

Cisco IOS Management for High Availability Networking: Best Practices White Paper

Document ID: 25562

Introduction

Overview of Cisco IOS Best Practices

Software Life–Cycle Management Process Overview

Planning – Building the Cisco IOS Management Framework

Strategy and Tools for Cisco IOS Planning

Software Version Track Definitions

Upgrade Cycle and Definitions

Certification Process **Design – Selection and Validation of Cisco IOS Versions**

Strategy and Tools for Cisco IOS Selection and Validation

Candidate Management

Testing and Validation **Implementation – Swift and Successful Cisco IOS Deployment**

Strategy and Tools for Cisco IOS Deployments

Pilot Process

Implementation

Operations – Managing the High Availability Cisco IOS Implementation

Strategies and Tools for Cisco IOS Operations

Software Version Control

Proactive Syslog Management

Problem Management

Configuration Standardization

Availability Management

Appendix A – Overview of Cisco IOS Releases

Release Life–Cycles Milestones

Cisco IOS Version Naming Convention

Appendix B – Cisco IOS Reliability

Cisco IOS Quality Program

Cisco IOS Release Testing

Software MTBF

Software Reliability Assumptions

Related Information

Introduction

Deploying and maintaining reliable Cisco IOS® software is a priority in today's business critical network environment that requires renewed Cisco and customer focus to achieve non–stop availability. While Cisco must focus on their commitment to software quality, network design and support groups must also focus on best practices for Cisco IOS software management. The goal is higher availability and software management efficiency. This method is a combined partnership to share, learn, and implement software management best practices.

This document provides an effective operational framework of Cisco IOS management practices for both Enterprise and Service provider customers that help promote improved software reliability, reduced network complexity, and increased network availability. This framework also helps to improve software management efficiencies by identifying areas of responsibility and overlaps in software management testing and validation between Cisco release operations and the Cisco customer base.

Overview of Cisco IOS Best Practices

The following tables provide an overview of Cisco IOS best practices. These tables can be used as a management overview of the defined best practices, a gap analysis checklist to review current Cisco IOS management practices, or as a framework for creating processes around Cisco IOS management.

The tables define the four lifecycle components of Cisco IOS management. Each table starts with a strategy and a tools summary for the identified lifecycle area. Following the strategy and tools summary are specific best practices that apply only to the defined lifecycle area.

Planning – Building the Cisco IOS Management Framework—Planning is the initial phase of Cisco IOS management needed to help an organization determine when to upgrade software, where to upgrade, and what process will be used to test and validate potential images.

Best Practice	Detail
Strategy and Tools for Cisco IOS Planning	Getting started with Cisco IOS management planning begins with an honest assessment of current practices, the development of achievable goals, and project planning.
Software Version Track Definitions	Identifies where software consistency can be maintained. A software track can be defined as a unique software version grouping, differentiated from other areas by unique geography, platforms, module, or feature requirements.
Upgrade Cycle and Definitions	Upgrade cycle definitions can be defined as basic quality steps in software and change management used to determine when a software upgrade cycle should be initiated.
Certification Process	Certification process steps should include track identification, upgrade cycle definitions, candidate management, testing/validation, and at least some pilot production use.

Design – Selection and Validation of IOS Versions—Having a well defined process for selecting and validating Cisco IOS versions helps an organization to reduce unplanned downtime due to unsuccessful upgrade attempts and unplanned software defects.

Best Practice	Detail
Strategy and Tools for Cisco IOS Selection and Validation	Define processes for selecting, testing, and validating new Cisco IOS versions. This includes a network test lab that emulates the production network
Candidate Management	Candidate management is the identification of software version requirements and potential risks for

	the particular hardware and enabled feature sets.
Testing and Validation	Testing and validation is a critical aspect of software management and high availability networking. Proper lab testing can significantly reduce production downtime, help to train network support staff, and assist in streamlining network implementation processes.

Implementation – Swift and Successful Cisco IOS Deployment—Well defined implementation processes allows an organization to quickly and successfully deploy new Cisco IOS versions.

Best Practice	Detail
Strategy and Tools for Cisco IOS Deployments	The basic strategy for Cisco IOS deployments is to perform final certification via a pilot process and rapid deployment using upgrade tools and a well defined implementation process.
Pilot Process	In order to minimize potential exposure and to more safely capture any remaining production issues, a software pilot is recommended. The individual pilot plan should consider pilot selection, pilot duration, and measurement.
Implementation	After the completion of the pilot phase, the Cisco IOS implementation phase should begin. The implementation phase may include several steps to ensure software upgrade success and efficiency including slow-start, final certification, upgrade preparation, upgrade automation, and final validation.

Operations – Managing the High Availability Cisco IOS Implementation—Best practices for Cisco IOS operations include software version control, Cisco IOS Syslog management, problem management, configuration standardization, and availability management.

Best Practice	Detail
Strategies and Tools for Cisco IOS Operations	The first strategy of Cisco IOS operations is to keep the environment as simple as possible, avoiding variation in configuration and Cisco IOS versions. The second strategy is the ability to identify and quickly resolve network faults.
Software Version Control	Software version control is the process of implementing only standardized software versions and monitoring the network to validate or possibly change software due to non-version compliance.
Proactive Syslog Management	Syslog collection, monitoring, and analysis are fault management processes recommended to resolve more Cisco IOS specific network problems that are difficult or impossible to identify by other means.

Problem Management	Detailed problem management processes that define problem identification, information collection, and a well analyzed solution path. This data can be used to determine root-cause.
Configuration Standardization	Configuration standards represent the practice of creating and maintaining standard global configuration parameters across like devices and services resulting in enterprise wide global configuration consistency.
Availability Management	Availability management is the process of quality improvement using network availability as the quality improvement metric.

Software Life-Cycle Management Process Overview

Cisco IOS software life-cycle management is defined as the set of planning, design, implementation, and operational processes that are recommended for reliable software implementations and high availability networking. This includes processes to select, validate, and maintain Cisco IOS versions in the network.

The goal of Cisco IOS software life-cycle management is to improve network availability by lowering the probability of production identified software defects or software related change/upgrade failures. The best practices defined within this documentation have been shown to reduce such defects and change failures based on the practical experience of many Cisco customers and the Cisco Advanced Services team. Software life-cycle management may initially increase expenses, however, lower overall cost of ownership can be realized from fewer outages and more streamlined deployment and support mechanisms.

Planning – Building the Cisco IOS Management Framework

Planning is the initial phase of Cisco IOS management needed to help an organization determine when to upgrade software, where to upgrade, and what process will be used to test and validate potential images.

Best practices include software version track definitions, upgrade cycle and definitions, and the creation of an internal software certification process.

Strategy and Tools for Cisco IOS Planning

Begin Cisco IOS management planning with an honest assessment of current practices, the development of achievable goals, and project planning. Self-assessment should be done by comparing best practices in this document to processes within your organization. Basic questions should include the following:

- Does my organization have a software certification process that includes software testing/validation?
- Does my organization have Cisco IOS software standards with a limited amount of Cisco IOS versions running in the network?
- Does my organization have difficulty determining when to upgrade Cisco IOS software?
- Does my organization have difficulty deploying new Cisco IOS software both efficiently and effectively?

- Does my organization have Cisco IOS stability issues following deployment that seriously impact the cost of downtime?

Following the assessment, your organization should begin defining goals for Cisco IOS software management. Start by pulling together a cross-functional group of managers and/or leads from architecture planning groups, engineering, implementation, and operations to help define Cisco IOS goals and process improvement projects. The goal of the initial meetings should be to determine overall objectives, roles and responsibilities, assign action items, and define initial project schedules. Also, define critical success factors and metrics to determine software management benefits. Potential metrics include:

- availability (due to software issues)
- cost of software upgrades
- time required for upgrades
- number of software versions running in production
- software upgrade change success/failure rates

In addition to overall Cisco IOS management framework planning, some organizations also define on-going software planning meetings to occur monthly or quarterly. The goal of these meetings is to review the current software deployment and to begin planning any new software requirements. Planning may include revisiting or modifying current software management processes, or simply defining roles and responsibilities for the different software management phases.

Tools in the planning phase consist solely of software inventory management tools. The CiscoWorks 2000 Resource Manager Essentials (RME) Inventory Manager is the primary tool used in this area. The CiscoWorks2000 RME Inventory Manager greatly simplifies the version management of Cisco routers and switches through web-based reporting tools that report and sort Cisco IOS devices based on software version, device platform, memory size, and device name.

Software Version Track Definitions

The first Cisco IOS software management planning best practice identifies where software consistency can be maintained. A software track is defined as a unique software version grouping, differentiated from other areas by unique geography, platforms, module, or feature requirements. Optimally, a network should run only one software version. This greatly lowers software management related costs and provides a consistent and easily managed environment. However, the reality is that most organizations must run several versions in the network because of feature, platform, migration, and availability issues within specific areas. In many cases, the same version does not work on heterogeneous platforms. In other cases, the organization cannot wait for one version to support all their requirements. The goal is to identify the fewest software tracks for the network with consideration for testing/validation, certification, and upgrade requirements. In many cases, the organization may have slightly more tracks to lower testing/validation, certification, and upgrade costs overall.

The first differentiating fact is platform support. Typically, LAN switches, WAN switches, core routers, and edge routers each have separate software tracks. Other software tracks may be needed for specific features or services, such as data-link switching (DLSw), Quality of Service (QoS), or IP telephony, especially if this requirement can be localized within the network.

Another criteria is reliability. Many organizations try to run the most reliable software towards the network core and data center, while offering newer advanced features, or hardware support, towards the edge. On the other hand, scalability or bandwidth features are often most needed in core or data center environments. Other tracks may be needed for specific platforms, such as larger distribution sites that have a different WAN router platform. The following table is an example software track definition for a large enterprise organization.

Track	Area	Hardware Platforms	Features	Cisco IOS Version	Certification Status
1	LAN core switching	6500	QoS	12.1E(A8)	Testing
2	LAN access switch	2924XL 2948XL	Unidirectional Link Detection Protocol (UDLD), Spanning Tree Protocol (STP)	12.0(5.2)XU	Certified 3/1/01
3	LAN distribution/access	5500 6509	Supervisor 3	5.4(4)	Certified 7/1/01
4	Distribution switch Route Switch Module (RSM)	RSM	Open Shortest Path First (OSPF) Routing	12.0(11)	Certified 3/4/02
5	WAN headend distribution	7505 7507 7204 7206	OSPF Frame Relay	12.0(11)	Certified 11/1/01
6	WAN access	2600	OSPF Frame Relay	12.1(8)	Certified 6/1/01
7	IBM connectivity	3600	Synchronous Data Link Control (SDLC) headend	11.3(8)T1	Certified 11/1/00

Track assignments can also change over time. In many cases, features or hardware support may integrate into more mainline software versions allowing different tracks to eventually migrate together. Once track definitions have been defined, the organization can use other defined processes to migrate towards consistency and validation of new versions. Track definitions are also an ongoing effort. Anytime a new feature, service, hardware, or module requirement is identified, a new track should be considered.

Organizations wishing to initiate a track process should start with newly defined track requirements, or in some cases, stabilization projects for existing networks. An organization may also have some identifiable commonalities with existing software versions that can make current track definition possible. In most cases, rapid migration to identified versions is not required if the customer has sufficient network stability. The network architecture, or engineering group, normally owns the track definition process. In some cases, one individual may be responsible for track definitions. In other cases, project leads are responsible for developing software requirements and new track definitions based on individual projects. It is also a good idea to review track definitions on a quarterly basis to determine if new tracks are required, or if old tracks require consolidation or upgrade.

Organizations that identify and maintain software tracks with strict version control have been shown to have the highest success with a decreasing number of software versions in the production network. This generally leads to improved software stability and overall network reliability.

Upgrade Cycle and Definitions

Upgrade cycle definitions are defined as basic quality steps in software and change management used to determine when a software upgrade cycle should be initiated. Upgrade cycle definitions allow an organization to properly plan for a software upgrade cycle and to allocate required resources. Without upgrade cycle definitions, an organization typically experiences an increase in software reliability issues due to feature requirements in the current stable versions. Another exposure could be the organization missing the opportunity to properly test and validate a new version before production usage is required.

An important aspect of this practice is identifying when and to what degree software planning processes should be initiated. This is due to the fact that a leading cause of software problems is turning on a feature, service, or hardware capability in production without due diligence, or upgrading to a new Cisco IOS version with no software management considerations. Another problem is not upgrading. By ignoring normal software cycles and requirements, many customers face the difficult task of upgrading software through a number of different major releases. The difficulty is due to image sizes, default behavior changes, command level interpreter (CLI) changes, and protocol changes.

Cisco recommends a well defined upgrade cycle, based on the best practices as defined in this paper, to be initiated whenever new major feature, service, or hardware support is required. The degree of certification and testing/validation should be analyzed (based on risk), to determine the precise testing/validation requirements. Risk analysis can be done by geographic location, logical location (core, distribution, or access layers), or the estimated number of people/customers affected. If the major feature or hardware capability is contained in the current release, some streamlined upgrade cycle processes should also be initiated. If the feature is relatively minor, consider the risk and then decide which processes should be initiated. In addition, software should be upgraded in two years or less to help ensure that your organization stays relatively current and that the upgrade process is not too cumbersome. Customers should also consider the fact that no bug fixes will be done to software trains that have passed the End Of Life (EOL) status. Some consideration should also be given to business requirements since many environments can tolerate, or even welcome, more feature additions with little or no testing/validation processes and some resulting downtime.

Customers should also consider the newer data gathered in Cisco release operations when considering their testing requirements. An analysis of bugs and root causes showed that the vast majority of bug root causes were the result of developers coding within the impacted software area. This means that if an organization is adding a particular feature or module to their network in an existing release, there is the probability of experiencing a bug related to that feature or module, but a much lower probability that the new feature, hardware, or module will impact other areas. This data should allow organizations to lower testing requirements, when adding new features or modules that are supported in existing releases, by testing only the new service or feature in conjunction with other enabled services. The data should also be considered when upgrading software based on a few critical bugs found in the network.

The following table shows the recommended upgrade requirements for a major high availability enterprise organization:

Software Management Trigger	Software Life–Cycle Requirement
New network service. For example, a new ATM backbone or a new VPN service.	Complete software life–cycle validation including new feature testing (in conjunction with other enabled services), collapsed topology testing, what–if performance analysis, and application profile testing.

New network capability is not supported in the current software release. Examples include QoS and Multiprotocol Label Switching (MPLS).	Complete software life-cycle validation including new feature testing, in conjunction with other enabled services, collapsed topology testing, what-if performance analysis, and application profile testing.
New major feature or hardware module that exists in the current release. For example, adding a new GigE module, multicast support, or DLSW.	Candidate management process. Possible full validation based on release requirements. Possible limited testing/validation if candidate management identifies the current release as potentially acceptable.
Minor feature addition. For example, a TACACS device for access control.	Consider candidate management based on the risk of the feature. Consider testing or piloting the new feature based on risk.
Software in production for two years or a quarterly software review.	Candidate management and business decisions with regards to complete life-cycle management to identify the current supportable release.

Emergency Upgrades

In some cases, organizations face the need to upgrade software due to catastrophic bugs. This can lead to problems if the organization does not have an emergency upgrade methodology. Problems with software can range from unmanaged software upgrades, where software is upgraded with no software life-cycle management, to situations where network devices are continually crashing, but the organization does not upgrade since certification / testing on the next candidate release has not been completed. Cisco recommends an emergency upgrade process for these situations where limited testing and pilots are performed in less business critical areas of the network.

If catastrophic errors occur with no apparent workaround and the problem is software defect related, Cisco recommends that Cisco support be fully engaged to isolate the defect and determine if or when a fix is available. When the fix is available, Cisco recommends an emergency upgrade cycle to quickly determine whether the problem can be repaired with limited downtime. In most cases, an organization is running a supported version of the code and the problem fix is available in an existing newer interim version of the software.

Organizations can also prepare for potential emergency upgrades. Preparation includes the migration to supported Cisco IOS releases and the identification /development of candidate replacement versions within the same Cisco IOS train as the certified version. Supported software is important since it means that Cisco development is still adding bug fixes to the identified software train. By maintaining supported software in the network, the organization reduces validation time due to the more familiar and stable code base. Typically, a candidate replacement is a new interim software image within the same Cisco IOS train with no feature or hardware support additions. A candidate replacement strategy is especially important if the organization is in the early adopter phase of a particular software train.

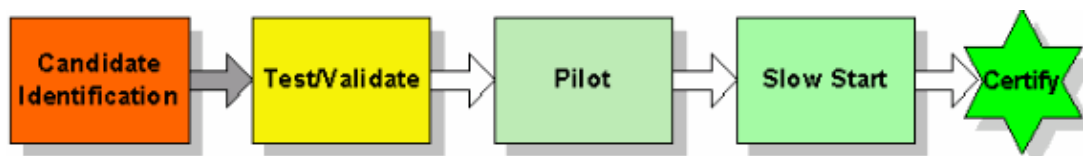
Certification Process

A certification process helps to ensure that validated software is consistently deployed in the organization's production environment. Certification process steps should include track identification, upgrade cycle definitions, candidate management, testing/validation, and some pilot production use. A simple certification process, however, still helps to ensure that consistent software versions are deployed within the identified tracks.

Start a certification process by identifying individuals from architecture, engineering/deployment, and operations to draft and manage the certification process. The group should first consider business goals and resource capabilities to ensure that the certification process will have continued success. Next, assign individuals or groups overall responsibility for key steps in the certification process including track management, life-cycle upgrade definitions, testing/validation, and pilots. Each of these areas should be defined, approved, and formally communicated within the organization.

Also include guidelines for quality or approval at each phase of the certification process. This is sometimes called a quality gate process because certain quality criteria must be met before the process can move to the next step. This helps to ensure that the certification process is effective and is worth the resources assigned. In general, when issues are found with quality in one area, the process pushes the effort back one step.

Software candidates may not meet the defined certification criteria because of software quality or unexpected behavior. When issues are found that impact the environment, the organization should have a more streamlined process to certify a later interim release. This helps to reduce resource requirements and is generally effective if the organization can understand what changed and what defects were resolved. It is not unusual for an organization to experience a problem with an initial candidate and to certify a later interim Cisco IOS release. Organizations may also do a limited certification or provide caveats if some problems exist and can upgrade to a later fully certified release when a new interim has been validated. The flow-chart below is a basic certification process and includes quality gates (a review following each block):



Design – Selection and Validation of Cisco IOS Versions

Having a well defined methodology for selecting and validating Cisco IOS versions helps an organization to reduce unplanned downtime due to unsuccessful upgrade attempts and unplanned software defects.

The design phase includes candidate management and testing/validation. Candidate management is the process used to identify specific versions for the defined software tracks. Testing/validation is a part of the certification process and ensures that the identified software version is successful within the required track. Testing/validation should be done in a lab environment with a collapsed topology and configuration that closely resembles the production environment.

Strategy and Tools for Cisco IOS Selection and Validation

Every organization should have a process for selecting and validating standard Cisco IOS versions for the network starting with a process for selecting the Cisco IOS version. A cross-functional team from architecture, engineering, and operations should define and document the candidate management process. Once approved, the process should be turned over to the appropriate delivery group. It is also recommended that a standard candidate management template be created that can be updated with candidate information as it

is identified.

Not all organizations have a sophisticated lab environment that can easily mimic the production environment. Some organizations skip lab testing because of the expense and the ability to pilot a new version in the network without major business impact. However, high availability organizations are encouraged to build a lab that mimics the production network and to develop a testing/validation process to ensure high test-coverage for new Cisco IOS versions. An organization should allow about six months to build the lab. During this time, the organization should work to create specific test plans and processes to ensure that the lab will be used to its full benefit. For Cisco IOS, this means the creation of specific Cisco IOS testing plans for each required software track. These processes are key in larger organizations due to the fact that many labs go unused for new product and software introductions.

The following sections briefly describe candidate management and testing/validation tools to use for Cisco IOS selection and validation.

Candidate Management Tools

Note: To use most the tools provided below, you must be a registered user and you must be logged in.

- **Release Notes**—Provides information regarding the hardware, module, and feature support of a release. Release notes should be reviewed during candidate management to ensure that all required hardware and software support exists in the potential release, and to understand any migration issues including different default behavior or upgrade requirements.

Testing and Validation Tools

Testing and validation tools are used for testing and validating network solutions including new hardware, software, and applications.

- **Traffic Generators**—Generate multi-protocol traffic streams and raw packet rates used to model the rate across any particular link utilizing specific protocols. Users can specify the source, destination MAC, and socket numbers, These values can be incremented at specified steps or can be configured to be static/fixed or in random increments. Traffic generators can generate the packets for the following protocols:

- ◆ IP
- ◆ Internetwork Packet Exchange (IPX)
- ◆ DECnet
- ◆ Apple
- ◆ Xerox Network Systems (XNS)
- ◆ Internet Control Message Protocol (ICMP)
- ◆ Internet Group Management Protocol (IGMP)
- ◆ Connectionless Network Service (CLNS)
- ◆ User Datagram Protocol (UDP)
- ◆ Virtual Integrated Network Service (VINES)
- ◆ Data Link Packets

Tools are available from Agilent and Spirent Communications.

- **Packet Counter/Capture/Decoder (Sniffer)**—Allows the customer to selectively capture and decode packets at all packet and data-link layers. The tool has the capability to allow the user to specify the filters, which allows the capturing of only specified protocol data. Filters further allow the user to specify capturing the packets matching a particular IP address, port number or MAC address. Tools are available from Sniffer Technologies.

- **Network Simulator/Emulator**—Allows the customer to populate the routing tables of specific routers, based on the production network requirements. Supports the generation of IP Routing Information Protocol (RIP), OSPF, Intermediate System-to-Intermediate System (IS-IS), Interior Gateway Routing Protocol (IGRP), Enhanced IGRP (EIGRP), and Border Gateway Protocol (BGP) routers. Tools are available from PacketStorm Communications and Spirent Communication .
- **Session Emulators**—Generate sliding window multi-protocol traffic streams and are capable of sending multi-protocol traffic streams across the test network towards the receiving device. The receiving device echoes the packets back towards the source. The source device verifies the number of packets sent, received, out of sequence packets, and error packets. The tool also provides the flexibility to define the window parameters in Transmission Control Protocol (TCP), thus closely mimicking the client/server traffic sessions in the lab network. The tools are available from Empirix .
- **Large Scale Network Emulators**—Help in testing the scalability of larger environments. These tools are able to create and easily inject control type traffic into a lab topology in order to more closely mimic a production environment. Capabilities include route injectors, protocol neighbors, and Layer 2 protocol neighbors. Tools are available from Agilent and Spirent Communications .
- **WAN Simulators**—Ideal for testing enterprise application traffic where bandwidth and delay are potentially an issue. These tools allow organizations to locally test an application with the estimated delay and bandwidth to see how the application functions over the WAN. These tools are often used for application development and for application profiling test-types within enterprise organizations. Adtech, a division of Spirent Communications and Shunra provide WAN simulation tools.

Candidate Management

Candidate management is the process of identifying software version requirements and potential risks for the particular hardware and enabled feature sets. It is recommended that an organization spend four to eight hours properly researching software requirements, release notes, software defects, and potential risks before piloting a release. The following outlines the basis for candidate management:

- Identify software candidates via Cisco Connection Online (CCO) tools.
- Risk analysis software maturity, new feature, or code support.
- Identify and track known software bugs, issues, and requirements throughout the life-cycle.
- Identify default configuration behavior of the selected image.
- Maintain back-out and roll-forward candidates for potential candidate changes.
- Bug-scrubs.
- Cisco Advanced Services support.

Identifying software candidates became more complex with the increasing number of Cisco productions and software trains. CCO now has several tools including the Cisco IOS upgrade planner, software search tool, software-hardware compatibility matrix, and the product upgrade tool that can help organizations identify potential release candidates. These tools can be found at <http://www.cisco.com/kobayashi/sw-center/>.

Next, analyze the risk of the potential candidate software. This is the process of understanding where the software currently resides on the maturity curve and then weighing the requirements for deployment with the potential risk of the release candidate. For instance, if an organization is wishing to put early deployment (ED) software into a critical high availability environment, the associated risk and resource requirement for

successful certification should be considered. An organization should at least add software management resources for higher risk situations to ensure success. On the other hand, if a general deployment (GD) version is available that meets the needs of an organization, then less software management resources are needed.

When potential releases and risks are identified, perform a bug scrub to determine if any identified catastrophic bugs exist that would potentially prevent certification. Cisco's Bug Watcher, Bug Navigator, and Bug Watcher Agents can help identify potential problems and should be used throughout the software life-cycle to identify potential security or defect issues.

A new software candidate should also be reviewed for potential default configuration behavior. This can be accomplished by reviewing the release notes for the new software image and by reviewing configuration differences with the potential image loaded on the designated platforms. Candidate management can also include the identification of back-out versions or go-to versions if the chosen version does not meet certification criteria at some point in the process. By watching bugs related to features for a specified track, an organization can maintain potential candidates for certification.

Cisco Advanced Services is also an excellent tool for candidate management. This group can provide further insight into the development process and collaboration between a large number of industry experts in many different vertical-market environments. Typically, the best bug scrubs or candidate management capabilities exist within Cisco support, due to the level of expertise and visibility into production software versions running at other organizations.

Testing and Validation

Testing and validation is a critical aspect of management best practices and high availability networking, overall. Proper lab testing can significantly reduce production downtime, helps to train network support staff, and assists in streamlining network implementation processes. To be effective however, the organization must allocate the necessary resources to build and maintain the appropriate lab environment, apply necessary resources to perform the correct tests, and use a recommended testing methodology that includes measurement collection. Without any of these areas, a testing and validation process may not meet the expectations of an organization.

Most enterprise organizations do not have the recommended test lab environment. For this reason, many organizations have deployed solutions incorrectly, have experienced network change failures, or experienced software problems that could have been isolated in a lab environment. In some environments, this is acceptable, as the cost of downtime does not offset the cost of a sophisticated lab environment. In many organizations however, downtime cannot be tolerated. These organizations are strongly urged to develop the recommended test labs, test types, and test methodologies to improve production network quality.

The Test Lab and Environment

The lab should be an isolated area with enough space for desks, workbenches, test gear, and equipment cabinets or racks. Most large organizations will need between four to ten racks of equipment to mimic the production environment. Some physical security is recommended to help maintain a test environment while tests are in progress. This helps prevent a lab test from being disrupted due to other lab priorities including hardware borrowing, training, or implementation rehearsals. Logical security is also recommended to prevent bogus routes from entering the production network or undesirable traffic from exiting the lab. This can be done with routing filters and extended access-lists on a lab gateway router. Connectivity to the production network is helpful for software downloads and access to the lab network from the production environment.

The lab topology should be able to mimic the production environment for any specific test plans. Reproducing hardware, network topology, and feature configurations is recommended. Of course, reproducing the actual topology is nearly impossible, but what can be done is to reproduce network hierarchy and interaction between the production devices. This is important for protocol or feature interaction between multiple devices.

Some test topologies will be different based on the software test requirements. WAN edge Cisco IOS testing, for instance, should not require LAN type devices or testing and may only require WAN edge routers and WAN distribution routers. The key is to mimic software functionality without duplicating production. In some cases, tools can even be used to mimic large-scale behavior such as protocol neighbor counts and routing tables.

Tools are also needed to help with some test types by improving the ability to mimic the production environment and to collect test data. Tools that help mimic production include traffic collectors, traffic generators, and WAN simulation devices. Smartbits is a good example of a device that can collect and replay network traffic or generate large volumes of traffic. An organization may also benefit from devices that can help collect data, such as protocol analyzers.

The lab also requires some management. Many larger organizations have a full time lab manager who has the responsibility for managing the lab network. Other organizations utilize existing architecture and engineering teams for lab validation. Lab management responsibilities include ordering lab equipment and asset tracking, cabling, physical space management, defining lab rules and direction, lab scheduling, lab documentation, setting up lab topologies, writing test plans, performing lab tests, and managing potential identified issues.

Test Types

Overall, there are many different types of testing that can be done. Before building a complete test lab and test plan that can test everything in a multitude of configurations, an organization should understand the different types of testing, the intent of the testing, and whether or not Cisco engineering, technical marketing, or customer advocacy should or could be responsible for some of the various tests. Customer test plans generally cover the more exposed test types. The following table helps in understanding the different test types, when the tests should be performed, and responsible parties.

Of the tests below, proper testing of an organization's specific feature set, topology, and application mix is normally the most valuable. It is important to know that Cisco performs full feature and regression testing, however Cisco is not able to test your organization's application profile with your specific combination of topology, hardware, and configured features. In fact, it is infeasible to test the full range of features, hardware, modules, and topology permutations. Additionally, Cisco cannot test interoperability with third-party equipment. Cisco recommends that organizations test the precise combination of hardware, modules, features, and topology found in their environment. This testing should be conducted in a lab, with a collapsed topology representing your organization's production environment with other supporting test-types such as performance, interoperability, outage, and burn-in.

Test	Test Overview	Test Responsibility
Feature and Functionality	Determines if basic Cisco IOS features and Cisco hardware modules function as advertised. Feature or module functionality as well as feature configuration options should be tested. Configuration removal and addition should be tested. Basic outage testing and burn-in testing is included.	Cisco device testing
Regression		Cisco regression testing

	Determines if the feature or module functions in conjunction with other modules and features, and if the Cisco IOS version functions in conjunction with other Cisco IOS versions in relation to the defined features. Includes some burn-in and outage testing.	
Basic Device Performance	Determines the basic performance of the feature or module to determine if the Cisco IOS feature or hardware modules meet minimum requirements under load.	Cisco Device Testing
Topology/Feature/Hardware Combination	Determines if features and modules function as expected in a specific topology and module/feature/hardware combination. This testing should include protocol verification, feature verification, show command verification, burn-in testing and outage testing.	Cisco tests standard advertised topologies in labs such as Enterprise solutions engineering (ESE) and networked solutions integration test engineering (NSITE). High availability customers should test feature/module/topology combinations as required, especially with early adopter software and non-standard topologies.
Outage (What-if)	Includes common outage types or behaviors that may occur in a specific feature/module/topology environment and potential functionality impact. Outage testing includes card swapping, link-flaps, device failures, link failures, and card-failures.	Cisco is responsible for basic outage testing. Customers are ultimately responsible for outage performance issues related to the scalability of their individual environment. Outage testing should be done, if possible, in the customer lab environment.
NetworkPerformance (What-if)	Investigates device load in relation to a specific feature/hardware/topology combination. The focus is on device capacity and performance such as CPU, memory, buffer utilization, and link utilization in	Customers are ultimately responsible for device load and scalability. Load and scalability concerns are often raised by Cisco sales or Advanced Services and are often tested with

	<p>relation to a set traffic type and resource requirements for protocols, neighbors, number of routes, and other features. The test helps to ensure scalability in larger environments.</p>	<p>Cisco labs such as the Customer Proof-of-Concept Labs (CPOC).</p>
Bug Fix	<p>Ensures that bug fixes repair the identified defect.</p>	<p>Cisco tests bug fixes to ensure that bug is fixed. Customers should also test to ensure that the bug they have experienced is fixed and that the bug does not break any other aspect of the module or feature. Maintenance releases are regression tested but interim releases are usually not.</p>
Network Management	<p>Investigates Simple Network Management Protocol (SNMP) management capabilities, SNMP MIB variable accuracy, trap support, and Syslog support.</p>	<p>Cisco is responsible for testing basic SNMP features, functionality, and MIB variable accuracy. Customers should validate Network management results and are ultimately responsible for management strategy and methodology for new technology deployments.</p>
Large-scale Network Emulation	<p>Large-scale network emulation uses tools such as Agilent's router simulator and Spirent's test tool suite to simulate larger environments. This may include protocol neighbors, frame-relay permanent virtual circuit (PVC) counts, routing tables sizes, cache entries, and other resources typically required in production that are not in the lab by default.</p>	<p>Cisco customers are generally responsible for the aspects of network simulation testing that reproduces their network environment, which may include the number of routing protocol neighbors / adjacencies and associated routing table sizes and other resources that are in production.</p>
Interoperability	<p>Tests all aspects regarding connectivity to third-party network equipment, especially if protocol or signaling interoperability</p>	<p>Cisco customers are generally responsible for all aspects of interoperability testing.</p>

	is required.	
Burn-in	Investigates router resources over time. Burn-in tests typically require a device to be under some load with investigation into resource utilization including memory, CPU, and buffers over time.	Cisco performs basic burn-in testing. Customer testing is recommended in relation to unique topology, device and feature combinations.

Testing Methodology

Once an organization knows what they are testing, a methodology should be developed for the testing process. The purpose of a best practice testing methodology is to help ensure that the agreed upon testing is comprehensive, well documented, easily reproducible, and valuable in terms of finding potential production problems. Documentation and recreating lab scenarios is especially important for testing later versions or for testing bug fixes found in the lab environment. The steps of a testing methodology are shown below. Some testing steps can also be performed concurrently.

1. Create a test topology that simulates the production environment under test. A WAN edge test environment may include a few core routers and one edge router only, while a LAN test may include more devices that can best represent the environment.
2. Configure features that simulate the production environment. The configuration of lab devices should closely match the expected production device hardware and software configurations.
3. Write a test plan, defining tests and goals, documenting the topology, and defining functional tests. Tests include basic protocol validation, **show** command validation, outage testing, and burn-in testing. An example of a specific test within a test plan is found in the following table:
4. Validate routing and protocol functionality. Document or baseline expected **show** command results. Protocols should include both Layer 2 protocols such as ATM, Frame-Relay, Cisco Discovery Protocol (CDP), Ethernet and Spanning-Tree as well as Layer 3 protocols such as IP, IPX and multicast.
5. Validate feature functionality. Document or baseline expected **show** command results. Features may include global configuration commands and any critical features such as authentication, authorization, and accounting (AAA).
6. Simulate load, which would be expected in the production environment. Load simulation can be done with traffic collectors / generators. Validate expected network device utilization variables including CPU, memory, buffer utilization, and interface statistics with an investigation into any packet loss. Document or baseline expected **show** command results.
7. Perform outage testing where the device and software would be expected to deal with or prevent under load. For example, card removal, link flapping, route flapping, and broadcast storms. Ensure that the correct SNMP traps are being generated based on the features being utilized within the network.
8. Document test results and device measurements as tests should be repeatable.

Test Name	Hot Standby Router Protocol (HSRP) Failover
------------------	---

Test Configuration Requirements	Apply the load to the primary gateway interface. Traffic should roughly be 20% towards the gateway from the user station perspective and 60% incoming towards the user station perspective. Also, increase the traffic to a higher load.
Test Steps	Monitor STP and HSRP via show commands. Fail the primary gateway interface connection and then recover the connection after the information is collected.
Measurements Expected	CPU during failover. Show the interface before, during, and after for the primary and secondary gateway. Show HSRP before, during, and after.
Expected Results	Primary gateway fails over to the other router gateway within two seconds. show commands properly reflect the change. Failover to the primary gateway occurs when connectivity is restored.
Actual Results	
Pass or Fail	
Modifications Required to Achieve Pass	

Device Measurements

During the test phase, carry out and document the following measurements to ensure that the device is performing correctly:

- Memory usage
- CPU loads
- Buffer usage
- Interface statistics
- Route tables
- Specific debugging

The information for measurements varies depending on the particular test being implemented. There also may be additional information for measurement depending on the specific issues being addressed.

For each application that is being tested, measure parameters to ensure there is no adverse performance impact on the given application. This is completed by utilizing a performance baseline that can be used to compare performance pre and post deployment. Examples for application measurement tests include:

- The average time it takes to log onto a network.
- The average time it takes to Network File System (NFS) copy a group of files.
- The average time it takes to launch an application and get prompted with the first screen.
- Other application-specific parameters.

Implementation – Swift and Successful Cisco IOS Deployment

A well defined implementation process allows an organization to efficiently deploy new Cisco IOS versions.

The implementation phase includes the pilot process and the implementation process. The pilot process ensures that the Cisco IOS version will be successful in the environment and the implementation process allows swift and successful larger scale Cisco IOS deployments.

Strategy and Tools for Cisco IOS Deployments

The strategy for Cisco IOS deployments is to perform final certification via a pilot process and rapid deployment using upgrade tools and a well defined implementation process.

Before initiating a network pilot process, many organizations build general pilot guidelines. Pilot guidelines should include expectations for all pilots such as success criteria, acceptable pilot locations, pilot documentation, pilot owner expectations, user notification requirements, and expected pilot durations. A cross-functional team from engineering, implementation, and operations is normally involved in building overall pilot guidelines and a pilot process. Once the pilot process has been created, individual implementation groups can normally conduct successful pilots using the identified best practice methods.

Once a new software version has been approved for deployment and final certification, the organization needs to start planning the Cisco IOS upgrade. Planning starts with the identification of new image requirements including platform, memory, flash, and configuration. The architecture and engineering groups normally define new software image requirements in the candidate management phase of the Cisco IOS management lifecycle. Once the requirements have been identified, each device must be validated, and possibly upgraded, by the implementation group. The CiscoWorks2000 Software Image Manager (SWIM) module can also perform the validation step through validating Cisco IOS requirements against device inventory. When all devices have been validated and or upgraded to the correct new image standards, the implementation group can begin a slow-start implementation process utilizing the CiscoWorks2000 SWIM module as a software deployment tool. Once the new image has been successfully deployed a number of times, the organization can begin a rapid deployment using CiscoWorks SWIM.

Cisco IOS Inventory Management

The CiscoWorks2000 Resource Manager Essentials (RME) Inventory manager greatly simplifies the version management of Cisco routers and switches through web-based reporting tools that report and sort Cisco IOS devices based on software version, device platform, and device name.

Cisco IOS SWIM

CiscoWorks2000 SWIM can assist in reducing the error-prone complexities of the upgrade process. Built-in links to CCO correlate the Cisco online information about software patches with Cisco IOS and Catalyst software deployed in the network, highlighting related tech notes. New planning tools find system requirements and send notifications when hardware upgrades (Boot ROM, Flash RAM) are needed to support proposed software image updates.

Before an update is initiated, the prerequisites of a new image are validated against the target switch or router's inventory data to help ensure a successful upgrade. When multiple devices are being updated, SWIM synchronizes download tasks and allows the user to monitor the progress of the job. Scheduled jobs are controlled through a signoff process, enabling managers to authorize a technician's activities before initiating each upgrade task. RME 3.3 includes the ability to analyze software upgrades for Cisco IGX, BPX, and MGX platforms, greatly simplifying and reducing the time required to determine the impact of a software upgrade.

Pilot Process

In order to minimize potential exposure and to more safely capture any remaining production issues, a software pilot is recommended. Pilots are generally more important for new technology deployments, however many new software deployments will be linked to new services, features, or hardware, where a pilot is more critical. The individual pilot plan should consider pilot selection, pilot duration and measurement. Pilot selection is the process of identifying when and where a pilot should be done. Pilot measurement is the process of collecting the required data to identify success and failure or potential problems.

Pilot selection identifies where and how a pilot will be completed. A pilot may start with one device in a low-impact area and extend to multiple devices in a higher-impact area. Some considerations for pilot selection where impact can be reduced include the following:

- Installed in an area of the network resilient to a single device impact due to redundancy.
- In an area of the network with a minimal number of users behind the selected device who can deal with some possible production impact.
- Consider separating the pilot along architecture lines. For instance, pilot it in the access, distribution, and/or core layers of the network.

The duration of this pilot should be based on the time it takes to sufficiently test and evaluate all of the device features. This should include both burn-in and the network under normal traffic loads. The duration is also dependant upon the step in code upgrade and the area of the network where the Cisco IOS is running. If the Cisco IOS is a new major release, a longer pilot period is preferred. Whereas if the upgrade is a maintenance release with minimal new features, a shorter pilot period will suffice.

During the pilot phase it is important to monitor and document results in a similar manner as initial testing. This can include user surveys, pilot data collection, problem collection, and success/failure criteria. Individuals should be directly responsible for tracking and monitoring pilot progress to ensure all issues are identified and that users and services involved in the pilot are satisfied with the pilot results. Most organizations will certify a release if it is successful in a pilot or production environment. This step is a critical failure in some environments due to a perceived success when no measurement or success criteria are identified or documented.

Implementation

After the pilot phase has been completed within the production network, begin the Cisco IOS implementation phase. The implementation phase includes several steps to ensure software upgrade success and implementation efficiency, including implementation slow start, final certification, upgrade preparation, upgrade automation, and final validation.

Implementation slow-start is the process of slowly implementing a newly tested release to ensure that the image has full exposure to the production environment before final certification and full scale conversion. Some organizations may start with one device and one day of exposure before moving on to two device upgrades the following day and perhaps a few more the following day. When approximately ten devices have been placed in production, the organization may wait up to one to two weeks before final certification of the particular Cisco IOS version. Upon final certification, the organization can more rapidly deploy the identified version with a much higher confidence level.

After the slow start process, all devices identified for upgrade should be reviewed and validated using the device inventory and a matrix of the minimum Cisco IOS standards for bootstrap, DRAM, and flash to ensure that the requirements are met. The data can be acquired through in-house tools, third party SNMP tools, or

through the use of the CiscoWorks2000 RME. The CiscoWorks2000 SWIM does review or inspect these variables prior to implementation. However, it is always a good idea to know what to expect during implementation attempts.

If more than one-hundred similar devices are scheduled for upgrades, it is strongly recommended that an automated method be utilized. Automation has been shown to improve upgrade efficiency and to improve the percentage of device upgrade successes during large deployments, based on an internal upgrade of 1000 devices with and without SWIM. Cisco recommends that CiscoWorks 2000 SWIM be used for large deployments due to the degree of verification that is performed during the upgrade. SWIM will even back out of a Cisco IOS version if a problem is detected. SWIM functions by creating and scheduling upgrade jobs, where a job is configured with the devices, desired upgrade images, and run time of the job. Each job should contain twelve or less device upgrades, and up to twelve jobs can run concurrently. SWIM also verifies that the scheduled Cisco IOS upgrade version is running successfully following the upgrade. It is recommended to allow approximately twenty minutes for each device upgrade (including verification). Using this formula, an organization can upgrade thirty-six devices per hour. Cisco also recommends that a maximum of one-hundred devices be upgraded per evening to reduce potential problem exposure.

Following an automated upgrade, some validation should be done to ensure success. The CiscoWorks2000 SWIM tool can run customized scripts following the upgrade to perform further success verification. Verification includes validating that the router has the appropriate number of routes, ensuring that logical/physical interfaces are up and active, or validating that the device is accessible. The following sample checklist can fully validate the success of a Cisco IOS deployment:

- Did the device properly reload?
- Is the device pingable and reachable via the network management system (NMS) platforms?
- Are the expected interfaces on the device up and active?
- Does the device have the correct routing protocol adjacencies?
- Is the routing table populated?
- Is the device passing traffic correctly?

Operations – Managing the High Availability Cisco IOS Implementation

High availability best practice operations of the Cisco IOS environment helps to reduce network complexity, improve problem resolution time, and improve network availability. The operations section of Cisco IOS management includes strategy, tools, and best practice methodologies recommended for managing Cisco IOS.

Best practices for Cisco IOS operations include software version control, Cisco IOS Syslog management, problem management, configuration standardization, and availability management. Software version control is the process of tracking, validating, and improving software consistency within the identified software tracks. Cisco IOS Syslog management is the process of proactively monitoring and acting upon higher priority Syslog messages generated by Cisco IOS. Problem management is the practice of quickly and efficiently collecting critical problem information for software related issues in order to help prevent future occurrences. Configuration standardization is the process of standardizing configurations to reduce the potential for untested code to be exercised in production and to standardize network protocol and feature behavior. Availability management is the process of improving availability based on metrics, improvement goals, and improvement projects.

Strategies and Tools for Cisco IOS Operations

Many quality strategies and tools exist to help manage Cisco IOS environments. The first key strategy for Cisco IOS operations is to keep the environment as simple as possible, avoiding variation in configuration and Cisco IOS versions as much as possible. Cisco IOS certification has already been discussed, however configuration consistency is another key area. The architecture/engineering group should be responsible for creating configuration standards. The implementation and operations group then have the responsibility to configure and maintain the standards through Cisco IOS version control and configuration standards / control.

The second strategy for Cisco IOS operations is the ability to identify and quickly resolve network faults. Network problems should generally be identified by the operations group before users call them in. Problems should also be resolved as quickly as possible without further impact or change to the environment. A few key best-practices in this area are problem management and Cisco IOS Syslog management. A tool to help quickly diagnose Cisco IOS software crashes is the Cisco Output Interpreter.

The third strategy is consistent improvement. The primary process is to improve a quality-based availability improvement program. By performing root-cause analysis on all issues, including Cisco IOS related issues, an organization can improve test coverage, improve problem resolution times, and improve processes that eliminate or reduce outage impact. The organization can also look at common problems and build processes to resolve those issues faster.

Tools for Cisco IOS operations include inventory management for software version control (CiscoWorks2000 RME), Syslog management to manage Syslog messages, and device configuration managers to manage device configuration consistency.

Syslog Management

Syslog messages are messages sent by the device to a collection server. These messages can be errors (for example, a link going down), or they can be informational, such as when someone has been in to configure a terminal on a device.

Syslog management tools log and track Syslog messages received by routers and switches. Some tools have filters to allow the removal of unwanted messages that can detract from the important ones. Syslog tools should also allow reporting to be created based on the messages received. Reporting can be displayed by time period, device, message type, or message priority.

The most popular Syslog tool for Cisco IOS management is CiscoWorks2000 RME Syslog manager. Other tools are available including SL4NT, a shareware program from Netal and Private I from OpenSystems.

CiscoWorks Device Configuration Manager

The CiscoWorks2000 Device Configuration Manager maintains an active archive and provides an easy way to update configuration changes across multiple Cisco routers and switches. The configuration manager monitors the network for configuration changes, updates the archive when a change is detected, and logs the change information to the Change Audit Service. A web based user interface allows you to search the archive for specific configuration attributes and compare the contents of two configuration files for easy identification of differences.

Cisco Output Interpreter

The Cisco Output Interpreter is a tool used in diagnosing software forced crashes. The tool can help to identify software defects without calling the Cisco Technical Assistance Center (TAC), or it can be used as primary information to the TAC following a software forced crash. This information will generally help expedite a resolution to the problem, at least in terms of the required information collection.

Software Version Control

Software version control is the process of implementing only standardized software versions and monitoring the network to validate or possibly change software due to non-version compliance. In general, software version control is accomplished using a certification process and standards control. Many organizations publish version standards on a central web server. In addition, implementation staff is trained to review what version is running and to update the version if it is not standards compliant. Some organizations have a quality gate process where secondary validation is completed through audits to ensure that the standard is followed during implementation.

During operation, it is not uncommon to see non-standard versions in the network, especially if the network and operations staff are large. This may be due to untrained newer staff, mis-configured **boot** commands, or unchecked implementations. It is always a good idea to periodically validate software version standards using tools such as CiscoWorks 2000 RME that can sort all devices by Cisco IOS version. When non-standards are identified, they should be immediately flagged and a trouble ticket or change ticket be initiated to bring the version to the identified standard.

Proactive Syslog Management

Syslog collection, monitoring, and analysis are fault management processes recommended to resolve more Cisco IOS specific network problems that are difficult or impossible to identify by other means. Syslog collection, monitoring, and analysis help to improve problem resolution time by identifying and resolving many faults proactively before more serious network problems are experienced, or are reported by users. Syslog also provides a more efficient method of collecting a wide variety of problems when compared to consistent SNMP polling for a large number of MIB variables. Syslog collection, monitoring, and analysis is accomplished by utilizing the correct Cisco IOS configuration, Syslog correlation tools, such as CiscoWorks2000 RME, and/or Syslog event management. Syslog event management is done by parsing collected Syslog data for identified critical messages and then forwarding an alert or trap to an event manager for real-time notification and resolution.

Syslog monitoring requires NMS tool support or scripts to help parse and report on Syslog data. This includes the capability to sort Syslog messages by date or time period, device, Syslog message type, or message frequency. In larger networks, tools or scripts may be implemented to parse Syslog data and send alerts or notifications to event management systems or operations and engineering personnel. If alerts for a wide variety of Syslog data are not used, the organization should review higher priority Syslog data at least daily and create trouble tickets for potential problems. In order to proactively detect network problems that may not be seen through normal monitoring, periodic review and analysis of historical Syslog data should be performed to detect situations that may not indicate an immediate problem, but may provide an indication of a problem before it becomes service impacting.

Problem Management

Many customers experience additional downtime due to a lack of processes in problem management. Additional downtime can occur when network administrators try to resolve the problem quickly using a combination of service-impacting commands or configuration changes rather than spending time on problem identification, information collection, and a well-analyzed solution path. Observed behavior in this area includes reloading devices, or clearing IP routing tables before investigating a problem and its root cause. In some cases, this occurs because of first level support problem resolution goals. The goal in all software related issues should be to quickly collect the necessary information needed for root-cause analysis before restoring connectivity or service.

A problem management process is recommended in larger environments. This process should include a certain degree of default problem descriptions and appropriate **show** command collections before escalation to

a second tier. First tier support should never be clearing routes or reloading devices. Optimally, the first level organization should quickly collect information and escalate to a second tier. By spending just a few more minutes initially on problem identification or problem description, a root-cause discovery is much more likely, thus allowing a workaround, lab identification, and bug reporting. Second level support should be well versed in the types of information that Cisco may need in order to diagnose a problem or file a bug report. This includes memory dumps, routing information output, and device **show** command output.

Configuration Standardization

Global device configuration standards represent the practice of maintaining standard global configuration parameters across like devices and services resulting in enterprise wide global configuration consistency. Global configuration commands are commands that apply to the entire device and not to individual ports, protocols, or interfaces. Global configuration commands generally impact device access, general device behavior, and device security. In Cisco IOS this includes service commands, IP commands, vty commands, console port commands, logging commands, AAA/TACACS+ commands, SNMP commands, and banner commands. Also important in global device configuration standards is an appropriate device naming convention that allows administrators to identify the device, device type, and device location based on the Domain Name System (DNS) name of the device. Global configuration consistency is important to the overall supportability and reliability of a network environment because it helps reduce network complexity and enhance network supportability. Support difficulty is often experienced without configuration standardization due to incorrect or inconsistent device behavior, SNMP access, and general device security.

Maintaining global device configuration standards is normally accomplished by an internal engineering or operations group that creates and maintains global configuration parameters for similar network devices. It is also a good practice to provide a copy of the global configuration file in TFTP directories so that they can be initially downloaded to all newly provisioned devices. Also helpful is a web accessible file that provides the standard configuration file with an explanation of each configuration parameter. Some organizations even globally configure similar devices on a periodic basis to help ensure global configuration consistency or periodically review devices for the correct global configuration standards.

Protocol and interface configuration standards represent the practice of maintaining standards for interface and protocol configuration. Protocol and interface configuration consistency improves network availability by reducing network complexity, providing expected device and protocol behavior, and improving network supportability. Protocol or interface configuration inconsistency can result in unexpected device behavior, traffic routing issues, increased connectivity problems, and increased reactive support time. Interface configuration standards should include CDP interface descriptors, caching configuration, and other protocol specific standards. Protocol specific configuration standards may include:

- IP routing configuration
- DLSW configuration
- Access-list configuration
- ATM configuration
- Frame Relay configuration
- Spanning-tree configuration
- VLAN assignment and configuration
- Virtual Trunking Protocol (VTP)
- HSRP

Note: It is possible to have other protocol specific configuration standards depending on what is configured within the network.

An example of IP standards may include:

- Subnet size

- IP address space used
- Routing protocol used
- Routing protocol configuration

Maintaining protocol and interface configuration standards is normally the responsibility of the network engineering and implementation groups. The engineering group should be responsible for identifying, testing, validating, and documenting the standards. The implementation group is then responsible for using the engineering documents or configuration templates to provision new services. The engineering group should create documentation on all aspects of required standards to ensure consistency. Configuration templates should also be created to help enforce the configuration standards. Operations groups should also be trained on the standards and should be able to identify non-standard configuration issues. Configuration consistency is of great assistance in the testing, validation, and certification phase. In fact, without standardized configuration templates, it is nearly impossible to adequately test, validate, or certify a Cisco IOS version for a moderately large network.

Availability Management

Availability management is the process of quality improvement using network availability as the quality improvement metric. Many organizations are now measuring availability and outage type. Outage types may include hardware, software, link/carrier, power/environment, design, or user-error/process. By identifying outages and performing root-cause analysis immediately following recovery, the organization can identify methods to improve availability. Almost all networks that have achieved high availability have some quality improvement process.

Appendix A – Overview of Cisco IOS Releases

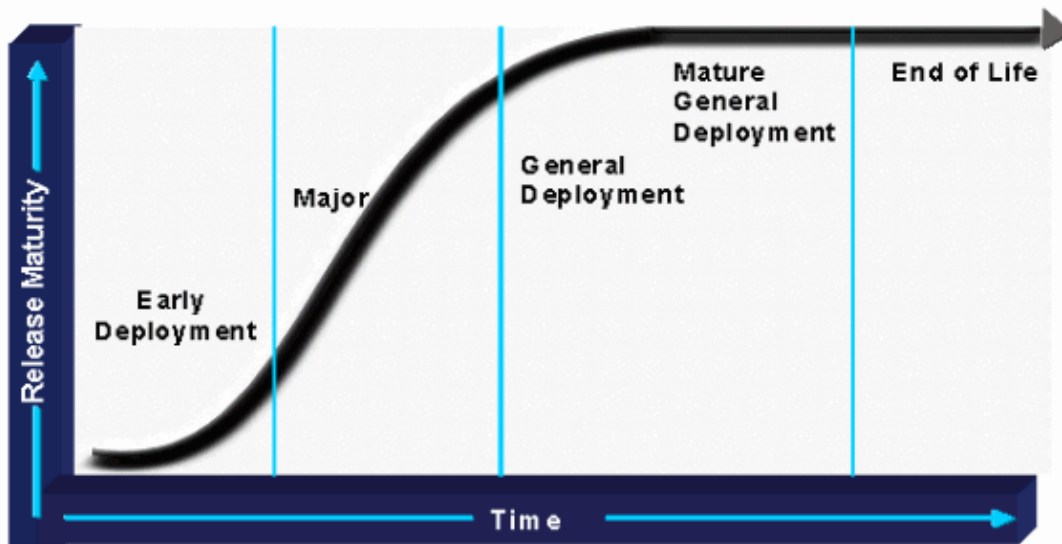
The Cisco IOS software release strategy is built around sound software development, quality assurance, and rapid time to market, which are fundamental to the success of Cisco's customers' networks.

The process is defined around four categories of releases, which are explained below:

- Early Deployment release (ED)
- Major release
- Limited Deployment release (LD)
- General Deployment release (GD)

Cisco creates and maintains an IOS roadmap that has information about individual releases, target markets, migration paths, new features descriptions, and so on.

The figure below illustrates the Cisco IOS software release life-cycle:



ED Releases

Cisco IOS ED releases are vehicles that bring new development to the marketplace. Each maintenance revision of an ED release includes not only bug fixes, but also a set of new features, new platform support, and general enhancements to protocols and the Cisco IOS infrastructure. Every one to two years, the features and platforms of the ED releases are ported to the next mainline Cisco IOS release.

There are four types of ED releases, each with a slightly different release model and life cycle milestones. The ED releases can be classified as:

Consolidated Technology Early Deployment (CTED) releases—The new Cisco IOS release model uses the consolidated ED release train, also known as the "T" train, to introduce new features, new hardware platforms, and other enhancements to Cisco IOS. They are called consolidated technology because they transcend the internal Business Units (BU) and Line Of Business (LOB) definitions. Examples of consolidated technology releases are Cisco IOS 11.3T, 12.0T, and 12.1T.

Specific Technology Early Deployment (STED) releases—STED releases have similar feature commitment characteristics as CTED releases except that they target a specific technology or market theater. They are always released on specific platforms and are solely under the supervision of a Cisco BU. STED releases are identified using two letters appended to the major release version. Examples of STED releases are Cisco IOS 11.3NA, 11.3MA, 11.3WA, and 12.0DA.

Specific Market Early Deployment (SMED) releases—The Cisco IOS SMEDs are differentiated from STEDs by the fact that they target a specific vertical market segment (ISPs, enterprises, financial institutions, Telcom companies, and so on). SMEDs include specific technology feature requirements only for specific platforms of relevance utilized by the intended vertical market. They can be differentiated from CTEDs by the fact that they are only built for specific platforms of relevance to the vertical market, whereas CTEDs would be built for more platforms based on a broader technology requirement. Cisco IOS SMED releases are identified by one alphabetic character appended to the major release version (just like the CTED). Examples of SMEDs are Cisco IOS 12.0S and 12.1E.

Short-lived Early Deployment releases, also known as X Releases (XED)—Cisco IOS XED releases introduces new hardware and technologies to the market. They do not provide software maintenance revisions nor do they provide regular software interim revisions. If a

defect is found in the XED prior to its convergence with the CTED, a software rebuild is initiated and a number is appended to the name. For example, Cisco IOS releases 12.0(2)XB1 and 12.0(2)XB2 are examples of 12.0(2)XB rebuilds.

Major Releases

Major releases are the primary deployment vehicles for Cisco IOS software products. They are managed by the Cisco IOS Technology Division and consolidate features, platforms, functionality, technology, and host proliferation from the previous ED releases. Cisco IOS major releases seek greater stability and quality. For that reason, major releases do not accept the addition of features or platforms. Each maintenance revision provides bug fixes only. For example, Cisco IOS Software Releases 12.1 and 12.2 are major releases.

Major releases have scheduled maintenance updates called maintenance releases that are fully regression tested, incorporate the most recent bug fixes, and support no new platforms or features. The release number of a major release identifies the major release and its maintenance level. In Cisco IOS Software Release 12.0(7), 12.0 is the number of the major release, and 7 is its maintenance level. The complete release number is 12.0(7). Similarly, 12.1 is a major release and 12.1(3) is the third maintenance release of major Cisco IOS Software Release 12.1.

Limited Deployment (LD) Releases

LD is the phase of Cisco IOS maturity between FCS and general deployment for main releases. Cisco IOS ED releases only live in the limited deployment phase because they never attain GD certification.

General Deployment (GD) Releases

At some point during the release life cycle, Cisco will declare a major release to be ready for GD certification. Only a major release can achieve GD status. It meets the GD certification milestone when Cisco is satisfied that the release has been:

- Proven through extensive market exposure in diverse networks.
- Qualified by metrics analyzed for stability and bug trends.
- Qualified through customer satisfaction surveys.
- A reduction in the normalized trend of customer found defects in the release over the previous four maintenance releases.

A customer advocacy GD certification cross-functional team composed of TAC engineers, Advanced Engineering Services (AES) engineers, System Test Engineering, and Cisco IOS Engineering is formed to evaluate every outstanding defect of the release. This team gives the final approval for GD certification. Once a release attains GD status, every subsequent revision of the release is also GD. Consequently, once a release is declared GD; it automatically enters the restricted maintenance phase. While in this phase, engineering modification of the code, including bug fixes with major code rework, is strictly limited and controlled by a program manager. This ensures that no adverse bug is introduced to a GD-certified Cisco IOS software version. GD is achieved by a particular maintenance version. Subsequent maintenance updates for that release are also GD releases. For example, Cisco IOS Software Release 12.0 got the GD certification at 12.0(8). Thus, Cisco IOS Software Releases 12.0(9), 12.0(10), and so on are GD releases.

Experimental or Diagnostic Images

Experimental or diagnostic images are sometimes referred to as engineering specials and are only created when critical software issues have been identified. These images are not part of the normal release process.

Images in this category are customer specific builds designed to help diagnose a problem, to test a bug fix, or to provide an immediate fix. An immediate fix may be provided when it is not an option to wait for the next interim or maintenance release. Experimental or diagnostic images may be built on any supported software base including maintenance or interim versions of any release type. No official naming conventions exist, but in many cases the developer will add initials, exp (for experimental), or additional digits to the base image name. These images are only supported on a temporary basis, in conjunction with Cisco development, because the Cisco TAC and Cisco IOS release operations does not maintain supporting documentation such as symbol tables, or base image history. These images undergo no Cisco internal testing.

Release Life-Cycles Milestones

At some point, GD releases are replaced by newer releases with the latest networking technologies. Therefore, a release retirement process has been established with the following three principal milestones:

End of Sales (EOS)—For major releases, the EOS date is three years after the First Commercial Shipment (FCS) date. This sets a final date for which the release can be purchased for new systems. The EOS release continues to be available for downloading from Cisco Connection Online (CCO) for maintenance upgrades.

End of Engineering (EOE)—The EOE release is the last maintenance release for the GD release, and typically follows about three months after the EOS release. Customers may continue to receive technical support from the Cisco TAC, as well as download the EOE release from CCO. The product bulletin announcing the EOS and EOE releases and dates are published one year before the planned EOS date. At this time, customers should begin to investigate upgrading their Cisco IOS software to take advantage of the latest networking technologies.

End of Life (EOL)—At the end of the release life cycle, all support for the Cisco IOS software release is terminated and no longer available for downloading on the EOL date. In general, the EOL date is five years after the EOE date. An EOL product bulletin is published approximately one year prior to the actual EOL date.

Cisco IOS Version Naming Convention

The Cisco IOS image naming convention provides a complete profile of all released images. The name always includes the major release identifier and the maintenance release identifier. The name may also include a train designator, a rebuild designator (for the maintenance release), business unit (BU) specific feature designators, and BU specific feature–designator rebuild–identifiers. The format can be broken down as follows:

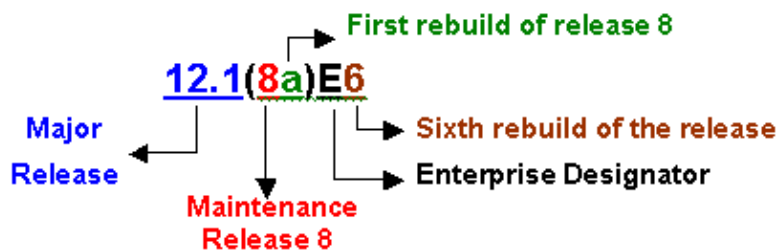
[x.y (z[p])] [A] [o [u(v[p])]] 12.1(8a)E6

Naming Convention Section	Explanation
x.y	A combination of two separate (one or two) digit identifiers separated by a '.' that identifies the Major Release Value. This value is determined by Cisco IOS marketing. Example: 12.1
z	One to three digits that identifies the maintenance release of x.y. This occurs every

	<p>eight weeks. The values are 0 at beta, 1 at FCS, and 2 for the first maintenance release.</p> <p>Example: 12.1(2)</p>
p	<p>One alpha character that identifies a rebuild of x.y(z). The value starts with a lowercase "a" for the first rebuild, then "b", and so on and so forth.</p> <p>Example: 12.1(2a)</p>
A	<p>One to three alpha letters are the designator of the release train and are mandatory for CTED, STED, and X releases. It also identifies a family of products or platforms. Technology ED releases use two letters. The first letter represents the technology and the second letter is used for differentiation. For example:</p> <p style="padding-left: 40px;">A = Access Server/Dial technology (example:11.3AA) B = Broadband (example:12.2B) D = xDSL technology (example:12.2DA) E = Enterprise feature set (example:12.1E) H = SDH/SONET technology (example:11.3HA) N = Voice, Multimedia, Conference (example:11.3NA) M = Mobile (example:12.2MB) S = Service Provider (example:12.0S) T = Consolidated Technology (example:12.0T) W = ATM/LAN Switching/Layer 3 (example:12.0W5)</p> <p>An "X" in the first position of the release name identifies a one-time release based on the CTED "T" train. For example, XA, XB, XC, and so on. An "X" or "Y" in the second position of the release name identifies a short-lived ED release based on, or affiliated to, an STED release. For example, 11.3NX (based on 11.3NA), 11.3WX (based on 11.3WA), and so on.</p>
o	<p>Optional one or two digit numeric designator that identifies a rebuild of a particular release value. Leave blank if not representing a rebuild. Starts with 1, then 2, and so forth.</p>

	Example: 12.1(2)T1, 12.1(2)XE2
u	One or two digit numeric designator that identifies the functionality of the BU-specific release. The value is determined by the BU marketing team. Example: 11.3(6)WA4, 12.0(1)W5
v	One to two digit numeric designator that identifies the maintenance release of the BU-specific code. The values are 0 at beta, 1 at FCS, and 2 as the first maintenance release. Example: 11.3(6)WA4(9), 12.0(1)W5(6)
p	One alpha character designator that identifies a rebuild of a specific technology release. The value starts with a lowercase "a" for the first rebuild, then "b", and so on. Example: 11.3(6)WA4(9a) would be a rebuild of 11.3(6)WA4(9).

The following graph labels the different sections of the Cisco IOS naming convention:



Appendix B – Cisco IOS Reliability

Cisco IOS reliability is an area where Cisco continually strives to improve. Before discussing customer oriented best practices, some understanding of Cisco internal IOS quality and reliability efforts is needed. These sections are primarily intended to provide an overview of Cisco's more recent efforts in Cisco IOS software quality and what customer assumptions should be made regarding software reliability.

Cisco IOS Quality Program

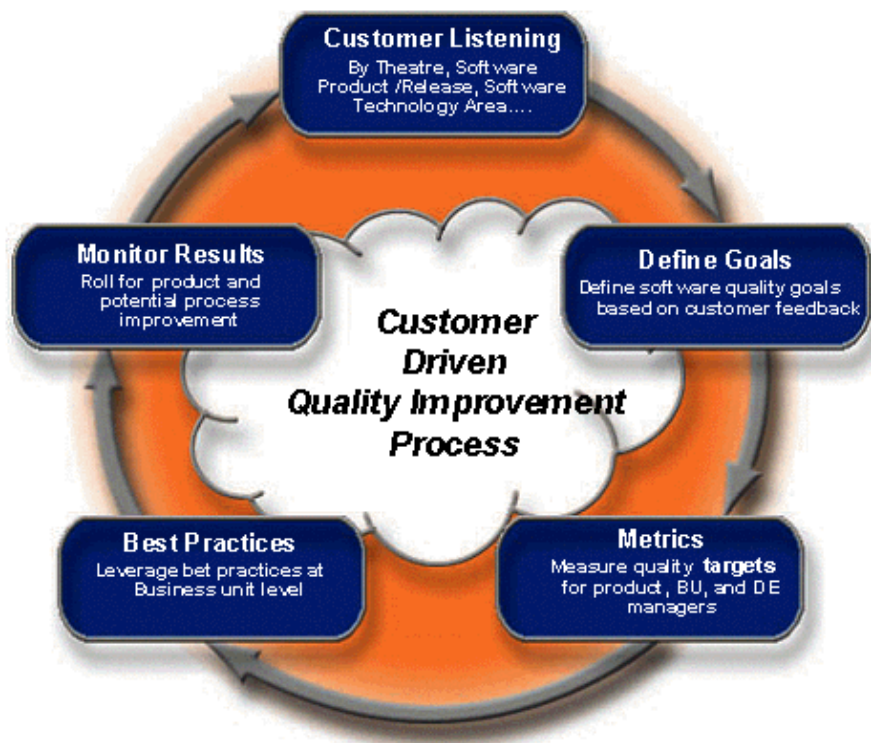
Cisco has a well-defined IOS development process called GEM Great Engineering Methodology (GEM). This process has a three-phase life cycle:

- Strategy and planning
- Execution
- Deployment

General areas within the lifecycle include feature introduction prioritization, development, the testing process,

software introduction phases, First Customer Shipped (FCS), GD, and sustaining engineering. Cisco also follows a number of software quality best-practice guidelines from organizations such as International Standards Organization (ISO), Telcordia (formerly Bellcore), IEEE and the Carnegie Mellon Software Engineering Institute. These guidelines are incorporated into Cisco's GEM processes. Cisco software development processes are ISO 9001 (1994) certified.

The primary process for Cisco IOS software quality improvement is a customer driven process by which Cisco listens to customers, defines goals and metrics, implements best practices, and monitors results. A cross-organizational team that is committed to improving software quality drives this process. A diagram of the Cisco IOS quality improvement process is shown below:



The quality improvement process has distinct measurable goals for FY2002 and beyond. The primary focus of these goals is to reduce defects by identifying software issues earlier in the testing cycle, to reduce the defect backlog, to improve feature consistency and software release clarity, and to provide consistent predictable release schedules and software quality. Initiatives to address these areas include new test coverage tools (identifying areas of weaker test coverage), test corrective action process improvement, and Cisco IOS system regression testing enhancements. Additional resources have been applied to address these issues and there is executive and cross-functional commitment for all primary Cisco IOS software releases.

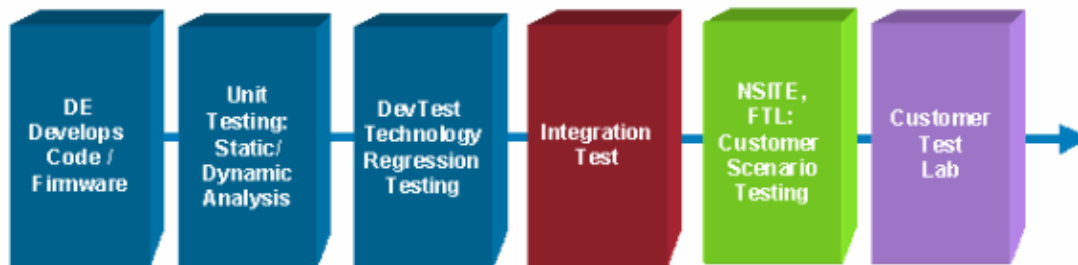
Cisco IOS Release Testing

An integral part of the software reliability quality effort within Cisco is the quality, scope, and coverage of testing. Overall, Cisco has the following IOS quality goals:

- Reduce found Cisco Internal regression defects. This includes higher quality in development and identification of more problems in static/dynamic analysis.
- Reduce customer found defects
- Reduce total outstanding defects

- Increase software release clarity and feature consistency
- Provide feature and maintenance releases with schedules and quality

Cisco internal testing can be thought of as a process where different defects are identified in different stages of testing. The overall goal is to find the right kinds of defects in the right lab. This is important for several reasons. The first and most important is that adequate test coverage may not exist in later testing stages. Testing costs also increase dramatically from stage to stage due to the ability to automate at earlier stages and the increasing complexity and expertise required later. The following diagram depicts the testing spectrum for Cisco IOS.



The first stage is software development. Cisco has several efforts in this area to help improve initial software quality. Development groups also perform code reviews or even multiple code reviews to ensure that other developers approve software changes or new feature code.

The next stage is unit testing. Unit testing utilizes tools that examine software interaction without the use of a lab. DevTest are lab tests that include feature/functionality testing and regression testing. Feature/functionality testing is designed to examine the functionality of a given feature. This includes configuration, de-configuration, and the testing of all feature permutations as defined in the feature specification. Regression testing is done in an automated test facility designed to validate feature functionality and behavior on an ongoing basis. The testing is focused primarily on routing, switching, and feature functionality in a number of different network topologies using pings and limited traffic generation. Regression testing is only done on a limited combination of features, platforms, software versions, and topologies due to the extreme number of possible permutations, however over 4000 regression test scripts are utilized today. Integration testing is designed to expand on lab testing capabilities for a more comprehensive suite of products and interoperability. Integration testing also increases testing code coverage by expanding testing to include interoperability tests, stress and performance tests, system tests, and negative testing (testing unexpected events).

The next lab phase provides end-to-end testing for common customer environments. These are shown in the diagram above as Financial Test Lab (FTL) and NSITE, Customer Scenario Testing. FTL was built to provide testing for the mission critical financial community. NSITE is a group that provides more in depth testing for different Cisco IOS technologies. The NSITE and FTL labs focus on areas such as scalability and performance testing, upgradability, availability and resiliency, interoperability, and serviceability. Serviceability focuses on bulk provisioning issues, event management/correlation and troubleshooting under load. Other labs exist within Cisco for different vertical markets to help test these areas.

The final lab shown in the diagram above is identified as the customer lab. Customer testing is an extension of the quality effort and recommended for high availability environments to ensure that the exact combination of features, configuration, platforms, modules, and topology have been fully tested. Test coverage should include network scalability and performance in the identified topology, specific application testing, negative testing in the identified configuration, interoperability testing for non-Cisco devices, and burn-in testing.

Software MTBF

One of the most common metrics of overall reliability is mean time between failure (MTBF). MTBF for software reliability is useful because of the analysis capabilities that have been developed for hardware reliability using MTBF. Hardware reliability can be more accurately determined using some existing standards. Cisco utilizes the *parts count method* based on standard MTBF data from Telcordia Technologies. MTBF software, however, has no corresponding analysis methodologies and must rely on field measurement for MTBF analysis.

For the past three years, Cisco has performed software reliability field measurements for the Cisco internal IT network and this work is documented within Cisco. The work is based on *software forced crashes* for Cisco IOS devices, which can be measured using network management SNMP trap information and uptime information. The study identifies software reliability using a statistical lognormal distribution model for the identified software releases. Mean time to repair (MTTR) of software failure is based on average router restart and recovery times. A six minute recovery time is used for enterprise environments and fifteen minutes is used for larger Internet service providers (ISPs). The result of this ongoing study is that software generally meets fine nines availability when released, or after a few maintenance versions, and is even higher over time, as measured using software forced crashes as the only downtime source. The study identified potential MTBF values as a range between 5,000 hours for early deployment software to 50,000 hours for general deployment software.

The most common rebuttal to this work is that software forced crashes do not include all outage times incurred due to software reliability issues. If this metric is used in quality improvement efforts, it may help improve the rate of software forced crashes but may ignore other critical areas of software reliability. This comment remains largely unanswered due to the difficulty in accurately predicting software reliability using a statistical methodology. Cisco software quality statisticians have concluded that a larger sample set of accurate data would be needed to reliably predict software MTBF using a wider range of outage types. Additionally, the theoretical statistical analysis would be difficult due to variables such as network complexity, expertise of staff to resolve software related issues, network design, features enabled, and software management processes.

At this time, no industry work has been completed to more accurately predict software reliability with field measurements due to the difficulty of accurately collecting this type of sensitive data. Also, most customers don't want Cisco collecting availability information directly from their network due to the proprietary nature of availability data. Some organizations do however collect data on software reliability and Cisco encourages organizations to collect metrics on availability due to software outages, and to perform root-cause analysis on those outages. Organizations with higher software reliability have used this proactive stance to improve software reliability through a number of practices that they can control.

Software Reliability Assumptions

As a result of customer feedback, proactive studies performed by the Cisco IOS Technologies group and root cause analysis performed by the Cisco Advanced Services team, some newer assumptions and best practices have been formed that help to improve software reliability. These assumptions center on testing responsibilities, software maturity or age, features enabled, and the number of software versions deployed.

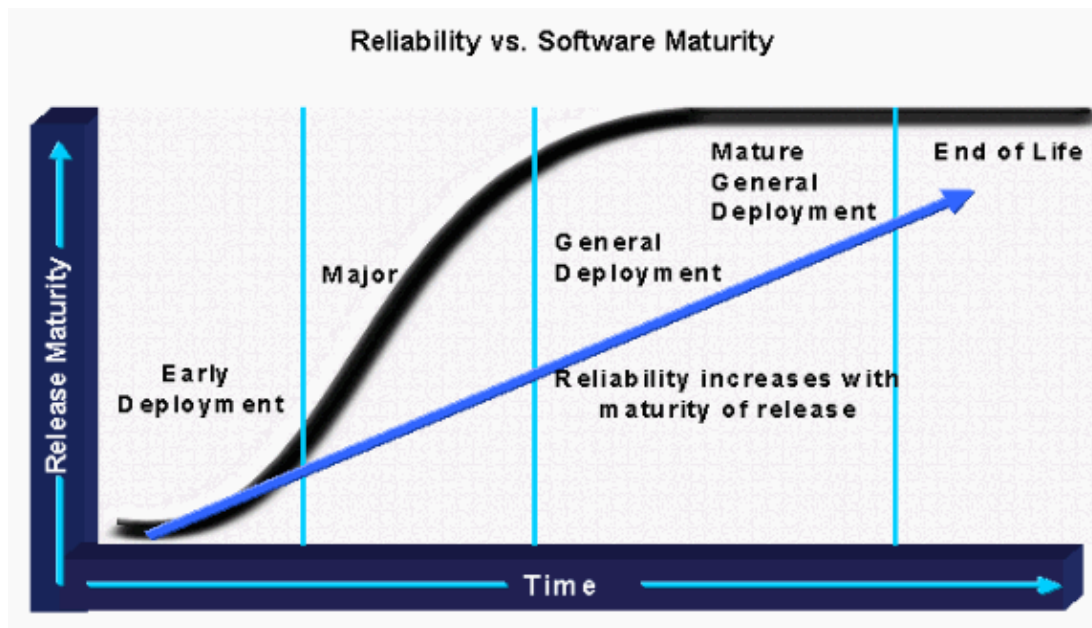
Testing Responsibility

The first new assumption deals with testing responsibility. Cisco is always responsible for testing/validating new features and functionality to ensure that they work in new products. Cisco is also responsible for regression testing to ensure that new software versions are backward compatible. However, Cisco can't validate every feature, topology, and platform against every potential caveat that a customer environment can bring to bear (design idiosyncrasies, load, and traffic profiles). High availability best practices for customers

include testing in a collapsed lab topology that mimics the production network using customer defined features, design, services, and application traffic.

Reliability vs. Software Maturity

Software reliability is primarily a factor of software maturity. Software matures as it receives exposure (usage) and as identified bugs are corrected. Cisco release operations have gone to a train release architecture to ensure that software matures without new features being added. Customers that require high availability are looking for more mature software with the features that they need now. A trade off then exists between the maturity of the software, availability requirements, and the business drivers for new features or functionality. Many organizations have standards or guidelines for acceptable maturity. Some will only accept the fifth interim release of a particular train. For others, it may be the ninth or GD certification. Ultimately, the organization needs to decide their acceptable levels of risk in terms of software maturity.

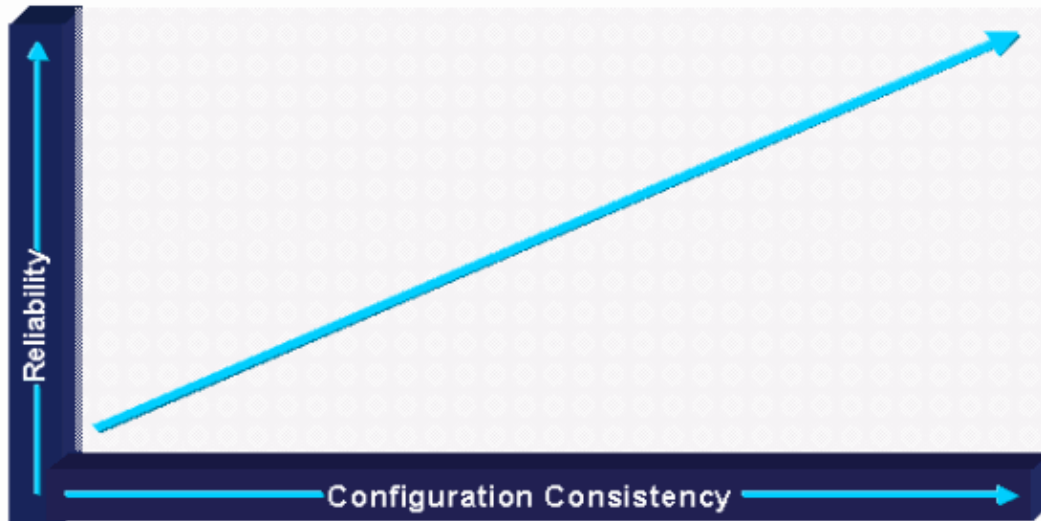


Reliability vs. Quantity of Features and Standards

Software reliability is also a factor of how much of the code is tested and exercised in a production environment. As the quantity of different hardware platforms and modules increases, the amount of code exercised also increases, which generally increases exposure to software defects. The same can be said for the quantity of protocols configured, the variety of configurations, and even the variety of topology or designs implemented. Design, configuration, protocols, and hardware module factors can contribute to the amount of code that is exercised and to the increased risk or exposure to software defects.

Software release operations now have special purpose software that generally limits the code available in one particular area. Business units have recommended designs and configurations that are more thoroughly tested within Cisco and are more widely utilized by customers. Customers have also started to adopt best practices for standardized modular topologies and standard configurations to lower the amount of untested code exposure and to improve overall software reliability. Some high availability networks have strict standard configuration guidelines, modular topology standards, and software version control to help reduce the risk of untested code exposure.

Reliability vs. Configuration Consistency

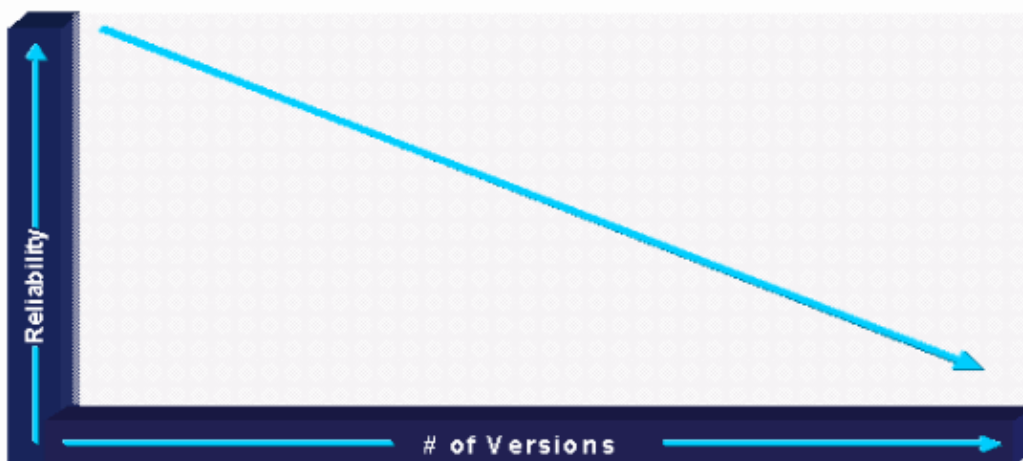


Reliability vs. Number of Deployed Versions

Another factor of software reliability is interoperability between versions and the sheer amount of code that gets exercised with multiple versions. As the quantity of software versions increases, the amount of code exercised also increases, which then increases exposure to software defects. The risk to reliability almost exponentially increases due to additional code exercised with multiple versions. It is now recognized that organizations do need to run at least a handful of versions in the network to cover specific feature and platform requirements. Running over fifty versions in a mostly homogenous network environment however, is normally indicative of software problems due to the inability to properly analyze or validate this many versions.

To improve software reliability, Cisco development performs software regression testing to ensure that different software versions are compatible. In addition, software code is more modular and core modules are less likely to change significantly between versions over time. Cisco release operations have also changed the amount of software available to customers as versions with known defects or interoperability issues are quickly removed from CCO as defects are found.

Reliability vs. Number of Deployed Versions



Related Information

- **A Brief History of Cisco IOS**

All contents are Copyright © 1992—2002 Cisco Systems, Inc. All rights reserved. Important Notices and Privacy Statement.

Updated: Jan 10, 2006

Document ID: 25562
