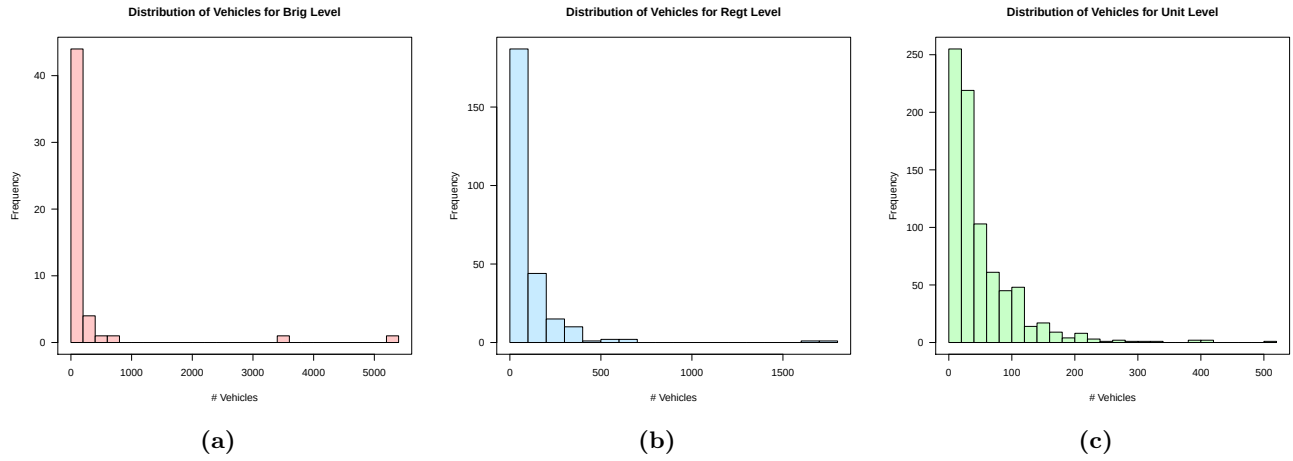

Figure 3: This figure shows histograms for the number of vehicles in each unit for the (a) brigade, (b) regiment, and (c) unit levels.

Not all service organizations are the same: air force, army, and navy. Even within a service, there is variety in the type of unit: armored, artillery, engineering, infantry, and medical. The type of unit is also a factor in determining its composition: the number of people, the number of users, and the amount and type of equipment.

What is the conclusion that we can draw at this time? Units do not have the same composition. Blindly selecting units for a test may cause the test to experience either extreme (i.e., too little or too much data). Units also have different visibility. This drives how much data *could* be accessed.

2.2 Data Scope

Let's look at two plausible arguments that may not be true. First, larger units have more people and, therefore, more users. We would expect larger units to be more frequently referenced. Secondly, lower-level units have users that spend more time off the system because they are "hands-on" maintaining the equipment; such users are not on the system frequently. Higher-level units may be managing their visible assets via the system with a higher frequency.

Arguments such as these can be made, and test scenarios created that align with them. But what do real data suggest? We have a problem here if the system's logged data cannot tell us what we want to know. Unfortunately, this is the problem we have. So, can we even justify any data for a test?

We can easily determine the organizational level of any user on the system. Table 3 shows the distribution of users' organizational levels during time intervals where the system was loaded. There are very few high-level users. We can also figure out which unit a user belongs to (although we did not actually do this). What we cannot tell is what data the user accessed in a straightforward automated way.

Table 3: User Organizational Level Frequency

Org. Level	Freq.
Fleet	0.2%
Corps	0.1%
Division	0.4%
Brigade	2.4%
Regiment	38.5%
Unit	58.4%

However, in our testing we control what data the user accesses. Figure 4 shows two charts for the same data from a test. In Figure 4a, the transaction performance is broken down by the organizational level of the user generating the transaction. We assume that the variation across the organizational levels is because of the scope variation. However, we generally cannot tell what data the user actually referenced from any logged data. This view of the data is misleading, just as Table 3 is.

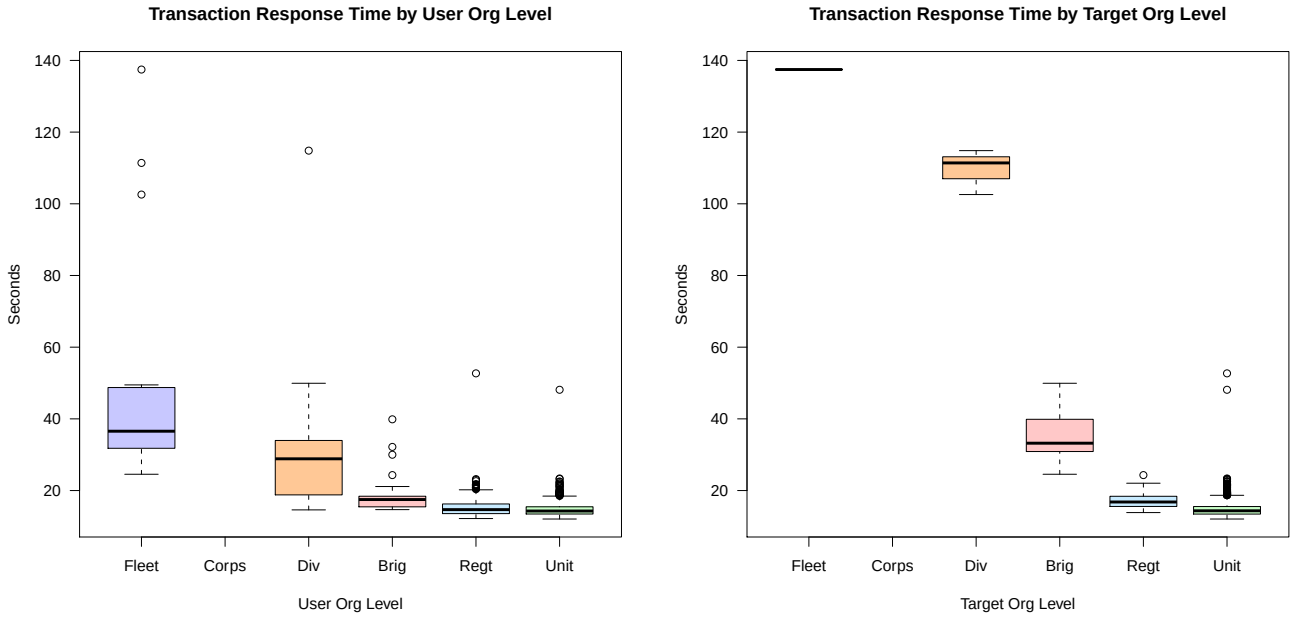

Figure 4: This figure shows the same transaction data broken down by the organizational level of (a) the user generating the transaction and (b) the targeted data.

In Figure 4b, the transaction performance is broken down by the target organizational level specified by the transaction. Again, we know this only because we created the test data. The variation is more definitive than before. Unfortunately, the results for this test are also impacted by load (refer to [Wil10a]). Nonetheless, it is apparent that the data scope is a factor. We must accurately represent the accessed data if we want to understand performance.

The transaction providing the results in Figure 4 generates an inefficient database query. A significant amount of database CPU resource is consumed by the query, as well as logical and physical database I/O. We learned from our testing that too many high-level organizational data references caused a significant performance degradation in many other transactions. So, we decreased the number of references to high-level organizational data. But, what will real users access? We do not know.

Data contention arises when multiple users access the same data. If we create a performance test where all users

and data are within Brigade A, two conflicting issues exist. (1) Many of the data references will encounter contention as users try to read and write the same areas of the database. (2) Most data end up getting cached, reducing the amount of I/O. The overall result is a confusing and possibly unrealistic test.

The system has the potential to reference long lists of data (e.g., all of the vehicles for a corps-level organization). The system is designed to return pages (or chunks) of the entire list. But how many pages will the user request before learning what he needs to know? The likely guess is "half of them". A test needs to account for this.

Once we understand how users realistically reference data, then we can have more confidence in evaluating the system, particularly the actual organization of data in the database. The reference system uses a database abstraction layer that isolates the application from the database implementation. While this has development advantages, performance should suffer. Having realistic data references will help us understand how good or bad performance really is. Having unrealistic data references will cause us to optimize the database improperly.

2.3 Database Size

When the database grows, it is mostly a result of new organizations being added, rather than existing organizations getting larger. The exception is when existing equipment is going to be managed by the system when it was not being managed in the past. Most organizations do not vary greatly in size over time. Similarly, a sports league grows when teams are added to it, rather than teams increasing their numbers of players.

Figure 5 depicts how the number of organizations will grow over time. The green region represents the state at a time early in the operations lifecycle. As time passes, more organizations are added (the right boundary of the green triangle sweeps to the right), with more low-level units being added than high-level units. Likewise, the database and number of users are growing. A test representing the green region would only contain data for the organizations that exist. Only users in those organizations would exist. A test representing the entire region (green and blue) would contain data for all organizations. The users will be appropriately dispersed across the organizations. If the latter test restricts the users and data to the green region, the results might be misleading.

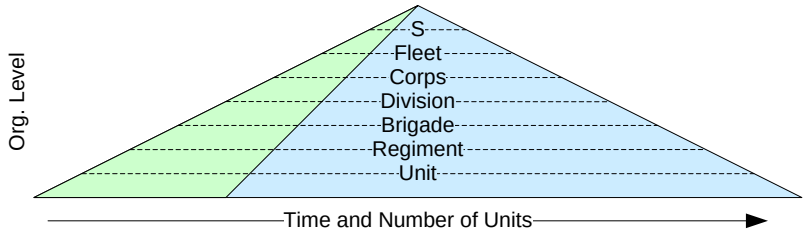

Figure 5: This figure illustrates how both the database and number of organizations will grow over time. The green region shows the state at a time early in the operations lifecycle. (S=Service)

Figure 6 shows the relationship that should exist between the maximum number of users and the maximum number of vehicles (and, thus, the maximum database size). Holding the number of users constant while varying the database size (indicated by the red dashed line) is a possible test, but it is not a realistic scenario anticipated during growth. A similar situation exists for varying the number of users while holding the amount of equipment constant (indicated by the blue dashed line). This at least accounts for different time periods where the number of users varies (refer to [Wil10a]), although it should never surpass the green line (e.g., the black dashed line). If the system must handle a certain maximum number of users when the database is at its maximum size, the number of users will be smaller when the database is smaller. Early in the operations lifecycle, performance should be assessed against the appropriate number of users and database size. Hardware and software upgrades may be necessary to achieve the maximum numbers. By that time, real workload data will exist. The growth line might be a curve, but the conclusion is the same.

Figure 7 shows the performance of a specific transaction for two tests where the database size varies. The small database has 0.5 million items of equipment; the large database has 5.0 million items. There are 30 unit-level users, 25 regiment-level users, and 1 brigade-level user executing this transaction; there are other users executing other transactions. As previously mentioned, holding the number of users constant (which we did do) results in a misleading comparison. If the number of users is the same, then we are comparing a heavily-loaded period on the left and a lightly-loaded period on the right. Was that the goal? Nonetheless, the sets of users and the targeted data should not be the same. Other transactions may not reflect this amount of degradation. A good database design and efficient queries may have near-constant performance. However, neither aspect exists in this system. The database design and the quality of the queries reflect the importance of development attributes (e.g., portability,

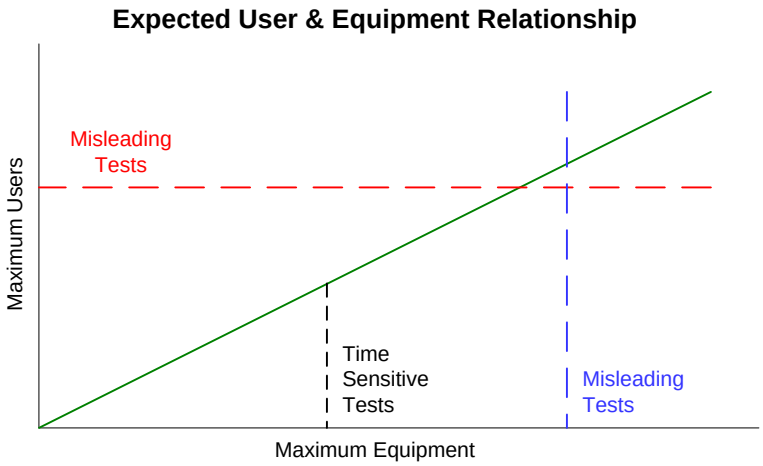

Figure 6: This figure shows the expected relationship between the database size or amount of equipment) and the number of users. Performance tests that fall along either dashed line, but not on the solid line, may not be realistic situations.

maintainability, cost, schedule); various system attributes (e.g., performance) are a lower priority. This is acceptable and must be understood, so that the hardware can compensate.

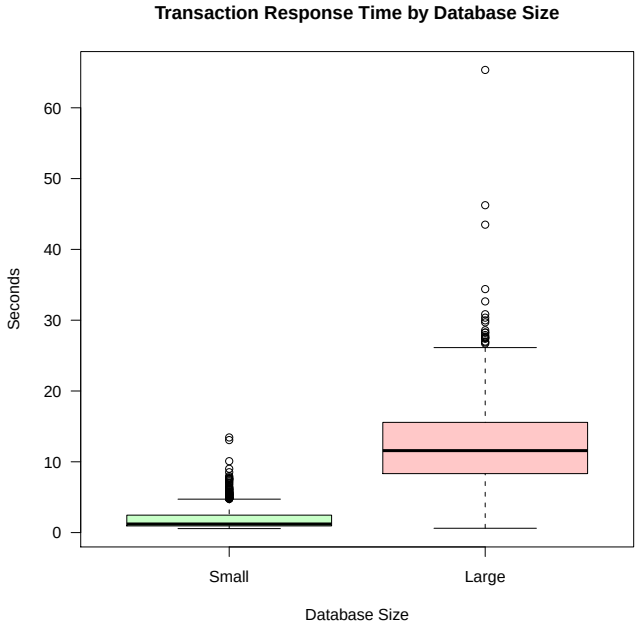

Figure 7: This figure shows a boxplot for the response times for a specific transaction. Two independent tests were performed with small (0.5 million items) and large (5.0 million items) databases. The same users exist in both tests. The same data are referenced in both tests.

For users in low-level organizations, the size of the database should be irrelevant because they can only access their organizations' data, and these organizations do not vary in size with the database. For users in high-level organizations, this is not true. However, viewing the entire high-level organizations is not a likely operation. So, the size of the returned data should be fairly consistent for a growing database. Contention should not increase significantly. Inefficiencies in the database design can defeat this principle.

3 Conclusion

This paper discussed how the targeted data affect performance. In the absence of instrumentation that helps us mine how the users access data, we need to do some significant study of the system's concept of operations and users' anticipated behavioral patterns. In this case, a key was understanding how the data are organized (i.e., the hierarchy of organizations). We also investigated how the numbers of vehicles in units are not uniformly distributed, and speculated that the same was true for the number of people and users.

We also discussed the operations lifecycle of the system and how the database was anticipated to grow. A relationship between the size of the database and the number of users was argued. Understanding these lifecycle aspects is key to creating a performance test that will help us understand the performance expected in real operations.

Abbreviations

Table 4 contains the abbreviations, with the corresponding definitions, that occurred in the figures, tables, and text of this paper. Abbreviations that were explicitly defined in the text are not included.

Table 4: Abbreviations

Abbreviation	Definition
Brig	Brigade
Bty	Battery
CPU	Central Processing Unit
Cpy	Company
CS	Close Support
Eng	Engineering
GS	General Support
HQ	Headquarters
I/O	Input/Output
Org.	Organization
Regt	Regiment
Sqn	Squadron
Veh.	Vehicle

References

[Wil10a] Tom Wilson. "Developing Toward an SLA: Understanding the Testing Interval". *CMG MeasureIT*, October 2010.

[Wil10b] Tom Wilson. "Developing Toward an SLA: Understanding Transaction Performance". *CMG MeasureIT*, July 2010.