Consequences of Poorly Performing Software Systems

Poorly performing software systems can have significant consequences to an organization well beyond the costs of fixing the software. Poor performance can hinder employee effectiveness, degrade customer experiences, reduce process efficiency, create work-arounds, divert resources, and impede many aspects of overall business performance. It is a mistake to treat a potential performance problem as if it were the same complexity as a coding error. Coding errors are relatively easy to fix; performance issues often require extensive reengineering. This article will explore the challenges, costs, and mitigating solutions in addressing poor system performance.

The following typical scenarios can occur when business applications exhibit poor performance that impacts the user experience:

1. Enterprise software releases, for Production, that are scheduled either monthly or quarterly can be delayed due to performance engineering challenges during the application design and development phases of the SDLC, for instance;
    a. Application coding not meeting the performance goals
    b. Technical architecture components not working as planned
    c. Dependency on other systems, where the other systems are not ready
2. Performance issues due to increasing business volumes, or system consolidation
    a. Production systems are processing more volume and the rate of increase is not monitored (not watching the system)
    b. Workload of consolidated systems not fully understood
    c. Business volume exceeds original design goals
3. Performance issues due to flash or sudden increases in business volume
    a. Planned and limited time marketing events
    b. System not designed for such events and system is not monitored during the event
    c. The risks to business was not articulated and limited performance testing occurred
4. Performance issues due to new channel of business transactions
    a. New business partner
    b. New frequency of communication to or from partners
5. Performance issues during system recovery (the system is recovering from a significant error)
    a. Pending build-up of transactions due to queuing in the system that flood down stream systems when recovered
    b. Large user population waiting for system and just waiting to hit the send button

Release-based performance issues

Many large companies require constant changes in their software systems as these systems support many different business processes. For instance, a brokerage platform consists of various interdependent subsystems, each doing their part in the flow of an equity trade. The processes and systems are grouped into product lines that are managed as a few large units of

functionality. Most often, they are systems of systems. When one system changes, others that interact with the system must be tested, and then the newly revised system must be released into production. The systems will be on different hardware and software platforms.

Businesses typically manage these changes by building a detailed schedule and a release calendar. The release schedule can be monthly, bi-monthly, quarterly, etc.

Common performance characteristics of a release-based schedule are:

- Immediate occurrence of performance issues after a release, for either online, or batch
- Delayed occurrence of performance issues due to the ability to enable functionality at a later date

In either case, immediate response is required to stop the interruptions. This is very disruptive to teams, as people will be redeployed to solve the production problems and taken away from their normal tasks. Production becomes a priority, and other tasks are slowed down or suspended until the issue is resolved and a fix implemented.

The performance disruption could be a scalability problem and possibly solved easily by adding servers to the system, or a database issue as the database size is long past the original intent and goals of its design. Database issues tend to be difficult, requiring more resources to solve. The problem could be design and coding practices resulting in too chatty an application, calling the web service or database too often, or it could be coding practices that use polling instead of an event model.

Increasing business volumes

When an increase in business volumes occurs, if the application and system is being monitored, people can see a performance issue building. As the business volume increases, the response time of key business transactions start to degrade. For instance, viewing the history of an account may now take four seconds instead of two seconds. The customers' experience on the web site might be moving from *satisfied* to *tolerated*, or may be on its way to *frustrated*. The nightly processing tasks may now complete 30 minutes later than they did six months ago. You must gain visibility and get ahead of the upcoming performance issue.

To help with this, you need software in production that can provide insight into the nature of the looming performance issue. This software can help you get to the root cause of the issue, allowing you to spend the right amount of resources to remedy the problem.

If not monitored, the pending performance issues will become a sudden and immediate issue, requiring the use of unplanned, unscheduled and unbudgeted resources to solve. There are a number of application performance monitoring and management tools that measure the transactions from the user perspective. They allow you to proactively monitor response times while providing a detailed transaction decomposition showing where the response is within the system. These tools show what tier is taking the most time during the transaction, and they provide diagnostics on how the time is being consumed with the tier. This allows the development and support team to focus on the root cause.

Flash or sudden increase in business volume

A flash or sudden workload increase can originate from many different sources and can exhibit different characteristics. For example, these events can vary by duration and intensity, in terms of the steady-state workload during the event. The following scenarios illustrate how the solution depends on the characteristics of the problem:

Scenario 1 - Mergers and Acquisitions: The acquired business is being merged into the system of the acquiring business. The surviving systems can suddenly be processing a significant amount of additional information, far beyond the original design intent and coding practices.

As a result, this new volume will be the new normal state of the business. The system will not go back to the original business volume. This new combined business volume can significantly alter the peak volume as well.

Mergers and acquisitions are planned well in advance, and the systems and applications are often modified to support the expected volume increase that the new business will bring. In this case, you must consider how accurate the predictive models are for the new business volume and the new peak.

Scenario 2 - Marketing Event: Customers are encouraged to visit your web site for a special purchase or limited time event. This indeed results in a temporary spike in activity, and that activity may be from a population of “new” users that represent a use case that is somewhat different than that of the usual transactions. As with the first scenario, however, predictive models can foresee this variation to some degree, as well as anticipate the volumes of users (based upon the volume of advertising communications).

In this case, you must consider the size of the target segment, the likelihood of response, and the nature and duration of the customer response. (How do you prepare for this? Is it possible that your system can handle the overall transaction volume under normal circumstances, but in this case it fails because the promotion stresses out a specific resource or data space?)

Scenario 3 - Predictable Recurring Event: For example, a monthly or quarterly statement, in which the system generates an email inviting customers to visit their account and view the report. This scenario differs from the second in that the population of users is much more predictable, both in number and in the specifics of their use cases. So, as with the prior two scenarios, predictive models can provide substantial insights. In this case, you must consider rescheduling non time-sensitive workloads outside the peak window to compensate for the burst in demand.

In any of these scenarios, the insights gleaned from predictive models can be leveraged into tangible value when they are used to formulate performance test scenarios (including transactional distributions and aggregate user profiles).

Performance increase due to a new channel for work

In some instances an existing system will be retrofitted to accommodate a new channel for business transactions. For instance, a new business partner will now funnel work to your systems. This could be via real-time web services or batches of transactions transmitted hourly.

Planning for this should involve a predictive model for the number of transactions expected for average and peak times. You need to assess how the new load and workflow pattern will impact your existing systems.

You must not disappoint the new business partner when the channel is enabled. Usually these are high profile and visible to management for both companies. If the performance does not meet expectations, it will be extremely disruptive to the support and development teams during the triage duration.

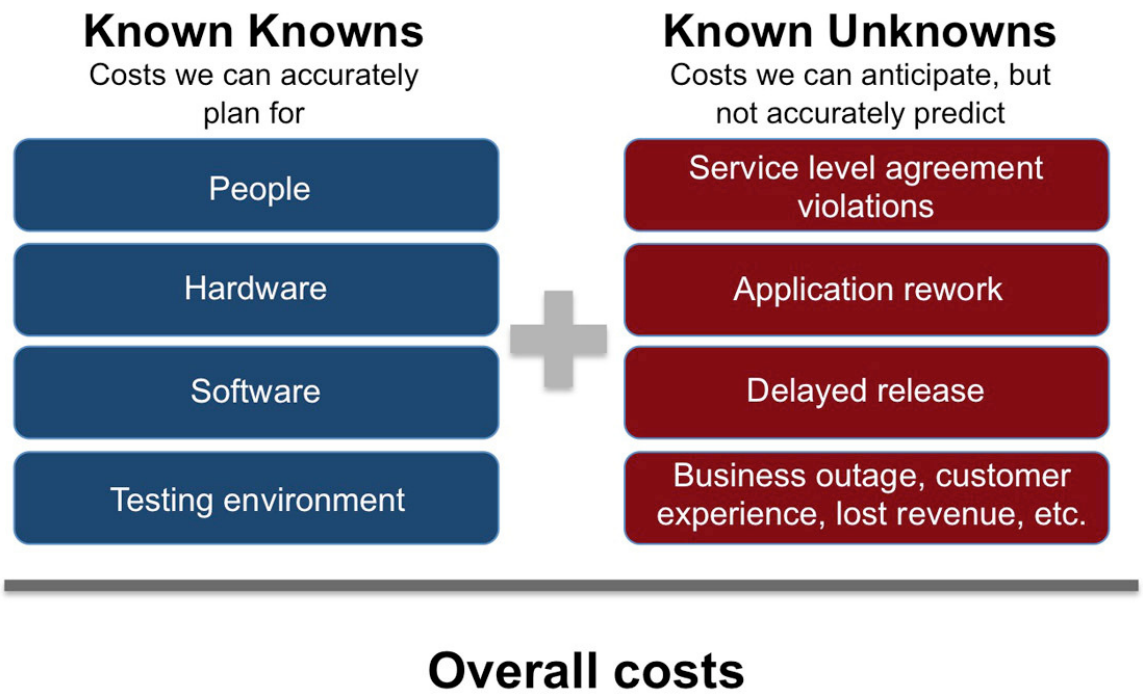
Impact

Each of these categories introduces a series of unplanned and unbudgeted tasks for the business. These issues must be solved in production, and fixes must be introduced into the production environment. This is called *firefighting*—an activity that is typically far more disruptive and costly than its more proactive counterpart.

A production rescue or firefight moves through three phases:

- **Initial crisis** - The impact is known but the cause is unknown. A swat team is formed to solve the problem.
- **Investigation into the situation** - The team must gather information from the system and determine a set of root causes. Possible solutions are produced and vetted in test environments while still processing new information from production.
- **Tactical solutions are developed** - Both short-term to alleviate the issues and longer term to prevent recurrence and avoid manual intervention.

Identifying the root cause can take days or weeks depending on the complexity involved and the tools available to the team. All the while, the swat team is under intense pressure to solve the problem, oftentimes reporting status every two hours and planning at the hourly level. Here are the categories of costs associated with a firefight:



The costs

1. **People** – Hours required to solve the problem. Many people can be involved from different disciplines within the company including the technical architect, DBA, system administrator, key business contact, software vendor, support staff, professional services, and executive sponsorship. To fully capture the cost of this, the business must require time and task tracking.
    a. *Resource* – Hours, loaded costs, distracted from other activities, and interrupted vacations.
    b. The $1,000 conference call example (i.e., so many resources attending a call that its cost is disproportionately large), and the costs of daily/hourly updates.
    c. Work environment – If firefighting becomes a core competency for the IT group, you will have a challenging work environment with some unhappy people and you should measure department turnover. As people are always responding to production or pre-production issues. Firefighting should be an exception.
2. **Hardware** – Short-term performance issues can be temporarily alleviated by adding hardware to the identified bottleneck. However, this game can be called “moving the bottleneck” and can take a recursive path. Hardware can range from additional CPU’s, memory, disk, or additional servers. This hardware is either expedited at a premium (record this cost) from your vendor or redirected from another project or environment. The hardware most certainly will be used in production; it may also be needed in testing

environments. You may have the unplanned cost of now getting all the environments in sync with the new production configuration.

3. **Testing Environments** – Trying out solutions and discovering the root cause can require the commandeering of other environments, and the interruption of their work schedule. You must record and measure the cost of taking over an environment. What does the displaced team do during the event? How is their time recorded?
4. **Software** – Additional monitoring and debugging software may be required to diagnose the problem. Licensing issues may be encountered if additional CPU's are required.
5. **Business Outage** – The performance issue can cause a business outage, reducing or deferring revenue. The business and IT teams must be fully aware of the cost of an outage. The cost of the outage can increase based on the peak usage patterns.
    a. One outage can frustrate many people; recurring outages cause behaviors to change, resulting in a loss of confidence and credibility.
    b. If you have proper measurement and monitoring tools in place, you can see the behavior change on your web site.
6. **Violation of Service Level Agreements with Business Partners** – If you have business partners that rely on your systems for key information in a near real-time context, or even nightly, you might have to pay penalties due to missing the SLA's. The business and the IT organization must be fully aware of the costs associated with these contracts.
7. **Application Rework** – Once the root cause of the bottleneck has been identified, and if it has been associated to the application software code, a development cycle will be required to design a fix, develop the code, test, and deploy it. This will have to be planned and integrated into existing releases. Does the fix require an emergency production install? Or will be part of the previously scheduled install?
8. **Delayed Application Roll Out** – The nature of the performance issue may delay a planned roll out schedule for the enterprise. What is the cost of a delayed roll out?
    a. Regulatory issues and penalties
    b. Preparation required for a retailer to roll out and train the store workforce.
    c. Publicly traded companies - business commitments to investors.
    d. ROI – Calculations may be influenced by the timeline of the new application, the original RIO my now take longer to achieve and push the benefits into a different business quarter or year. That has a significant impact to the business.

We measure and monitor systems, we track response times and database space growth, how well do we track the costs of performance issues to the business?

How did you do? Where did your performance issues come from and did you accurately capture the costs associated with the issue? Assessing your overall cost for each performance issue, where did you incur unplanned and unbudgeted costs?

Getting on top of poorly performing software can help mitigate many of the unfortunate and costly consequences an organization might experience. Developing a strong discipline in understanding performance issues and developing a keen capability to address them can create more reliable business performance, more satisfied partners and customers, and ultimately more profitable business operations.