Redpaper

IBM WebSphere Application Server
SWAT team

Increasing Resiliency for IBM WebSphere Application Server Deployments

This IBM® Redpaper™ publication explains information technology (IT) practices that can interfere with IBM WebSphere® Application Server deployments. These practices were observed by the IBM WebSphere Application Server SWAT team while assisting IBM clients in recent engagements. This paper provides examples that can give a better understanding of how and why these issues can occur and explains possible solutions. Understanding these practices can help provide increased resiliency for WebSphere Application Server deployments.

The intended audience for this paper includes IT and system administrators as well as senior architects.

Understanding the practices that can interfere with the WebSphere Application Server lifecycle

Companies increasingly depend upon websites as part of their standard business operations. Thus, when a website goes down, it can cost the business money, and getting a website back up quickly is essential. However, practices that your company's IT organization performs each day can sometimes unintentionally keep the company from meeting this simple goal.

In recent client engagements, the IBM WebSphere Application Server SWAT team observed several IT practices that can sometimes prove problematic in the lifecycle of WebSphere Application Server.

Lifecycle stages: For the purposes of this paper, the WebSphere Application Server lifecycle includes the following stages:

- Planning
- Development
- Validation
- Production and post production

Table 1 lists the practices in each stage of the WebSphere Application Server lifecycle that can cause issues with WebSphere Application Server platforms, including distributed and IBM z/OS® environments.

Table 1 Practices that can cause issues with the WebSphere Application Server

Stage	Practice
Planning	No capacity or scalability plan
	No plan for education
	No current architecture plan
Development	Not managing the entire application lifecycle
Validation	No production traffic diagram
	No load or stress testing
	No test environment that is equal to the production environment
	Changes are put directly into production
Production and post production	No migration plan
	No record of changes

In addition to these practices, common to all stages is the possibility of a communication breakdown. Communication is critical in today's business environment where changes occur rapidly. Key to success is clear communication between the correct people. Miscommunication can be frustrating and can lead to a misdirected effort. In some cases, the SWAT team observed ineffective communication between client teams, between IBM and clients, or between IBM clients and other vendors who were involved.

The following sections provide details regarding the practices of each lifecycle stage.

Planning stage

The planning stage is fundamental and determines what the final application does and how the resources are allocated. Key plans for the lifecycle of the application are made or neglected in this stage. When working with WebSphere Application Server platforms in the planning stage, your organization might run into issues if the following plans are not in place:

- No capacity or scalability plan
- No plan for education
- No current architecture plan

No capacity or scalability plan

Part of the planning stage determines the amount of data that will flow into, through, and out of the application. The type of data and the volume of that data will grow over time. If a capacity or scalability plan is not provided for, organizations often encounter the following types of issues:

- No prediction for the needed production capacity or response time
- No prediction of which transactions will occur, in what combinations, and how often
- No consideration for the path-length of each transaction
- No prediction of how many concurrent users will make up the production workload
- No periodic update to the plan for market changes, for example the holiday season

No plan for education

The planning stage also includes allowing time for the organization to learn the requirements of and how to effectively use and tune any applications. Time and resource constraints can prevent formal class attendance, or organizations might not allocate time to study educational materials, both of which results in limited knowledge on the following critical topics:

- Problem determination
- Performance tuning
- Available product features

No current architecture plan

Having an architecture plan that is current can help you to understand how a change in one component affects the other components. This plan allows your organization to upgrade applications and diagnose issues more easily. When no architecture plan is in place, an organization can experience the following types of conditions when working with WebSphere Application Server platforms:

- A diagram of the application flow between the various software products does not exist. A diagram of where these products reside in the topology does not exist. Where is IBM DB2® located? Where is WebSphere Application Server located? What is in the cluster?
- A current architectural diagram does not exist, but a diagram does exist that is based on an older version of the architecture.

Development stage

The development stage is the creation stage, which builds a foundation for the production stage. This stage is more than just the original development of the application. It also includes all of the enhancements or modifications to the application.

When working with WebSphere Application Server platforms in the development stage, keep in mind the complete lifecycle of the application and possible application errors that can occur. For example, the following common issues can cause applications errors:

- Allocation of large objects that causes the heap size to grow too large
- Redundant computation calls that increases CPU usage
- Infinite loops that cause increased CPU usage

Validation stage

The validation stage is a testing phase that verifies that the application works as you planned in a complex environment. The environment is similar to your production environment in terms of the network, hardware configuration, back-end database, and load.

When working with WebSphere Application Server platforms in the validation stage, the following common practices can cause issues:

- No production traffic profile
  - A diagram of network, routers, switches, and hubs does not exist.
  - Data on the capacity of the network or network segments does not exist.
- No load or stress testing
  - Load or stress testing is not done.
  - Load or stress testing is done, but the testing is not based on the production load.

  - The transaction pattern is not simulated accurately to the production transaction pattern.
  - The load is simulated accurately, but the back-end database size is significantly smaller.
- No test environment is equal to the production environment
  - The test environment is too small or is overcommitted and not available when it is needed.
  - The test hardware, network, or software levels differ from the production environment.
  - The z/OS logical partition (LPAR) on the same machine or network is not isolated from the production system.
  - The configuration settings for the test systems are different from settings for the production systems.
- Changes are put directly into production
  - Changes to a configuration or a fix are put into the production system directly, and then it is discovered that the changes do not work.
  - A test environment does not exist that is equivalent to the production environment. No environment exists that can be used to simulate the problem or the load. The client is forced to test the changes live in the production environment.

Production or post-production stage

The production or post-production stage is the key stage for the lifecycle of the application and is far longer than the original application development stage. However, it is possible to overlook this stage. Note, however, that at this stage, the application is now in production. Thus, fixing problems is often costly and is visible to the public.

The software platform that the application is built upon evolves over time. Thus, co-requisite packages need upgrades and integration testing. Hardware and operating systems might be upgraded, and then the application itself might expand or interface with additional applications or databases. All these changes require adequate planning, documentation updates, and plan completion.

In recent engagements, the SWAT team observed the following issues in this stage:

- No migration plan
  - Adequate time is not allocated for planning the migration or studying the ramifications.
  - Enough investigation time is not taken to determine how new Java Platform, Enterprise Edition (Java EE) specifications affect the system.
  - Developers are unaware of the co-requisite software requirements.
- No record of changes
  - A concurrent record of changes is not made. Changes are made under stress.
  - Multiple different teams make changes to the environment without records and without communicating well with each other.

Percentage of occurrences observed in SWAT team engagements

The SWAT team works with a small percentage of WebSphere Application Server clients. However, the types of issues described in this paper are a common occurrence among these clients and are often directly related to the situation in which the clients find themselves.

Sample data: Although the data was taken from WebSphere Application Server clients, these practices are not specific to WebSphere Application Server. For example, if you do not complete stress testing on WebSphere Application Server, you might not complete stress testing on other products or applications in that same environment.

Figure 1 describes the percentage of the frequency of the practices and patterns that the SWAT team observed in recent engagements.

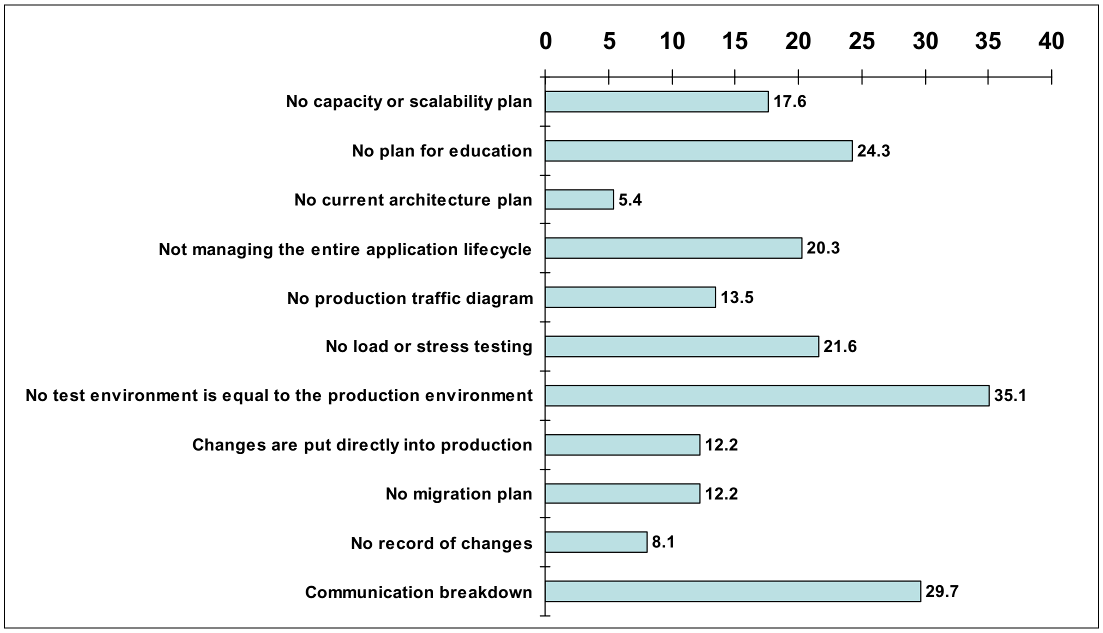

Figure 1 Percentage of frequency of practices in recent SWAT team engagements

This sample data provides the following information:

- Having no test environment that is equivalent to the production environment continues to have the highest occurrence among IBM clients who experienced a highly critical situation (35.1%).
- Problems with communication continue to be an issue, with the second highest occurrence (29.7%).
- No available education plan is the third highest occurrence issue (24.3%).

Case studies and possible solutions

The following sections provide examples and possible solutions for each of the practices listed in Table 1 on page 2.

No capacity or scalability plan (17.6%)

Although a capacity plan is not absolutely necessary, it is helpful if the business ever needs to grow. A plan is not always successful if it is as simple as buying a bigger piece of hardware. This simple plan might work for a while, but it probably is not the most cost effective method to increasing an application's capacity. A successful capacity plan also includes a high availability (HA) plan.

The need for an HA plan is not unusual. Every business wants to get the most work with the least amount of hardware for the entire business day. A business cannot accomplish these goals without an HA and capacity plan in place. If an entire application is put on a single system, it might be able to handle the capacity for a while. However, if that system goes down, the entire application will not be handling traffic, and the business will not be receiving web-based income until the system is up again.

If the application is deployed on two smaller systems that can handle the same capacity, the hardware cost is nearly the same price. However, the chance of losing the entire application is now much lower. This process is a good beginning to a pair of capacity and HA plans. Also, virtualization is an increasing popular and effective way of handling some aspects of HA.

Keep in mind: Take care regarding placement of virtual machines (VMs) on hardware so that you do not create a single point of failure (SPOF).

Case study

This case study shows the importance of a well-researched capacity plan.

In this case study, the client's workload is not properly distributed, which results in a capacity issue. The organization planned for enough hardware and incorporated a good HA plan into the capacity plan. However, the application cannot handle the additional workload as demand increases.

Initially, CPU, memory, and network utilizations are all thought to be the cause of the capacity issue. However, as a code review of the application is completed, an issue is discovered in the application itself. The client is doing cell-scoped Java Naming and Directory Interface (JNDI) calls where each client looks up the enterprise beans through the deployment manager. Although the application is spread across six nodes, the deployment manager is the bottleneck in this case.

The point here is that a capacity plan is more than a hardware plan. Capacity planning needs to consider the application, the application deployment, *and* the application client use.

In this case, restructuring the naming calls to go to cluster members resolves the issue. The application can then handle more traffic. In addition, the WebSphere Application Server Development Team learned from this issue and now caches naming more strategically in response to the client calls.

Possible solution and preventive care

When you put a capacity plan together, remember that the plan includes what both the hardware and the application can handle. The biggest mistake in capacity plans is overlooking the needs of the application. For example, how many connections per transaction are needed to access the database? With a well-tuned application design, you can save on hardware costs. So, do not overlook the resources that the application uses.

Use the following checklist to help prevent this type of capacity issue:

- What capacity plan can the back-end database handle for an additional load?
- What can the network routers handle for increased network traffic?
- What can the Lightweight Directory Access Protocol (LDAP) server handle for user authentication?
- If the application is spread across a wide area network (WAN), are remote calls kept to a minimum?
- Is part of the hardware also being used for disaster recovery or HA?
- What is the maximum amount of hardware utilization before you need to provide additional hardware? Take periodic measurements and plot a trend line.
- Are object types being properly used in light of the increased capacity?
  - Entity enterprise beans use a caching policy for the bulk of client database reads.
  - Stateful enterprise beans are passivated to a fast storage device.

No plan for education (24.3%)

WebSphere Application Server is built to comply with an open, but evolving, standard. Each new release of WebSphere Application Server conforms to a succeeding standard of Java EE for application servers. As the standards change, so do the succeeding releases of WebSphere Application Server.

Generally, your organization should take the time to learn about WebSphere Application Server when deploying a release, regardless of whether it is a first release or a subsequent release. Especially for updates, ask the following questions:

- What features are new?
- What features are now deprecated?
- What features are simply not there anymore?

Typically, organizations do not invest in education because they do not want employees to spend time away from work at a class. However, lack of education can lead to the following types of production crises:

- Migration problems

  Each WebSphere Application Server release evolves from the preceding release. Organizations who do not review an application for changes that are needed to run efficiently under a new release might see the following types of issues:

  - Performance problems

    Performance that was efficient in the *n* release might be suboptimal in the *n*+1 or *n*+2 release.

  - Functional problems

    A function in the *n* release might be depreciated in the *n*+1 release and removed in the *n*+2 release.

Alternatively, a new function might not be used. This opportunity might be missed because each enhancement reflects the increasing capability that is requested and reflected in the new Java EE standards.

It is also possible that the original application was not Java EE compliant but that the release to which you are moving handles only Java EE-compliant applications.

- Integration problems

  Each release has established explicit levels of prerequisite or co-requisite software. You might encounter issues that can affect the performance of an application if you do not check and migrate to the prerequisite and co-requisite software.

Case study

A toy company started their online site using WebSphere Application Server V3.5. The company moved successfully to V5.0 using the V3.5 web server plug-in without making any application changes. Now, however, V5.0 is out of support. In addition, the toy company's online business has grown so that they need the added performance of V6.1.

The company buys WebSphere Application Server V6.1, installs it, and then deploys their application. However, the application does not deploy and nothing works. The company contacts IBM support and files several problem management records (PMRs). They believe that WebSphere Application Server is the cause of the problems because they did not change their application. They are frustrated. They were aware that WebSphere Application Server V6.1 conformed to the Java EE standard. However, because their application ran on V4.0, which also was Java EE compliant, they were certain that the problem was with WebSphere Application Server.

Before their migration to V6.1, the company did not learn about the enhancements and changes in V6.1 that necessitated changes to their application before moving from the V4 plug-in to V6.1. In addition, they had no plan in place to learn about these enhancements and changes in the future.

Possible solution and preventive care

Allowing time to periodically review your application can decrease possible frustrations and issues when moving to a new version of WebSphere Application Server. Consider both the changes in your business needs as well as sections that are affected by Java EE capability changes.

The issue of proper education implies that you need the time to learn about new or depreciated functions in new WebSphere Application Server releases. However, education no longer implies that you must go to the classroom or decipher a manual by yourself. IBM offers the following educational opportunities at no initial cost:

- IBM Education Assistant

  The IBM Education Assistant enables you to learn task-specific procedures from your desktop without going to a classroom. Many educational plug-ins are available for IBM Rational®, WebSphere, and IBM Lotus® products, including WebSphere Application Server and WebSphere Portal Server.

  For more information, see the following website:

  `http://www.ibm.com/software/info/education/assistant`

- Technical Exchange

  The WebSphere Support Technical Exchange is another learning opportunity available at no initial cost. These learning opportunities involve live presentations that are open to all clients on topics of interest. You can suggest topics for these sessions. All sessions include a question and answer time and presentations are done several times per month. Talks are recorded and archived and are available for viewing after the live presentation.

  For more information, see the following website:

  http://www-947.ibm.com/support/entry/portal/scheduled_tech_exchanges/software/websphere/websphere_brand_support_(general)

- Technotes

  IBM Product Support pages contain information that is provided by the IBM Support and Information Development teams to assist you in maintaining the WebSphere Application Server environment. You can search on any topic to find a related technote, but IBM support suggests that you periodically check the featured documents. These featured documents are technotes that apply to many client situations, from performance to migration, and so on.

  For more information, see the following websites:

  – IBM Support

    http://www.ibm.com/software/support

  – WebSphere Application Server for distributed platforms

    http://www.ibm.com/support/docview.wss?rs=180&uid=swg27006233

  – WebSphere Application Server for z/OS platform

    http://www.ibm.com/support/docview.wss?rs=404&uid=swg27006898

- IBM Redbooks® publications for WebSphere

  There are many pointers to IBM Redbooks publications, a popular source of technical product information. This particular page helps focus your search specifically on Redbooks publications for WebSphere products. For more information, see the following website:

  http://publib-b.boulder.ibm.com/Redbooks.nsf/portals/WebSphere

- Education videos on YouTube

  The WebSphere Education channel on YouTube is an excellent source for short course previews that cover WebSphere products and service-oriented architecture (SOA) design principles. Currently, 50 previews are available at no cost, with more planned for the future.

  For more information, see the following website:

  http://www.youtube.com/websphereeducation

- IBM Support Assistant

  The IBM Support Assistant is fundamentally a platform, or framework, for software support engineers. An evolving list of IBM provided product support libraries and tools can be downloaded and added to the framework. Content updates are automatically and regularly performed. IBM Support Assistant gives clients the same access to these tools and libraries that IBM support engineers use. There is no charge to install or use IBM Support Assistant.

  You can find out more information and download a copy from the following website:

  http://www.ibm.com/software/support/isa

No current architecture plan (5.4%)

What does your application do and how does it do it? Where does it get the data and where does it put the answers, and all the other things in between? How does the next person know how to safely add a feature when a new function is added? How do you rearrange the environment when different systems, either software or hardware, are added, removed, or replaced in the environment? The key to doing any of these tasks is having a current, up-to-date architecture plan. Having an expert who knows all this information is convenient, but not realistic or sufficient. Having a rough sketch of the original architecture plan is handy, but might not be helpful six months later.

It is certain that changes will occur. The best you can do is to plan to make that change easier. In the case of the architecture, having an up-to-date architecture plan can help you to avoid production outages and, in the case of an outage, save weeks or months of problem determination.

Case study

A large retail company uses WebSphere Application Server for its online business. When the company started, they hired an IBM Software Services for WebSphere consultant to assist with developing the architecture to ensure that they had the needed capability. The architecture was defined, documented, and implemented.

The company purchased other businesses and planned to add these businesses to their online shopping application. The company also added other services to their online website, such as signing up for the store credit card and making personal services appointments. Each division of the company brought these requests to their central IT department, where a person was assigned to add the function. Some of these additions were documented in a memo, some in notebooks, and one was documented in an addendum to the original architecture plan.

Three years and many changes later, there is a production outage, and no one can determine what is wrong. The company opens a severity one problem management record (PMR) for WebSphere Application Server claiming that "Nothing was changed." According to the company, WebSphere Application Server "just hung."

An IBM engineer visits the company onsite and asks for the architecture plan to better understand the data flow. No one in the company can locate the architecture plan. The last update to the plan was two years earlier, but it included only three updates after the original design was complete. This situation reveals that most of the additions over the recent two-year period were never added to the original architecture plan. No one in the company really knows how the application flows or how it should flow.

The situation results in two weeks of intense work by both the company and IBM, with many subsequent application hang instances. The issue results from a new credit card feature that was added several months earlier. The feature causes a database lockup with an older credit card billing feature when the feature is accessed during a quarterly bill calculation. The error did not surface until the quarterly billing cycle began when additional customers from the new business acquisitions were added.

What is still not known is whether there are other potential deadlocking flows still in the application or whether new additions will cause new problems. Without a current architecture plan, time-consuming debugging must be done to determine the application flow and deadlocks. Without processes in place to keep the architecture document up-to-date, the document becomes less useful as changes are made and personnel turnover occurs.

Possible solution and preventive care

The key to solving these types of issues is self-imposed discipline. Place a priority on keeping architecture documentation up-to-date. Maintain an architecture document that can minimally answer the following questions:

- What is the data flow?
- What is the environment including both hardware and software?
- What are the internal structures?

Additionally, you can use one of the architecture patterns that are recommended for the level of WebSphere Application Server that you are using. These architecture patterns are defined in various IBM Redbooks publications. The pattern recommendations can vary by WebSphere Application Server version level and do not absolve you of maintaining documentation for your own installation, but they do provide a solid starting point.

Architecture patterns: For more information about WebSphere Application Server architecture patterns, see IBM Redbooks publications:

`http://www.redbooks.ibm.com`

Not managing the entire application lifecycle (20.3%)

On IBM client sites, the SWAT team observes issues because organizations do not effectively manage the entire application lifecycle. Initially, these types of issues are often thought to be caused by WebSphere Application Server. Many times, however, the application error is caused by issues discussed in the previous section (that is, a current architecture plan does not exist).

Arguably, the most important piece of the architecture is the application. Without the application, there would not be a business. For that reason alone, tightly manage the application lifecycle, even after the application is deployed into production. Application errors often are the result of not managing the entire application lifecycle.

Case study

A large US company ventures into a new area of business and has major issues with their application. The problems are so severe that the production date of the application is delayed, which is a serious issue for the company. The company eventually makes a decision to put the application into production and to endure the daily outages until they can address problems. The issue is that WebSphere Application Server just restarts by itself, for no apparent reason.

The company's application was developed by a third-party company and also includes pieces from yet other companies in the application. The application went through a good design review, and IBM also participated in these reviews. The application was thoroughly unit tested and held to a good application lifecycle. It was later used in preproduction where estimated live loads were simulated. However, in preproduction, WebSphere Application Server just restarted itself.

IBM and the company spent a lot of time collecting trace information from WebSphere Application Server and the Java virtual machine (JVM). Upon investigation, it is discovered that the third-party software application that contains pieces of additional code is causing WebSphere Application Server to restart.

The additional code that the third-party company thought would not cause issues is in fact causing the JVM to hit page faults and throw a kill signal, which ultimately causes WebSphere

Application Server to restart. After removing the additional code from the application, the problem goes away.

Possible solutions and preventive care

Ask yourself the following questions regularly about the application lifecycle:

- Does the application do what it is intended to do?

  Every application is written to solve a problem. The problem might range from offering a product for sale to helping to solve a medical mystery. Over time, applications become more complex, and the problem that the application was intended to solve is forgotten. You add a function to an application here and a function there. The next thing you know, the application is selling books on medical mysteries in addition to solving the medical mystery.

  Keep the application distinct, and use the application for what it was originally intended to do. If the application needs to do multiple things, such as sell books and do research, separate those functions into multiple applications. Then, when you have issues with the application, you can more easily isolate problems and possibly reduce the impact to the business. If the entire business process is written into a single application and that single application has problems, the entire business has problems.

- Is the application designed using a modeling language?

  The quality of an application starts with the quality of the idea for which the application is intended. The design of that idea is the next step in producing a high-quality application. If the application design is scratched out on napkins versus in a modeling tool using Unified Modeling Language (UML), the application reflects it. There are times when applications have problems that are on a much higher level than a NullPointerException from something that was not properly checked.

  Problems can and do begin in the application design. When problems begin at the application design level, the problems become difficult to discover from a serviceability point of view and even more difficult to fix.

  Begin with a solid design. Review the design, re-review it, and keep it up to date. If the design needs to change because of a technical oversight, update the design, and determine how that change impacts the remainder of the application. After the application is deployed and it is responding to live traffic, is not a good time to discover the impacts.

- Is the application code reviewed by someone other than the person who wrote it?

  Like the application design, the quality of the design implementation reflects how well the application performs the task that it is designed to accomplish. The design implementation phase is also a phase when problems occur. Every person knows that if you put a design in front of 100 programmers, the chances are extremely good that those 100 programmers will implement the design in 100 different ways. The different implementations are not a result of a poorly written design but occur because each programmer has a unique interpretation and implementation.

  During the code review, also look at how the translation of the design to code is implemented. Where are the best data structures used most effectively? Were the objects themselves managed properly to prevent memory leaks?

  The best way to make sure that the code matches the design is to have a second or third person, or a team, review, and compare the design to the code. The second person or team also needs to make sure that the code is technically correct. This process can save time later when the application problem is not in the design.

- Is the application serviceable?

  During the code review, was the code reviewed from the serviceability point of view? Code serviceability is extremely important. Both WebSphere Application Server, and the applications that run on it, must have good serviceability. There are times when the serviceability quality of WebSphere Application Server is used to help find, troubleshoot, and resolve problems that are purely application problems. In these instances, the application developer did not put good serviceability code into the application.

  To resolve application errors, the last measure of defense is application logging. Ask the following questions:

  – Is it more cost effective to save a few hours of a programmer's time?
  – Is it more important to get an e-business back in business at the cost of lost revenues that the e-business is not generating while it is unavailable?

No production traffic profile (13.5%)

This common practice usually involves a missing network diagram for the routers, networks, switches, and hubs. It also involves the lack of data on the capacity of the networks or network segments.

When you need to get somewhere you have never been before, you can search for directions, ask someone who knows how to get there, or use a map. Highway traffic is similar to a computer network. If the network is too clogged to handle the level of traffic, it grinds to a halt. We know this is going to happen, and we know how to prevent it. However, our budgets and deadlines do not always allow for proper, timely intervention.

How do highway planners know what size road to build? How do they know where a new stoplight will save lives? How do shops and stores know where to locate for visibility? Municipalities create complex traffic studies and models of future growth prior to investing in new highway infrastructure. This planning enables municipalities to plan for future growth and accommodate their citizens for decades to come. The same type of study and attention to detail is needed when building a network for an enterprise-level web application.

How does your data know how to get from one area to another? A routing table is analogous to a road map. The routing table makes for an accurate, but not detailed, roadmap. You need to take this process a step further to get control of your data. You must have a detailed view of your network topology.

Case study

An insurance company appears to have a hang condition in their WebSphere Application Server configuration. IBM problem determination analysts are engaged and an apparent hang situation is not reported in the software. The operating system does not report any problems. Problem determination rules out all of the possible problems except in the network.

The system analyst that is assigned to help troubleshoot the issues suspects that there might be problems in the network and asks to review the network topology charts. Unfortunately, current diagrams of the network do not exist. The analyst interviews the teams that, together, are responsible for maintaining the network.

They determine that a new hub was recently added to the network. This hub is used to share the traffic with a different area and is making the rest of the network slow down to the degree that WebSphere Application Server seems to hang.

Possible solutions and preventive care

Create a network diagram and keep it current. If the network team is in a different group organizationally, involve them in your planning phase. Add hardware to your network carefully and communicate your plans with the entire team because even the most innocuous changes can have far-reaching impacts that you might not anticipate.

Keep the following information available:

- Know the logical location of the routers and firewalls.
- Know where the servers are located on each network segment.
- Know how much traffic each server is expected to handle.
- Ensure that there is sufficient bandwidth to handle the traffic.

Matching network design to the expected traffic is both an art and a science. Combining experience and data will tell you how well your network will hold up under pressure.

No load or stress testing (21.6%)

After functional tests, users are expected to conduct thorough and vigorous load and stress tests on the application before moving it to the production system. Ideally, the load and stress tests involve every part of the application and simulate the real-world workload and user patterns. These load or stress tests help determine the performance impact by putting the system under extreme situations to uncover any remaining program errors that have not been found during the function test.

A truly robust, production-like series of tests executed repeatedly can be useful in the identification of “slow” leaks in Java application memory. Memory usage measurements can provide both performance information and can identify memory leaks from the application. These slow memory leaks need the volume and repetition of stress and performance tests to help make their presence known before the application goes into a production environment.

Despite its importance, some organizations choose to bypass load or stress tests and move directly from a development to a production system. The consequences can be dire, as shown by the following real-world lessons.

Case study 1

A European company is experiencing severe performance issues with WebSphere Application Server V6.0 for the Linux on IBM System z® platform. The utilization rate causes a system slow down, which leads to long response times for users.

Upon close examination, the SWAT team finds the following issues:

- The z/OS virtual machine is not tuned properly. It is set to 2 GB of extended storage when the recommended amount for their setup is 4 - 6 GB.
- The application was designed poorly and has a potential to deadlock.
- The application has many large objects.

To do some tuning to find a proper setting and to spot potential program errors, the SWAT team conducts load tests with one to 500 simulated users. Through the load testing, IBM and the company determine the optimum settings and discover the deadlock within the application code. After the modifications are put into the production system, the application runs smoothly without a performance issue.

Case study 2

A large media company finds that its web-based catalog application experiences frequent slow downs and the web browser sometimes returns blank pages. The SWAT team examines the log file and finds excessive activities for the garbage collector.

The SWAT team conducts stress tests in a quality assurance environment and notices a correlation between the garbage collector activity and downloading large files that are greater than 1 GB. The root cause is a servlet that attempts to cache large files in memory. The problem resolves after servlet caching is disabled.

Possible solutions and preventive care

The previous two case studies show how important it is to conduct thorough and rigorous load and stress tests. In both cases, if the organizations had conducted load and stress tests in a quality assurance environment, they might have spotted issues before moving into a production system.

Keep in mind that unit and system tests are not substitutes for load and stress tests. Typically, load tests are conducted to verify that the new system satisfies its design specification. Stress tests verify the stability of system under load and in extreme situations.

For existing applications, you must have a good understanding of peak loads, duration, and time of occurrence. You also must analyze the pattern of usage, for example how many concurrent users are actively engaging in transactions, how many users are browsing the catalog, and so on. These patterns might have a significant effect on system workload.

You can use this knowledge to prepare the testing script. Prepare a test script that can drive up the workload to a level that is similar to the peak load with a comparable usage pattern. Test tools are available that can help you do automated load testing. Ideally, perform the load and stress tests on the same production system before it goes live in a production environment. If it is not feasible, find a quality assurance system that is as close as possible to the production system, and use it.

When you do not know how a new application might be used, you must rely on the design document and common sense when you prepare the test script. It is prudent to be conservative at the beginning and prepare for the worst-case scenario.

Here are useful links for Java memory leaks:

- Handling memory leaks in Java programs

  `http://www.ibm.com/developerworks/java/library/j-leaks`

  This article provides information about causes of Java memory leaks and when these leaks should be of concern. It also provides a quick hands-on lesson for tackling leaks in projects.

- Java performance hints

  `http://www.ibm.com/servers/eserver/zseries/software/java/javafaq.html#perform`

  This resource is the section for Java on z/OS with several other resource references.

- VerboseGC diagnostics—Garbage Collection and Memory Visualizer

  `http://www.ibm.com/software/support/isa`

  The Diagnostic Tools for Java (DTFJ) Garbage Collection and Memory Visualizer and the IBM Pattern Modeling and Analysis Tool (PMAT) are both part of the IBM Support Assistant and can be used to visualize and tune garbage collection for Java machines.

- IBM HeapAnalyzer

  http://www.alphaworks.ibm.com/tech/heapanalyzer

  This website provides statistics about your heap and allows you to look at objects of interest. It takes a heapdump as input.
- IBM Memory Analyzer Tool

  http://www.ibm.com/developerworks/java/jdk/tools/memoryanalyzer

  The IBM Memory Analyzer Tool can analyze memory dumps (heap dumps) from the Java virtual machine (JVM) that is running the WebSphere Application Server. It is part of the ISA Workbench tools.

No test environment equal to production environment (35.1%)

This practice is by far the most common practice observed at client sites by the SWAT team in recent client engagements. In tracking the trends, the occurrence of this practice has almost doubled in the past three years. Having a separate test system that is identical to the production system is critical. You can conduct load and stress tests on the separate test system before moving the application over to production as described in the previous section.

The following advantages exist for a separate test system:

- Separates the test system from production system and prevents unintended production disruption
- Provides a platform for you to perform functionality and integrity tests before performing major upgrades
- Provides an environment for duplicating production problems and testing fixes

Most WebSphere Application Server users have certain kinds of test environments. However, these environments are not necessarily identical to the production system because of the cost factor. There are cases where the lack of a separate test environment has caused a product outage.

Case study

A major bank in South America discovers that their WebSphere Application Server for z/OS environment experiences frequent product outages during tax season, which disrupts online tax business for their customers and damages the relationship with their customers. The SWAT team investigates and finds many factors that contribute to this situation but primarily finds configuration errors.

One particular mistake is that the company created two application servers on a single installation of the base version for WebSphere Application Server. One of the application servers is the test server and the other application server is the production server. Because both application servers run on the base WebSphere Application Server version, their logs are shared, their ports are carefully configured to avoid conflict, and, most importantly, because they run on the same binary codebase, any upgrade to the software development kit (SDK) disrupts both of the application servers. While frequent updating to the test system is necessary, the repeated disruption to production system is intolerable.

Recognizing this mistake, the SWAT team urges the company to separate the WebSphere Application Server test and production systems on different LPARs so that they do not affect each other. The subsequent problem determination is smooth after the separation is completed.

It might be convenient to have the test and production system together, or it might look wasteful to assign a dedicated test system. You must understand that this convenience comes at a price and a production system outage is costly.

Possible solution and preventive care

There are a number of approaches for creating a test environment. Each approach varies in cost, planning, and risk. Having the ability to test the application environment, not just the application, is a necessity.

Use the following checklist for the test environment:

- Understand that a test system is a must, not a luxury. Do not use a production system for test purposes.
- Physically separate a test system from a production system and ideally have two different groups of operators manage these systems.
- Clearly label the test and production systems with a different set of security identities.
- Install identical software on the test system and the production system. Use similar hardware for the test and production systems.
- Upgrade WebSphere Application Server and the applications themselves on the test system first before upgrading the production system.

The following list describes the different types of test environments that are based on the availability of resources:

- Maintain an exact duplicate of the production environment.
- Maintain a scaled-down environment with load generators that duplicate the expected load.
- Maintain a model or simulation of the environment.

Important: Implement and test all of the changes in a test environment before you add them to the production environment.

Changes put directly into production (12.2%)

Although most companies understand the difference between test and production systems, many find it convenient to put the change into the production system directly, which bypasses the test system altogether. This practice is especially true if the change is only a small change. The common argument for bypassing the test system is usually that the update to a production system is so urgent that the system administrator must make an exception. Problems can also be compounded when adequate change control procedures are not followed.

It is true that sometimes the production system needs urgent patching. However, always use formal channels for changes. For example, apply changes to test systems, conduct thorough tests, and then move the changes to the production system. Applying changes directly into the production system defeats the purpose of a test system and can leave the production system with undocumented changes. Additionally, direct updates to the production servers might create different versions and patched levels of WebSphere Application Server on one system and make future maintenance an issue.

Case study 1

A large media company experiences frequent website outages. The SWAT team determines that the outages are a result of a memory issue. Application code causes heap fragmentation. IBM suggests different garbage collector settings but receives inconsistent results from different systems. Close examination reveals that the four WebSphere Application Server installations are not identical.

Each installation has a different SDK level, different fix pack, and even a different interim fix installed. It is surprising that project worked at all. After upgrading each installation to the latest SDK version with the proper garbage collector settings and the same level of application code, the heap fragmentation problems are resolved.

Case study 2

A large bank diagnoses a product outage issue. When the SWAT team checks the design document, the team determines that the design document has not been updated in two years. Numerous modifications have been made to the production system without any record.

Possible solution and preventive care

Use the following checklist to prevent changes from being put directly into a production system:

- Establish a rigorous procedure for making any changes to the production system. Record all of the changes.
- Establish a production system administrative group that is independent from the development team and is responsible for approving changes placed into production.
- Assign a specific maintenance cycle for patching and updating the production system. Do not make changes to the production system during other times.
- For urgent situations, particularly security-related situations, make changes to the production system directly but ensure that the changes are well-documented with appropriate signoff and are considered as exceptions. Also, ensure that the changes are applied to the test systems as soon as possible to keep all the systems identical.

No migration plan (12.2%)

Product migration can be a real challenge for even experienced developers. Java EE migration is relatively easier in comparison to other products because the Java EE standard has been trying to make the Java EE server more platform- and vendor-independent.

The following types of major migrations can occur:

- Upgrading WebSphere Application Server from one major version to another version, for instance, from V6.1 to V7.0.
- Migrating from another Java EE server or another operating system.

To ensure a smooth transition, make a detailed list of differences and modify the code, if necessary. Otherwise, the migration might adversely affect the performance and stability of the production system.

Case study

A major bank in the Asia Pacific region migrates from WebSphere Application Server V4.0 to V6.1. After the migration, the bank finds that performance degrades significantly. They upgrade to a newer WebSphere MQ Workflow version and find that the system crashes randomly.

Close examination reveals the following issues:

- WebSphere Application Server V6.1 is Java EE V1.3 specification compliant, which requires the Java EE container to implement a “shareable” connection, because the default setting in Java EE V1.2 is not shareable. Developers did not change either the code or settings. This situation causes the application that was written for Java EE Version 1.2 to hold the connection longer than necessary, particularly for deeply recursive, and long forwarding method calls.
- For WebSphere MQ Workflow, the newer version uses the Java Native application programming interface (API) instead of the Java Native Interface (JNI). Java Native API adds certain synchronization blocks to prevent concurrent access to WebSphere MQ Workflow session objects. The developer failed to change the application code, which caused the deadlock and system crash.

Clearly, developers need to analyze carefully the differences between versions, analyze the Java EE specifications and other settings, and make the necessary modifications to either the application code or the default settings. The Java EE specification tries to make migration easier, but it does not catch everything.

Possible solutions and preventive care

During migration, developers need to refer to the existing documents on the current system, compare them with the newer WebSphere Application Server version, and document all of the differences and how to reconcile these differences.

Make note of the following items to prevent some migration hazards:

- Changes in the Java EE specifications
- Differences in the default settings between versions
- Differences in the vendor extensions
- Changes to the operating system-specific configurations

For more information about migration planning for WebSphere Application Server, see the following website:

`http://www.ibm.com/support/docview.wss?rs=180&uid=swg27008724`

No record of changes (8.1%)

The more people who have access to the production and test systems, the more important it is that you have a centralized change record. In this record, everyone who has access to the system can track the changes so that the record can help you to narrow down the problem areas.

Case study

A pharmaceutical company is testing a new application in their test environment. Everything proceeds well, and they decide to install the new application into their production environment. The application server does not start and issues a ClassCastException exception. The test server and the production server configurations are identical. WebSphere Application Server, the application, and the parameters are the same.

This issue becomes a puzzle. IT needs to launch this application because the remainder of the company has been informed that testing was going well. They try to remember what changed, but a centralized change record does not exist. Several developers and system administration staff members have access to both the test and production environments. Unfortunately, some of those people are on vacation.

One of the developers remembers that a shared library is a slightly different version between test and production. After they change that library, everything works fine.

Possible solution and preventive care

Use the following checklist to prevent this issue:

- Use a centralized change record system and maintain it accurately.
- Create a strict process to inform everyone of the changes.
- Record who changed what, and why.
- Be aware of the fact that taking a little time to follow this process can prevent huge problems later.

Communication breakdown (29.7%)

Communication might break down in many different ways. Communication is always 2-way and requires reinforcement by delivering the information in multiple forms such as speech and in writing. In project management, you communicate to get things done, pass on and obtain information, reach decisions, achieve joint understanding, and develop relationships. Using these techniques correctly determines the success of a deployment.

Recognize the following communication barriers:

- Installing products without considering product updates
- Understanding the environment
- Communicating with management
- Considering cultural issues
- Timing the deployment
- Coordinating the impending deployment with the affected products and services owners
- Having a back out plan

Who needs to be communicating what to whom, how it is communicated, and when that should be done is key to having a smooth and productive lifecycle for a product, application, or even a business process.

Case study

A large grocery chain runs their inventory nightly on an IBM System z platform. The company installs a new web application that generates orders to their suppliers based on the in-store inventory summary for each store.

This new application is written in Java and has been tested on distributed machines, but the application runs in production on the System z mainframe. The application prints a list of warning messages if unusual patterns are noticed in the order or if the application has errors. The printed error messages are in the application because this application is new.

The application is delivered and deployed on the System z mainframe as requested by the development organization. The operations group, which runs the System z mainframe, generally starts programs, stops programs, and performs other system programmer duties. They know what printed messages to expect from the z/OS system. They do not know anything about what the application does because the application writes only to disk.

The application developers know that the people in the operations group are called *system programmers*. Therefore, they make the assumption that these people write programs, not that they do system administration. Thus, they do not explain how the application runs, and they do not explain what to do when the application sees errors or when unusual patterns exist.

The new application does not seem to be doing well at ordering new supplies for the stores. However, because the developers have not received printed error messages, they do not know that there is a problem. When management confronts them, they say that they do not know anything about it because they have not received any printed error message. Meanwhile, the system programmers in the operations group are throwing away the extra paper that is now printing out because they do not know what the printed error messages are for or why they are printing.

Someone started following the lifecycle of the application development and deployment and discovered the communication breakdown. Because both groups are now frustrated with the process, the two groups must work to improve the communication issues.

Possible solution and preventive care

Use the following checklist to help prevent communication breakdown:

- Identify who needs to communicate on what subject and to what extent.
- Create a checklist to ensure that the current information is synchronized.
- Have a periodic meeting to make sure that everyone understands the process.
- When someone is out, the backup person must know exactly what is occurring.

Impact of production outages to your business

Having a website go down can cause frustrations and challenges for e-business vendors. In addition to the loss of revenue, the outage hurts the credibility of the vendors and opens doors for competitors. Many e-business vendors rank stability as the most important factor when they choose products. However, most vendors do not have a clear understanding of the actual cost of product outage.

The following tables estimate the monetary cost that is incurred during product outage or when a website is down. Other non-monetary costs, such as the loss of customer loyalty and goods, are difficult to quantify and, thus, are not discussed here.

Table 2 shows the cost of an outage for an online-only retailer.

Table 2 Cost of outage for an online-only retailer

Transactions/hour	Average transaction value ($)	Down time (hours)	Cost of the outage($)
T	V	H	C=T*V*H
100 (Normal season)	$56[a]	1[b]	$5,600
300 (Peak season)	$56	1	$16,800

a. The average transaction value for an American retailer is $56.

b. The typical recovery time from an outage is 45 minutes during normal business hours. However, the time can be considerably longer if the outage occurs during non-business hours.

Table 3 shows the cost of an outage for a retailer with both online and brick and mortar shops. This table illustrates the cost of a product outage for a typical small-to-medium e-business customer. You can plug in your own numbers to calculate the actual cost of a product outage. These numbers can be used to find the balance between the cost and benefit, what kind of service level to provide (24/7, 8 a.m. to 5 p.m.), and how much money to invest to improve product availability.

Table 3 The costs of an outage for a retailer with both online and brick and mortar shops

Transactions/hour	**Average transaction value ($)**	**Percentage of online business**	**Down time (hours)**	**Cost of the outrage ($)**
T	V	P	H	C=T*V*P*H
500 (Normal season)	$56	50%	1	$14,000
1500 (Peak season)	$56	50%	1	$42,000

Keep in mind that these costs are the direct losses from a product outage. For example, the losses can come from a loss of revenue during which time the product server is unavailable. There are many other indirect costs, such the loss of good faith, loss of customer loyalty, and an attack from competitors, to mention a few.

For a large online business, the indirect loss can be substantially larger than the direct loss. The negative publicity that is generated by the outage is particularly damaging. However, those losses are not described here because producing a precise monetary estimation of those losses can be a daunting task and are beyond the scope of this document.

Authors

This paper was produced by a team of specialists from around the world working with the IBM International Technical Support Organization (ITSO).

The IBM WebSphere Application Server SWAT team provides short-term, supplemental, product support for WebSphere products to IBM clients onsite or remotely. The team consists of WebSphere specialists, subject matter experts (SMEs), and certified project managers who are uniquely skilled in problem resolution and determination techniques. The team’s close ties with development places them in a strategic position to provide product usage feedback and requirements to the development process.

After each client engagement, the team documents all actions that are taken and observed onsite. In addition, the team holds debriefings with senior architects. Often, the team revisits the IBM database to determine how and why organizations experienced issues in the first place.

Now you can become a published author, too!

Here’s an opportunity to spotlight your skills, grow your career, and become a published author—all at the same time! Join an ITSO residency project and help write a book in your area of expertise, while honing your experience using leading-edge technologies. Your efforts will help to increase product acceptance and customer satisfaction, as you expand your network of technical contacts and relationships. Residencies run from two to six weeks in length, and you can participate either in person or as a remote resident working from your home base.

Find out more about the residency program, browse the residency index, and apply online at:

ibm.com/redbooks/residencies.html

Stay connected to IBM Redbooks publications

- Find us on Facebook:

  http://www.facebook.com/IBMRedbooks
- Follow us on Twitter:

  http://twitter.com/ibmredbooks
- Look for us on LinkedIn:

  http://www.linkedin.com/groups?home=&gid=2130806
- Explore new Redbooks publications, residencies, and workshops with the IBM Redbooks weekly newsletter:

  https://www.redbooks.ibm.com/Redbooks.nsf/subscribe?OpenForm
- Stay current on recent Redbooks publications with RSS Feeds:

  http://www.redbooks.ibm.com/rss.html

Notices

This document REDP-5033-00 was created or updated on August 14, 2013.

Send us your comments in one of the following ways:

- Use the online **Contact us** review Redbooks form found at:
  ibm.com/redbooks
- Send your comments in an email to:
  redbooks@us.ibm.com
- Mail your comments to:
  IBM Corporation, International Technical Support Organization
  Dept. HYTD Mail Station P099
  2455 South Road
  Poughkeepsie, NY 12601-5400 U.S.A.

Redpaper™

Trademarks