Software Support and Maintenance Guidelines

The purpose of these guidelines is to provide Government institutions with recommendations on best practices for software support and maintenance. This ensures that software is kept usable and updated. It also ensures that support and maintenance is carried out using a well-defined process that ensures that additional risks are not introduced to software.

- Introduction
  - Purpose of the software support and maintenance guidelines
  - Overview of the software support and maintenance guidelines
  - Scope and applicability of the guidelines
- Roles and Responsibilities
  - Definition and description of key roles involved in software support and maintenance
  - Responsibilities of each role
- Software Support
  - Definition of software support
  - Channels for user support
  - Procedures for issue prioritization and escalation
  - Communication and response time expectations
  - Approach for troubleshooting and resolving software-related problems.
- Software Maintenance

- Overview of software maintenance activities
- Types of software maintenance
- Software change management process
- Testing and quality assurance procedures for software updates

- Software Upgrades and Updates
  - Importance of software updates
  - Avoiding malicious software updates
  - Sources for software updates
  - Software updates deployment and release procedures
  - Risk assessment and impact analysis for updates
  - Backup and rollback procedures in case of update failures

- Security Considerations
  - Importance of security in software support, maintenance, and updates
  - Guidelines for handling security vulnerabilities and patches
  - Best practices for ensuring the integrity and confidentiality of software and its data
  - Compliance requirements related to security.

- Documentation and Knowledge Base
  - Guidelines for documentation of support and maintenance
  - Knowledge base creation
  - Accessibility and availability of documentation for authorized personnel

- Monitoring and Performance Management
  - Tools and processes for monitoring software performance
  - Metrics and key performance indicators (KPIs) for evaluating software support and maintenance effectiveness
  - Regular reporting requirements to management or stakeholders

- Compliance and Legal Considerations
  - Data protection and privacy considerations
  - Intellectual property rights and licensing obligations

Introduction

Introduction

Purpose of the software support and maintenance guidelines

Software maintenance is carried out after a software product has been launched with the objectives of improving the software, correcting issues or bugs, boosting performance and more. Software maintenance is part of the Software Development Life Cycle (SDLC).

The purpose of these guidelines is to provide Government institutions with recommendations on best practices for software support and maintenance. This ensures that software is kept usable and updated. It also ensures that support and maintenance is carried out using a well-defined process that ensures that additional risks are not introduced to software.

The objectives of software maintenance and support are to:

- Correct any faults, bugs or issues reported by users.
- Improve the design of the software.
- Implement enhancements required based on evolving organization needs.
- Interface with other systems.
- Accommodate programs so that different hardware, software, system features, and telecommunications facilities can be used.
- Upgrade the software made on continuous product improvements and technology changes
- Migrate legacy software.
- Retire software.

Introduction

Overview of the software support and maintenance guidelines

Software support is reactive and covers activities to address urgent issues that arise from software use that can hamper software operations or cause downtimes. Software maintenance is proactive and includes scheduled activities to address non-urgent issues or perform planned improvements.

The support and maintenance plan will differ depending on the specifics of the supported software and its underlying infrastructure, as well as the scope of the required support and maintenance activities.

To determine the scope of the support and maintenance activities, you need to analyze:
- The software’s related infrastructure.
- Prospective new features and changes.
- The current and planned number of software users.

These guidelines provide recommendations and best practices on software support and maintenance and include guidance on the support process, maintenance activities, software updates and patch management, software change management and release process. They also define the roles and responsibilities for support and maintenance.

Scope and applicability of the guidelines

This document covers guidelines for support, maintenance, software updates and patch management, software change management and release processes. They apply to all Government institutions in Rwanda.

Roles and Responsibilities

Definition and description of key roles involved in software support and maintenance

The following roles are involved in the software support and maintenance process:

1. End users – End users use software systems on a day to day basis. They are responsible for reporting any issues encountered in the software to the support team. End users also request for enhancements or improvements in the software based on evolving needs
2. IT Support/Support team - The primary function of a support team is to provide technical assistance for any issues that arise with the software that are reported by end users. This includes troubleshooting problems, answering questions about features and functionality, and helping users understand how to use the software
3. Business Analysts – Business Analysts are responsible for analysing the requests for enhancements from users and making recommendations on their implementation
4. Developers – Developers make software changes that require coding to either fix issues or to introduce new features
5. External Vendors – Where the software was acquired from a third party vendor, the vendor may provide some of the maintenance activities such as addressing issues that the institution's support team cannot fix, providing software updates/upgrades and making changes requested by users
6. System administrators – System administrators monitor technical performance of software, manage security and access control and also deploy patches to Production environment

Responsibilities of each role

a) End users responsibilities

1. Report issues encountered on the software
2. Request for enhancements or improvements in the software based on evolving needs
3. Test changes that are made to confirm they work as expected before they are deployed to the production environment

b) Software support team responsibilities

1. Receive and record support requests from users and/or provide a self service help desk system
2. Analyse and troubleshoot issues raised by users
3. Escalate issues they cannot resolve to the relevant teams such as to Software developers
4. Notify users when issues are resolved
5. Provide any training or instructions required to end users on new changes to software
6. Evaluate user feedback and share with relevant teams

c) Business Analysts

1. Analysing the requests for enhancements from users and making recommendations on their implementation
2. Working with developers to design solutions for software enhancements
3. Support users to test new enhancements or updates
4. Support training of users on new enhancements

d) Software developers’ responsibilities:

1. Analyse issues escalated to them to determine the best solution
2. Design algorithms and solutions to fix issues identified
3. Produce clean, efficient code based on specifications
4. Integrate software components and third-party programs
5. Verify and deploy changes made
6. Troubleshoot, debug and upgrade existing software
7. Evaluate user feedback

e) System administrators’ responsibilities:

1. Install and configure software and hardware
2. Manage network servers and technology tools
3. Set up software access accounts and workstations
4. Monitor performance and maintain systems according to requirements.

5. Troubleshoot issues and outages
6. Ensure security through access controls, backups and firewalls
7. Deploying software changes and patches to the Production environment
8. Ensuring availability of software including managing backup and disaster recovery procedures

f) External vendor responsibilities:

For software that is procured from an external vendor, they are also involved in the support and maintenance process. Their responsibilities include:

1. Addressing issues escalated to them by the institution's support team
2. Providing software updates and patches including security patches
3. Making changes requested by the institution
4. Upgrading and enhancing the software as per the product roadmap

Software Support

Definition of software support

Software support is provided to users after software has been deployed to Production and users have started using it. Support is usually provided by an internal support team but may also include support from the vendor of the software system if its third party software. It involves resolving arising issues or challenges that hamper effective use of software. It also includes guidance provided to users on correct use of software.

Channels for user support

The support process is initiated by users reporting issues or faults found in software systems or requests for software changes based on ongoing needs. It is therefore important for Government institutions to provide users with convenient channels for reporting issues and required changes. Below are some of the channels used for software support. Government institutions may provide a combination of one or more of these channels:

- Telephone. The software support team can provide a help desk telephone line where users can call to report issues. This requires a dedicated resource(s) to man the telephone to ensure user calls are responded to on a timely basis.
- Live Chat. Some organisations provide a live chat where users can chat with the software support time on a real time basis. This also requires dedicated resources who can respond to chat requests on a timely basis. Some organisations provide automated chatbots that can respond to common issues.
- Email. The software support team can provide an email address where users can call to report issues.
- Self-Service portal. Users can also report issues through a self service online ticketing system.

Whatever channels are adopted, it is important to have a ticketing system for recording and tracking issues to ensure no issues are lost or forgotten. A ticketing system is a help desk software program used to process, manage, and track user issues from submission to resolution. Ticketing systems automatically organise and prioritise support requests in a central dashboard. Issues can be categorised, prioritised and assigned on the tracking system.

Procedures for issue prioritization and escalation

Issue prioritisation

Once issues are received from users and recorded as tickets, they should be analysed and prioritized based on their impact to the operations of the user departments. For example they can be classified as High, Medium or Low priority based on their criticality.

Prioritisation helps to ensure that the most critical issues affecting operations of the institute are handled first and also ensure efficient use of the available support resources. Reports can be obtained from the system and analysed on a periodic basis to provide visibility on the number of issues raised per system, nature of commonly reported issues, their criticality and how long it takes to close issues. This can inform decisions such as the need for additional support resources to improve user support as well overall health of the software. A system with a consistent high number of critical issues may be and indication of software that required to be upgraded or replaced.

Issue escalation

Ticket escalation is the process followed to move a user issue to a higher-level support agent or manager. The goal of escalating a ticket is to achieve a faster resolution for an issue that may have taken longer to resolve based on its level of priority. There are several steps you can take to improve the ticket escalation process:

1. Understand which issues require escalation.
2. Define clear roles and responsibilities in the support process.
3. Automate your ticket escalation process. This can be through defining timelines for various status of tickets. For example new issues that have not been responded to within a specific timeline can be escalated.
4. Allow the support team to escalate their issues from multiple channels.
5. Keep users informed about the ticket progress.

The escalation process clarifies the boundaries and channels of decision-making throughout an organization in order to solve the problem quickly and with clarity.

Escalation management is also a process put in place to deal with incidents and problem management within a system, ensuring problems get resolved and issues are addressed at the right level.

Communication and response time expectations

The response time is simply the amount of time it takes for the support staff to respond to an issue raised by a user. It is the most important metric when it comes to delivering great user support service. Response times and issue resolution times can be defined through a software Services Level Agreement (SLA). The SLA is a contract between your business and your IT supplier who can be an external vendor or inhouse IT team. The SLA outlines acceptable levels of service as well as compensation you would receive if the supplier fails to provide those services.

Government institutions should develop software Service Level Agreements (SLAs) that inform users on what timelines they should expect for response time as well as issue resolution times. This is regardless of whether the software is internally supported or supported by an external software vendor. In the case of support provided by an external vendor the SLAs should be part of their support contract.

There should be communication to users on expectations for key SLA metrics such as response times. This ensures users understand what to expect and can provide a basis for escalation when the SLAs are not met.

Software Support

Approach for troubleshooting and resolving software-related problems.

Resolution of software related problems typically follows the following steps:

- Identify and record the issue. Issues are typically reported by users and recorded in a support register or online help desk system.
- Assign the issue to someone with the skills to resolve it.
- The issue is assessed to understand the root cause and to determine how it will be fixed. This may require gathering more information about the issue.
- Fix the issue based on the root cause. In some cases where it is not possible to provide a fix in a timely manner a workaround may be provided.
- Track the progress of resolution of the issue and keep users informed.
- Test the issue after it has been resolved. The users should also test and confirm that it works as expected.
- Deploy the fix in Production after they have been tested and inform users of the change

Software Maintenance

Overview of software maintenance activities

Software maintenance is a widely accepted part of the Software Development Life Cycle. It refers to the modifications and updates done after the delivery of a software product. There are number of reasons, why modifications are required, some of which are listed below:

- New user Requirements - Over the time, user may ask for new features or functions in the software based on evolving needs in an organisation.
- Change to the Hosting Environment - If any of the hardware and/or platform (such as operating system) of the target host changes, software changes may be needed to keep adaptability.
- Updates, patches and upgrades – For externally sourced systems, the software vendor may make improvements from time to time to keep pace with technology trends or to enhance security or performance of the system.
- Legal and regulatory changes – Changes to laws and regulations may require changes to software.

Types of software maintenance

There are four types of software maintenance:

- Corrective Software Maintenance
- Adaptive Software Maintenance
- Perfective Software Maintenance
- Preventive Software Maintenance

Corrective Software Maintenance

Corrective software maintenance addresses the errors and faults within software applications that could impact various parts of your software, including the design, logic, and code. These corrections usually come from bug reports that were created by users but corrective software maintenance can also help to spot them before your users do.

Adaptive Software Maintenance

Adaptive software maintenance becomes important when the environment of your software changes. This can be brought on by changes to the operating system, hardware, software dependencies, cloud storage, or even changes within the operating system. Sometimes, adaptive software maintenance reflects organizational policies or rules as well.

Perfective Software Maintenance

Perfective software maintenance focuses on the evolution of requirements and features that exist in your system. As users interact with your applications, they may notice things that you did not or suggest new features that they would like as part of the software, which could become future projects or enhancements. Perfective software maintenance takes over some of the work, both adding features that can enhance user experience and removing features that are not effective and functional. This can include features that are not used or those that do not help users to meet their end goals.

Preventive Software Maintenance

Preventive software maintenance helps to make changes and adaptations to software so that it can work for a longer period of time. The focus of the type of maintenance is to prevent the deterioration of your software as it continues to adapt and change. These services can include optimizing code and updating documentation as needed. Preventive software maintenance helps to reduce the risk associated with operating software for a long time, helping it to become more stable, understandable, and maintainable.

For all businesses and organizations, software maintenance is an essential part of the software development lifecycle. It is absolutely necessary for the success of any software and any evolution into the future. It is important to know that maintenance needs to go much further than fixing issues or bugs – that is only one step of the software maintenance process.

Software change management process

Software change management is the process of managing any modifications to software while minimizing the risk that may be posed by new changes. To mitigate the risk, a structured process should be followed in making any changes to software. Government institutions are should follow the following steps when introducing new software changes:

i. Raise a formal Change Request

A change request should be formally documented requesting for any changes that need to be done to software. This could be an enhancement or modification requested by users to improve the software or to respond to changing operational requirements. The change request describes the changes required and reasons for the change. A change request document or form is typically completed to clearly define the user requirement.

ii. Impact Analysis

An impact analysis is done by the technical team to understand the impact and feasibility of the change requested. This will involve:

- Identifying all the systems and the system products affected by the requested change.
- Assessing any technical impacts of the change e.g. on security or infrastructure.
- Making an estimate of the resources or costs needed to effect the change.
- Analyzing the benefits of the change against the costs.

The impact of the change is updated on the change request document with a recommendation on whether the change should be approved.

iii. Change request approval

The change request should be formally approved based on the impact analysis. The approval should be made by an authorised person in the requesting user department as well the IT department. Any resources required to implement the change should be allocated.

iv. System Release Planning

In this phase, the schedule and the contents of the software release are planned. This includes defining the release process which covers the process for testing and deployment to the production environment.

v. Change Implementation

The implementation of changes involves designing the changes based on the user requirements, writing software code to implement the change and finally testing the changes. The change could

also require changes to software configurations or setups which may not require coding.

vi. Software Change Release

During software release, the following points should be carried out:

- Updating software documentation – relevant documentation should be updated including user manuals, technical specifications, and design documents.
- Training of users – users should be provided with communication and any relevant training that needs to be carried out to the user community regarding the change.
- Making any relevant hardware changes – major software changes may require hardware changes such as enhancement of processing or storage resources.
- Migration of changes - Migration of the changes from test environment to production environment. This is usually during non work hours.
- Configuration and data conversion – any software configuration or data changes that need to be done as a result of the change.
- Communication to users - informing users about the software change after it has been released

Testing and quality assurance procedures for software updates

Testing and quality assurance procedures refer to the processes that should be followed to ensure that software changes are adequately tested to avoid introducing negative impacts such as new bugs or security risks when they are deployed in the Production environment. Before software changes are introduced in the Production environment they should be thoroughly tested. For more guidance on testing refer to the RISA testing and quality assurance guidelines.

Software Upgrades and Updates

Importance of software updates

Software needs to be regularly updated for various reasons including responding to new technology changes, preventing security issues, improving compatibility and enhancing program features. Some of the key reasons for making software updates include:

i. Patch security flaws

Changes to manage security flaws should be made as soon as they are available to address software vulnerabilities that enable cybercriminals to gain unauthorized access to an institution's computing resources.. Threat actors see these vulnerabilities as open doors, enabling them to plant malware for illegal purposes such as data theft, sabotage, fraud or for blackmail purposes.

Malware enables threat actors to take control of computers and steal information. Malware can also encrypt files, documents and other programs so they are unusable. Security patches block these open doors in the software to protect a device from attacks.

It is therefore important for Government institutions to keep software updated by deploying patches and updates provided by the software vendors as these fix vulnerabilities that are identified in the software. Any patches deployed should only be sourced from official sources provided by the vendor.

ii. Get new features

Installing updates may add new features and remove old ones that are no longer necessary. Technology is constantly changing and updates offer the latest features and improvements.

iii. Improve performance

Software vendors may find bugs in a program or need to make necessary enhancements to a program to improve the performance of the software.
Electronic devices also need regular maintenance and routine updates to run their best. Having the latest patches can help prevent software from crashing.

iv. Ensure compatibility

Software manufacturers send updates to ensure their software is compatible with the latest technology. Without updates, older software may not be able to work with newer technology.

Avoiding malicious software updates

Government software maintenance teams and users should also be informed to be aware of fake update messages. These messages are another way threat actors try to access computers and mobile devices.
Malicious messages can appear to users in the following ways:

- **Pop-ups.** They may look real but clicking on a pop-up may lead to a hack. Users should be advised to avoid malicious pop-ups. Updates should only be obtained from verifiable sources such as directly from the vendor website. Updates should also appear on the software's website, app store or within the program.
- **Emails.** The email may state that an update is long overdue. It may even have the company logo with all the correct spelling in the text. Don't click on links or respond to these types of emails. For information about updates, it's best to contact the software's support desk or check the vendor website.

One of the best ways to avoid malware downloads is to let the software automatically update where available. Many external vendor software solutions have an option to automatically run updates when available.
Mobile devices are increasingly being used for work purposes. Users should therefore also be reminded to keep their mobile devices updated. They should check for software update notices in the settings of their mobile devices.

Sources for software updates

A software update is a change or fix to a program that replaces the older version. There may just be a few tweaks to improve the way it works, or it could offer entirely new features. Updates or patches are also done to fix security vulnerabilities. These updates are usually provided at regular intervals by the software manufacturer and are usually downloaded from the internet to the device. Software updates can also come from internal development teams.

It is important to ensure that updates or patches are only obtained from a genuine source provided by the software manufacturer to avoid installing malicious software. Official sources usually include websites provided by the software manufacturer or automatic updates. Government Institutions should follow instructions from the software manufacturer on how to get the updates and how to properly install them.

Software updates deployment and release procedures

Deployment is a shift of software from one controlled environment to another. On the other hand, releases are a collection of changes delivered to the Production environment for users to experience. Applications need several updates, security patches, and code changes. Deploying them across platforms and environments requires proper management of releases.

The following process should be followed to deploy software changes:

Preparation

During this stage, developers test if the code is running smoothly and ready to deploy. Then, they must gather all the code along with all the configuration files, libraries, and resources needed for the seamless functioning of the application. These items combined create a single package to be released. Important activities to carry out during preparation stage:

- Notify all the stakeholders about the upcoming deployment.
- Identify and inform all the collaborators to reduce friction between the teams involved in the software development lifecycle (SDLC) process.
- Check third-party software requirements to ensure everyone knows their uses to minimize any issues in the future.
- Map out the deployment process to ensure everyone is on the same page.
- Have a rollback plan ready in case any critical issues occur during deployment.
- Establish key performance indicators (KPIs) to measure the deployment’s effectiveness and determine whether it succeeds or fails.

Testing

The objective of the testing stage is to validate the software update or change before it reaches the deployment stage. During this phase, the update is deployed to a test environment and subjected to testing. Developers then review the results of the testing to fix any errors before deploying the update to a live environment. Key activities carried out include:

- Conducting unit tests on components of the software affected by the change to verify that the results meet user requirements.
- Integrating unit tests with the continuous integration (CI) tools where available to automatically build and verify each portion and remove bugs easily.
- Creating a staging environment similar to target the production environment and run tests to ensure the code works as intended.
- Run end-to-end tests to check the application’s workflow and its compatibility with other components like hardware and network connectivity.

- Seek feedback from stakeholders including requesting end users to undertake functional testing to determine whether the code is ready for production.
- Use smoke tests after deployment to make sure the software is running smoothly.

Deployment

Once the update is thoroughly tested, you can deploy it to the live environment. At this stage, the main task is to look out for any bugs and errors that might occur on the live server and fix those to deliver the best experience possible. Key activities carried out include:

- Deploy to the production environment to let users interact with the software.
- Monitor performance closely by matching it with the predetermined KPIs and checking for errors.
- Automate rollbacks so you can go to the previous version if the release is unsuccessful.
- Check environment health to identify potential issues with the operating system (OS) or database.
- Keep track of logs to gain insight into how the software is running and identify any security threats.
- Document everything to maintain consistency when you want to change the new version.

The software deployment process can be manual or automated, but it should be done at the least disruptive time in an organization’s work routine.

DevOps tools such as continuous delivery software can enable developers to generate deployment-ready code quickly. This means software releases remain in a deployable state, enabling automatic deployment to production within seconds.

Risk assessment and impact analysis for updates

Risk is the future uncertainty between potential events and the potential for loss. Classic software documentation states that every deployment process should undergo a thorough inspection. This may increase test costs but reduce risks of failure. Some guidelines to mitigate risks of software guidelines include:

- Use legitimate sources of updates. Only download software updates from trusted sources to avoid introducing malicious software
- Implement continuous integration and delivery. Continuous integration and delivery (CI/CD) is a set of practices and tools that automate the building, testing, and deploying of a system. CI/CD enables you to deliver frequent and reliable updates to the system, as well as to detect and fix errors quickly. CI/CD also reduces the risk of conflicts and inconsistencies among the system components, as well as the need for manual interventions. Some common CI/CD tools are Jenkins, Travis CI, and CircleCI.
- Test thoroughly. Testing is a crucial step in ensuring the quality and functionality of a system. Testing should be done at different levels, such as unit testing, integration testing, system testing, and user acceptance testing. The system should also be tested in different environments, such as development, staging, and production. Testing will help you verify the system requirements, identify and resolve defects, and evaluate the system performance and usability. You should also use automated testing tools and frameworks, such as Selenium, JUnit, and TestNG, to speed up and simplify the testing process.
- Monitor and evaluate. Once you deploy the system or make changes to it, you should monitor and evaluate its performance and behavior. You should use tools and metrics that can track and measure the system availability, reliability, scalability, security, and user satisfaction. You should also collect and analyze feedback from the stakeholders, users, and customers, as well as from the logs and reports generated by the system. Monitoring and evaluating will help you identify and troubleshoot any issues or anomalies, as well as to improve and optimize the system over time.
- Consider taking a backup of the software before deployment
- Changes should be deployed outside of office hours and users should be informed of the changes

Backup and rollback procedures in case of update failures

System failures and errors can happen at any stage of a system implementation, from development to testing to deployment. To mitigate this risk the following should be put in place:

- Before starting implementing of a new system or making changes to an existing one, a clear plan of action should be put in place. This includes defining the scope, objectives, requirements, risks, and dependencies of the project, as well as the roles and responsibilities of the team members.
- Define a backup and restore strategy, a rollback and recovery plan, and a communication plan in case of emergencies. Having a plan will helps to avoid or mitigate potential issues and ensure a consistent and quality outcome.
- Use version control - version control is a system that tracks and manages the changes made to the source code, configuration files, and other artifacts of a system. It allows one to create, review, and merge different versions of the system, as well as to revert or roll back to a previous version if something goes wrong. Version control also facilitates collaboration and coordination among the team members, as well as documentation and auditing of the system changes. Some popular version control tools are Git, Subversion, and Mercurial.

Security Considerations

Importance of security in software support, maintenance, and updates

Software security is critical because a malware attack can cause extreme damage to any piece of software while compromising integrity, authentication, and availability. If programmers take this into account in the programming stage and not afterward, damage can be stopped before it begins.

Software maintenance includes regular security updates, patches, and bug fixes, which help keep the software secure from potential threats. This reduces the risk of data breaches and ensures that the business and its customers' data is safe and secure.

Guidelines for handling security vulnerabilities and patches

Vulnerability management refers to the process of discovering, identifying, cataloging, remediating, and mitigating vulnerabilities found in software or hardware. Patch management refers to the process of identifying, testing, deploying, and verifying patches for operating systems and applications found on devices.

To successfully embed patch management into your vulnerability management program, the following steps should be implemented:

- Identify vulnerabilities by conducting regular software vulnerability assessment. The vulnerability identification process enables you to identify and understand weaknesses in your system, underlying infrastructure, support systems, and major applications. A vulnerability assessment is the testing process used to identify and assign severity levels to as many security defects as possible in a given timeframe. This process may involve automated and manual techniques with varying degrees of rigor and an emphasis on comprehensive coverage.
- Prioritize vulnerabilities based on their potential impact. Prioritisation helps to direct resources to address vulnerabilities like to cause the most damage
- Remediate vulnerabilities to reduce risk. This includes deploying patches to fix the identified vulnerabilities. Patches are often used to address security vulnerabilities. If a software vendor discovers a security risk associated with one of its products, it will typically issue a patch intended to address that risk
- Measure the success of your vulnerability management program. Regularly assess effectiveness of your vulnerability management process and make improvements based on the lessons learnt

Best practices for ensuring the integrity and confidentiality of software and its data

Software and data integrity failures frequently occur when the code implementation and the underlying infrastructure lack the ability to protect the code against all integrity violations. This happens when security is not considered during the design or the code is obtained from some untrusted source or repositories. The attackers take advantage of this code and sneak into the system through unauthorized access. As a result, the system becomes vulnerable to the following attacks The system vulnerabilities can cause unimaginable damage to the system. Here are some of the techniques that can prevent such vulnerabilities:

- Authentication: Authentication makes sure that the data is coming from and going to a valid and trusted source.
- User-level restrictions: It is essential to make sure that the libraries and dependencies used in the system are coming from trusted and verified sources and have restricted access.
- Testing: The system’s code should go through extensive testing before deploying it to real users. Testing should be conducted whenever an update is performed or if any configuration changes. This ensures that the vulnerabilities or bugs in the system are identified earlier. This prevents the system from failure.
- Encrypt and validate all data: It is essential to make sure that all data is encrypted before sharing. All data must go through an extensive integrity check and should be backed up by a digital signature. This method will help the user gain access to valid data and protect the system from attackers.
- Firewalls: Build strong firewalls to make sure that no malicious code sneaks into the system.
- Update security checks: Keep updating the system with new and strong security measures.

Compliance requirements related to security.

Security refers to the systems and controls that a company implements to protect its assets, and compliance refers to meeting the standards that a third-party has set forth as best practices or legal requirements.

Security compliance management is an ongoing process of defining security policies, auditing compliance in line with those policies, and ensuring that compliance violations are resolved. Compliance violations must be managed according to policies developed for the specific organization as well as relevant laws and regulations such as the law relating to the protection of personal data and privacy.

Documentation and Knowledge Base

Guidelines for documentation of support and maintenance

Software documentation is any documentation produced to describe various aspects about a particular software system This makes the life of all the members associated with the software easier. It may contain anything from requirements documentation, design documentation, Testing documentation , API documentation, build notes, user manuals or help content. Documentation is a very critical process in software development and maintenance. Poorly documented systems can make it difficult to understand how the system works, making it difficult to carry out activities such as training users, transfer knowledge to new technical teams or to identify and fix problems.

As part of software maintenance, it is important to ensure that software documentation is updated to reflect any changes to software. Documentation that may need to be updated as a result of changes to software includes user manuals, software technical documentation, user manuals and training materials.

Here are some examples of technical software product documentation that should be kept updated:

- Product requirements document
- Product design document
- Technical architecture documentation
- Test documentation
- Source code documentation
- Product roadmap

Here are some examples of user product documentation that should be kept updated:

- Quick start guide
- User manual
- Installation manual
- FAQs and troubleshooting tips

Knowledge base creation

A software knowledge base is a repository or library of information about a software product. Users can consult knowledge base texts to answer their questions and clear up ambiguities regarding the software. The documents in a knowledge base address the users' frequently asked questions and provide solutions to the most common issues.

They focus on problem-solving, often incorporating workarounds or troubleshooting options. Therefore, a knowledge base is often used as a self-service portal. Users can browse self-help articles instead of contacting Customer Support.
It is therefore important as part of software maintenance to ensure that any knowledge bases are updated as part of maintenance activities. For example maintaining a log of issues noted and how they have been resolved.

To make maintenance of software easier it is important to keep updated documentation of the following:

- Configuration document - Configuration management documentation is primarily concerned with the functional relationships among parts, subsystems, and systems for controlling system change. It supports verifying that proposed changes are systematically analyzed to minimize adverse effects.
- Documentation of common issues and their workarounds or resolutions. It is important to keep a record of common known issues and their workrounds and resolutions. This fast tracks

Accessibility and availability of documentation for authorized personnel

Accessible documents benefit all authorized persons by making information clear, direct, easy to understand, and most importantly, usable by a wide audience, including people with disabilities and those who use a variety of devices, software and hardware.

To improve accessibility and availability of documentation , the following should be considered:

- Have a common repository of documentation in the organization where relevant and authorised staff can easily access it . Staff should be informed of such repositories on a regular basis
- Keep the documentation updated to ensure it remains relevant.
- To improve accessibility especially to those with disabilities see RISA Software Accessibility guidelines

Monitoring and Performance Management

Tools and processes for monitoring software performance

System monitoring is the process of collecting and analyzing data about the performance and availability of your IT infrastructure. It helps you identify and troubleshoot issues, optimize resource utilization, and plan for future needs.

Adopt automated tools. Such tools are used to continuously scan your entire network of apps for issues with speed, usability, security, and more. Application performance monitoring is a proactive process that helps you avoid downtime and other potential problems with business-critical apps.

Government institutions should use a system monitoring tool for performance to allow tracking progress, identify challenges, and assess the implementation of a program or project. These tools can include key performance indicators (KPIs), dashboards, checklists, and monitoring plans.

Metrics and key performance indicators (KPIs) for evaluating software support and maintenance effectiveness

Key Performance Indicators help define your strategy and clear focus. Metrics are your “business as usual” measures that still add value to your organization but aren't the critical measure you need to achieve. Every KPI is a metric, but not every metric is a KPI.

Software support and maintenance should set performance metrics which can help them drive continuous improvement. Such KPIs may include Software Uptime/downtime, Average issue response time, average issue resolution time and user satisfaction

Regular reporting requirements to management or stakeholders

Software support and maintenance teams should regularly report their performance to their stakeholders. Reporting should be based on agreed metrics and KPIs such as Software Uptime/downtime, Average issue response time, average issue resolution time and user satisfaction.

It should also including reporting on any resources required to improve performance such as investing in automated tools, training needs or need to hire more support staff.

Compliance and Legal Considerations

Software support and maintenance teams should be aware of applicable laws and regulations and ensure compliance. For example in Rwanda, Government Institutions should comply to the Rwanda Directives on Cyber Security for Network and Information systems and the law on data protection and privacy. In addition Government institutions should follow software guidelines and recommendations provided by RISA.

Data protection and privacy considerations

The terms data protection and data privacy are often used interchangeably, but there is an important difference between the two. Data privacy defines who has access to data, while data protection provides tools and policies to actually restrict access to the data. Compliance regulations help ensure that user’s privacy requests are carried out by companies, and companies are responsible to take measures to protect private user data.

Software maintenance and support teams should be aware of the requirements of Rwanda’s law on data protection and privacy and ensure compliance.

Intellectual property rights and licensing obligations

The definition of intellectual property (IP) rights is any and all rights associated with intangible assets owned by a person or company and protected against use without consent. Intangible assets refer to non-physical property, including right of ownership in intellectual property.

Rwanda Intellectual Property law provides the main branches intellectual property: industrial property and copyright. Industrial Property covers Trademarks, Patents, Utility Models, Industrial Designs, Layout Designs and integrated circuits, Geographic Indication and Protection against unfair competition.

In the context of software , Government Institutions should be aware of the IP rights for any software in use and should only use it in accordance with its license. For example the software may be licensed for use by a specific number of users or licensed to only certain institutions. Violation of license requirements may have legal and financial implications to an institute. In addition, use of software that is not properly licensed may expose the organization to risks such as security risks as it may not receive relevant patches and security updates.