APPDYNAMICS

An AppDynamics Business White Paper

# Bringing development and operations together with real-time business metrics

It may be that developers and operations work at the same company, but historically they have had different goals, different strategies, and different ways of measuring success. Developers are interested in writing code that is designed according to the best design patterns, that is high performing and scalable, and that is extensible enough to allow them to quickly release new features to satisfy business requirements. Development measures success by code quality tests, performance and load tests, and the rate at which they can release new features. Operations, on the other hand, is concerned with ensuring that applications are highly available, that outages are infrequent, that they are best utilizing their physical and virtual environments, and that the Mean Time To Resolution (MTTR) of outages is short. Operations are responsible for the infrastructure and ensuring that applications are running and available - and their success is measured as such.

*If we add an additional "9" to our uptime, how does that affect the bottom-line?*

So while the business has a single goal, such as increasing profitability or helping the most people, and both development and operations have their roles in this goal, it is wholly possible for development and operations to be individually successful but for the organization to fail. It would be far better if we were able to measure the success of the company and use that as a gauge to assess the effectiveness of development and operations. For example, if we are able to release new features weekly, is the effect positive or negative on the performance of the company? If we add an additional "9" to our uptime, how does that affect the bottom-line? We certainly need to measure the effectiveness of individual groups in a company, but that measurement should be combined with real-time business intelligence to determine the health of the company as a whole.

This paper explores the concept of measuring business metrics in real-time so that you can accurately assess the performance of the company whilst also effectively managing the performance of the application and infrastructure. The goal is to assess the impact of development and operational initiatives against business metrics to determine the overall affect of those initiatives. From this information you can answer the question of whether or not a particular initiative is positively affecting the business.

## Business metrics

What constitutes a business metric and how to we find them? In short, a business metric is a measure of something important to the health of your business. If you are selling products on a website then metrics such as the number of orders placed per hour and the number of dollars earned per hour would be good businesses metrics. If you are tracking an accounts payable system or another type of accounting software then you might want to capture the number of dollars moved between accounts per hour. The point is that it should be possible to track and quantify the financial impact of your online application and measure that financial impact along side other operational metrics so that you can accurately assess, not only the health of your application environments, but also the health of the company as a whole.

Consider the effect of introducing added efficiency into your development organization so that you can reliably release new versions of your key applications weekly. Does this have a positive or negative impact on your business? How do you know? We can assume that shorter release cycles enable you to deliver new features to your customers more quickly, which fosters loyalty and improves the user experience, but has it increased the number of orders or the bottom-line? Are users able to be up sold additional products? Are they able to accomplish their businesses transactions in less time? The only way that you will know is to define the business metrics that impact your company, measure them, and then analyze them to determine the impact of changes on these businesses metrics.

*What is the impact of increasing availability from 99% uptime to 99.9% uptime?*

On the operational side, do you know how much an application slowdown, or outage has cost you? What is the impact of increasing availability from 99% uptime to 99.9% uptime (see figure 1 for more information)? If operations is able to reduce their Mean Time To Resolution from an hour to 30 minutes, how does that affect the business? Is it worth adding additional unused capacity to your environment in the name of resiliency? Operational goals should likewise be tracked against businesses metrics so that you can quantity the impact of operational initiatives.

| Availability % | Downtime per year | Downtime per month* | Downtime per week |
|---|---|---|---|
| 90% (“one nine”) | 36.5 days | 72 hours | 16.8 hours |
| 95% | 18.25 days | 36 hours | 8.4 hours |
| 97% | 10.96 days | 21.6 hours | 5.04 hours |
| 98% | 7.30 days | 14.4 hours | 1.68 hours |
| 99% (“two nines”) | 3.65 days | 7.20 hours | 1.68 hours |
| 99.5% | 1.83 days | 3.60 hours | 50.4 minutes |
| 99.8% | 17.52 hours | 86.23 minutes | 20.16 minutes |
| 99.9% (“three nines”) | 8.76 days | 43.8 hours | 10.1 hours |
| 99.95% | 4.38 days | 21.56 hours | 5.04 minutes |
| 99.99% (“four nines”) | 52.56 hours | 4.32 minutes | 1.01 minutes |
| 99.995% | 26.28 minutes | 2.16 minutes | 30.24 seconds |
| 99.999% (“five nines”) | 5.26 minutes | 25.9 seconds | 6.05 seconds |
| 99.9999% (“six nines”) | 31.5 seconds | 2.59 seconds | 0.605 seconds |
| 99.99999% (“seven nines”) | 3.15 seconds | 0.259 seconds | 0.0605 seconds |

*Figure 1. The downtime permitted by various percentages of application availability*

Because operations and development see the world differently, we can help align their individual views of success and failure by assessing success and failure using a common set of metrics. If they share the goal of increasing revenue, optimizing the amount of time users spend accomplishing their tasks, or even helping more users find the closest food bank, then they can share a common language, which encourages better collaboration. The key task is then to identify the core metrics that affect your business, capture them, analyze them in real time, and report on them to your operations and development teams so they can better understand the business impacts of their work. If we observe that cutting release cycles down from monthly to weekly positively impacts business metrics then we know it is a good idea, but if the effects are negative then we have to ask the question of whether or not it is worth the additional effort.

*The key task is then to identify the core metrics that affect your business, capture them, analyze them in real time, and report on them to your operations and development teams so they can better understand the business impacts of their work.*

## Real-time analysis

You may have noticed that I slipped in the word "real-time" into the analysis statement above. Consider the release of a new version of your application that inadvertently breaks your application's checkout functionality for a certain browser. It may be that you have very patient users that try multiple browsers but it is more likely that you may lose sales. And at the very least, the rate of users checkouts will decrease. From a high-level performance management and monitoring perspective, you may not notice there is anything wrong until you receive a customer support call because you still see all of your business transactions running and they are all still responding within their acceptable thresholds. If your users do not alert you to the problem then you'll only notice a drop in revenue. What can we do to identify this problem earlier so that we can avoid losing revenue?

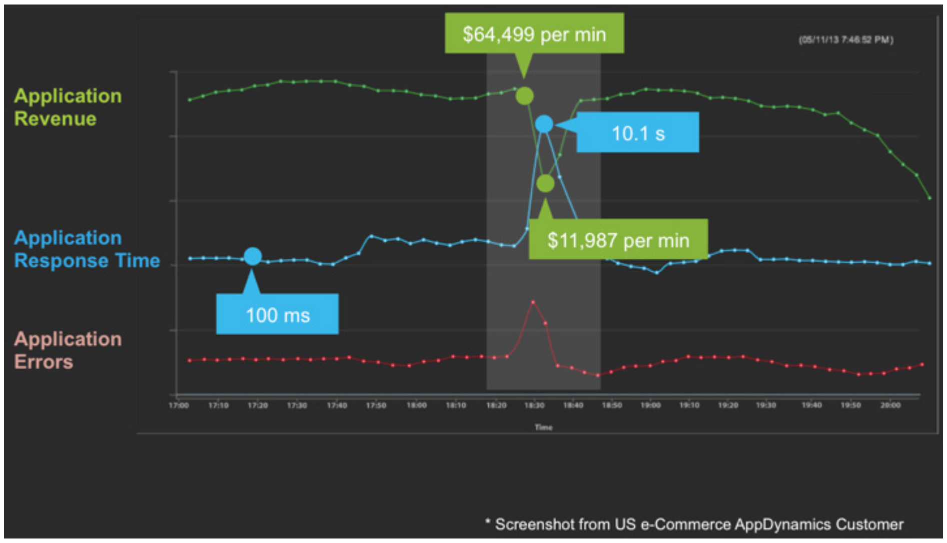


*Figure 2. An example that measures application behavior (response time and error rate) along side the revenue the application is generating*

The most direct way to isolate and alert the issue, would be to capture the associated business metrics in real-time and compare them to the 'normal' expected behavior for these metrics. For example, if we typically see $5k per minute of web sales generated on Monday lunch at 12:00pm and on this particular Monday we only see $1k per minute then an automatic alert should be triggered to start investigating the problem.  The importance of analysis in "real-time" is that learning about a problem such as this at the end of the month can result in a large and unnecessary loss of revenue. Real-time analysis can be used to identify problems as they happen.

There are however a number of challenges associated with capturing and responding to real-time metrics, such as how to process and analyze the data in a way that makes it immediately actionable. The key to this is example is to use automatically generated baselines in order to determine normalcy, which is what we will review in the next section.

## Business metric baselines

A key aspect to identifying when things are behaving abnormally is to first determine what constitutes normalcy. While this may seem simple, in actuality it is rather complex. We define “normalcy” with a baseline, which should be defined based on your business’s specific user behavior. Baselines can be defined as follows:

- All time average (over a time period): If your application is used uniformly every day and every hour of the day then you can define normalcy as the average value for a specific period of time, such as the average response time for the past 30, 60, or 90 days
- Hour of day: If your application is used variably by the hour of day, for example your see a spike in usage at 10am every day and dips in usage at 2am, then you would want to define normalcy as the average value per hour for the current hour of day
- Day of week: If your application has weekly variance, such as more usage on the weekend, then you would want to define normalcy as hour of day on the specific day of the week (compare Tuesday at 10am to the historic behavior for Tuesdays at 10am)
- Day of month: If your application has monthly variance, such as a banking application that might have spikes on the 15th and 30th of the month, then you would want to define normalcy as the hour of day on the specific day of the month (compare the 15th at 9am to previous month’s 15th at 9am)

After identifying the type of baseline that matches your business usage, the next step is to capture your business metrics, compute your baselines, and analyze current usage against the baselines.

## Capturing business metrics

From where you will capture your business metrics is really dependent on your application and how you model your solutions, but most likely you will have a method or methods in your application that, at the lowest level, contain the data you need to compute your business metric values. You can capture your business metrics manually or through an automated process:

- Augment your application code to publish the metric values to a database, a log file, or another service that you develop: the benefit to this approach is that you best know your application and can identify what metrics you want to capture so the effort is very targeted. The drawback is that it is invasive because you need to update your application to capture business metrics and, if you want to change them later, you need to re-release your code. Furthermore, you are responsible for computing the baselines, analyzing the current behavior against the appropriate baselines, displaying the behavior in a dashboard, and alerting to variations from the baseline.

*If your users do not alert you to the problem then you'll only notice a drop in revenue. What can we do to identify this problem earlier so that we can avoid losing revenue?*

- Automated solution: if you have a performance management solution in place, it would be far better to configure it to capture these metrics for you and to display them along side performance metrics. The benefits to this approach are: you do not need to change your source code to capture business metrics, new business metrics can be captured without needing to redeploy your application, you can correlate business metrics with performance metrics, and the performance management solution can derive the baselines for you and compare the current behavior to the appropriate baseline.

Regardless of where the business metrics come from, here are some suggestions for good business metric candidates:

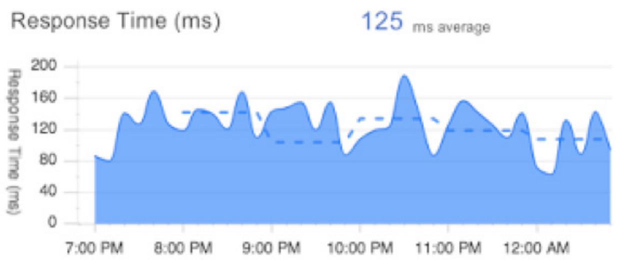
- Order or payment amounts: assuming that you have a method in your application that is invoked with a payment amount, capture the amount from this method call
- Categorization: if you break payment amounts down by categories, e.g. VISA customers versus American Express customers or premium versus standard customers, then group your business metrics as such
- Number of orders: simply capturing the number of orders for a given time period and comparing that value to your baseline can alert you to underlying problems; this might be accomplished by counting the number of times a process order method is invoked

## Detecting abnormal behavior

At this point you have your baseline and you have your business metrics, so how do you compare them? There are different strategies in determining when things have gone astray:

- Fixed SLA: while you don't need a baseline to analyze current behavior against a fixed SLA, defining fixed SLAs is one way of detecting problems. In this strategy you might determine that between Monday and Friday 10am-6pm you think it would be abnormal if you saw less that $5K sales in any given hour. If your usage is consistent then this strategy might be valid. The challenge in this strategy, however, is that more subtle issues, such as a drop from $8K to $6K may not be detected and, if your usage patterns are not uniform, then this may result in false positives.
- Percentages: you can choose to analyze the current value of a metric against a percentage of a baseline. For example, you might determine abnormality to be greater than or less than 20% of the baseline value. This strategy enables you to work with variable values and different usage patterns, but the choice of percentile does not necessarily account for the variance in usage data.

- Standard Deviation: probably the most accurate analysis for most data, you canis to define thresholds that alert based off of a number of standard deviations from the baseline mean value. In a normal distribution, 95% of all values will fall within two standard deviations from the mean and 99% of all values will fall within three standard deviations from the mean. Therefore alerting when a value is two or three standard deviations from the mean is a good indicator that things are abnormal. But this does not work for all data, such as highly variable data.



Whatever strategy you use to detect abnormal behavior, the process is now: capture business metrics and build a baseline that matches your usage patterns, compare the real-time behavior of your application against the baseline, and if your abnormality conditions are triggered then dive into your application and your performance metrics to identify the root cause of the problem.

## Bringing operations and development together

Once you have captured business metrics and have them available to report against, it is time to complete the circle and integrate business metric goals into operation and development goals. Now obviously we cannot task developers with increasing sales or operations with increasing user loyalty, but we can define specific development and operation goals targeted as improving these business metrics. For example, we might task development with reducing the number of pages required for a user to check out or to reduce the response time of the checkout process, and then see whether or not that increases the number of orders or even the user experience satisfaction rating. Likewise we might task operations with making the environment more elastic (can scale up and scale down to meet the user load) to (1) save hardware costs and (2) ensure that there is enough capacity for the user load at peak times, and then compare the result of that initiative to user satisfaction ratings.

The specific goals will be dependent on your business, but I hope you get the idea: there are quantifiable business goals that can improve the overall health of your business and you need to find a way to integrate those business goals into your development and operation goals.

## Conclusion

Any company that leverages technologies has two arms of its technical strategy: development and operations. Development builds products, integrates solutions, and in all other ways leverages technology to solve business problems. Operations takestakesOperations take what development has built and makes it available to the company's customers in a highly available, high performance, and low cost way. Both arms have the same goal: the success of the company, but they measure their own individual success differently.

This paper suggests that companies should leverage the concept of real-time business metrics to bridge this gap. A business metric is any important metric that identifies the health of the company as a whole, such as the number of orders placed or amount of revenue generated in an hour. Business metrics should be captured in real-time, analyzed against a baseline tailored to the company's usage patterns, and alerts should be generated if things go wrong. Finally, business metrics should be integrated into development and operation goals so that when development or operation initiatives are employed, their efficacy can be assessed by their impact on the business itself.