Release management

Release management process is a software engineering process intended to oversee the development, testing, deployment and support of software releases. A release is usually a named collection of software components and supporting documentation that is subject to change management and is upgraded, maintained, tested, statused, and obsolesced as a unit.

The goals of the release management are:

- To manage and implement software release successfully
- To manage and mitigate risks associated with the release
- To release software projects in orderly and repeatable manner

The release management process in general consists of the following steps:

- Requirements gathering and planning: When a new release is prepared, requirements and improvements that are needed to fix the current product are gathered. Once the requirements are known, the next release process can be planned.
- Release Building: When the release requirements are known and planned, the building process of the new release is started. It consists of main tasks used in traditional software development lifecycle.
- Acceptance Testing: When the system release build is ready, it is subjected to a comprehensive testing procedure.
- Release Preparation - A verified release is send to the production environment. The release is packaged, and prepared as a final product to be sent to a specific customer
- Release Deployment - the release is deployed to the customer and implemented

There are two main types of releases: external product releases and internal testing releases. They usually are a named collection of components treated as a single unit, controlled formally, and used as a package exchange between various user groups. For example, the development organization may release a product internally to the testing group or a project may release a product to the user community for beta testing.

In most environments three levels of baseline releases are required for most system: major, minor, and interim. Each level corresponds to a numbered identifier such as M.N.X, where M is the major release number, N is the minor release number, and X is the interim release identifier (or micro release number). A major release represents a new generation of the product or project, while a minor release represents the same basic product but with enhanced features, performance, or quality. Major and minor releases are intended to be external product releases that are persistent and supported for a period of time. An interim release corresponds to a developmental configuration that is intended to be transient.

Once the software is placed in a controlled environment, all changes applied to future releases are tracked. A distinction usually is made for the cause of a release change. The following change categories can be defined:

- Critical failures, which are defects that are nearly always fixed before any external release. In general, these sorts of changes represent blockers that have an impact on the usability of the software in its critical use case.
- Bug or defect that either does not impact the usefulness of the system or can be worked around. Such errors tend to correlate to nuisances in critical use cases or to serious defects in secondary use cases that have a low probability of occurrence.
- A change that is an enhancement rather than a response to a defect. Its purpose is typically to improve performance, testability, usability, or some other aspect of quality that represents good value engineering
- A change that is necessitated by an update to the requirements. This could be new feature or capabilities that are outside the scope of the current vision and business case
- Changes that are not accommodated by other categories. Examples include documentation only or a version upgrade to commercial components

Roles and Responsibilities

The key roles involved with executing the release management process are as follows:

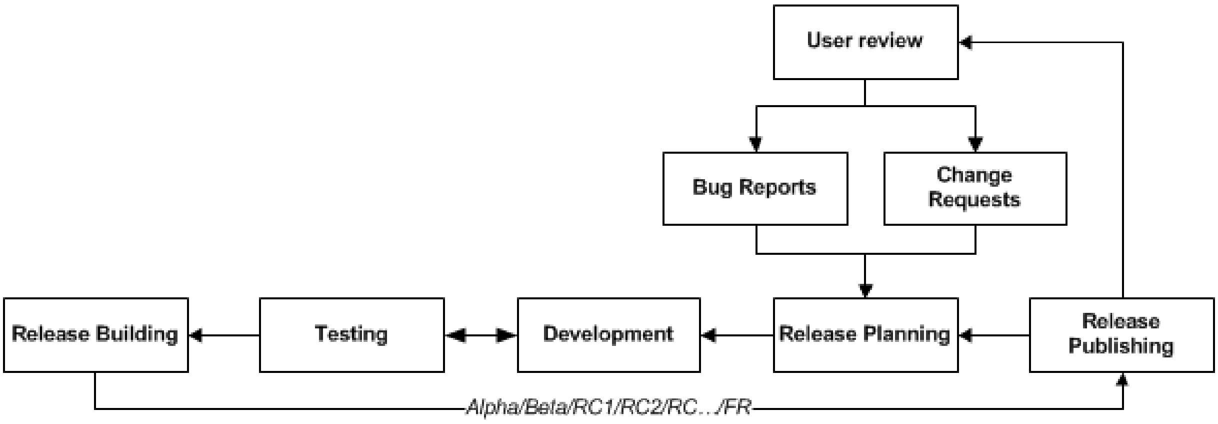
Role	Description and responsibilities
Release Manager	Person(s) who administrates the releasing process, proposes release schedules, and ensures that new release progress timely
Test Manager / Tester	Person responsible for quality assurance and testing procedures
Code Maintainer	Person(s) who maintains the source code in controlled environment
Development Manager / Developer	Person(s) who develops the required functionality and resolves the bugs.

The whole release process is controlled by the Release Manager who coordinates all the release efforts. He plans the release, elaborates the release timeline, and announces the date of feature and code freeze to the developers. After each release, he is also in charge of publishing it for wider audience in order to collect bugs reports or required features.

Release Manager also monitors the gathering requirements process, and development and testing process, constantly evaluating the project progress and alerting stakeholders to risks and issues. The manager uses information from the prior processes of change control initial scope definition and estimation, and approval to assemble the list of prioritized change controls to be included in the release.

SA Tool Kit release management process

The release management process usually begins in the development cycle with requests for changes or new features. If the request is approved, the new release is planned and designed. The new design enters the testing or quality assurance phase, in which the release is built, reviewed, tested and tweaked until it is ultimately accepted as a release candidate. The release then enters the deployment phase, where it is implemented and made available. Once deployed, the release enters a support phase, where bug reports and other issues are collected; this leads to new requests for changes, and the cycle starts all over again.



SA Tool Kit release life cycle

The SA Tool Kit project, under normal conditions, will develop new releases over a six month period to provide a regular and predictable release schedule.

It is proposed that the project will make use of typical three-component release naming system: the first part will be the major number, the second will be the minor number, and the third will be the micro number. A consistent release numbering policy will enable users to look at two release numbers of the system and be able to explain the important differences between those two releases.

The release numbering policy is proposed as follows:

- Changes to the micro number (changes within the same minor line) should be bug fixes only, or very small enhancements to existing features. No new features should be introduced in a micro release. It is expected that they are both forward- and backward-compatible (if user upgrade, for example, from version 2.0.3 to 2.0.4, and then downgrades back to 2.0.3, no functionality should be lost.).
- Changes to the minor number (changes within the same major line) might introduce new features in a minor release, but usually not a significant number of them at once. They must be backward-compatible, but not necessarily forward-compatible.
- Changes to the major number might define compatibility boundaries. A major release is expected to have new features, and may even have entirely new system environment. A new major release can be forward and backward-incompatible.

Release numbers will be groups of digits separated by dots, as for example SAToolKit 2.0.0.

In addition to the numeric components, the releases made for public evaluation might sometimes contain a descriptive label such as "Alpha", "Beta" or "RC" (release candidate), for example SAToolKit 2.1.0 (Alpha), SAToolKit 2.1.0(Beta). An Alpha or Beta qualifier means that this release precedes a future release that will have the same number without the qualifier: 2.1.0 (Alpha) will eventually be 2.1.0 release. In order to allow several such candidate releases in a row, the qualifiers themselves can have meta-qualifiers. For example, a possible series of releases in the order that they would be made available to the public might be as follows:

- SAToolKit 2.1.0 (Alpha 1)
- SAToolKit 2.1.0 (Alpha 2)
- SAToolKit 2.1.0 (Beta 1)
- SAToolKit 2.1.0 (Beta 2)
- SAToolKit 2.1.0 (RC)
- SAToolKit 2.1.0

The release process will be iterative, and each iteration will consists of four main sub-processes:

- Elaboration of release requirements
- Preparation of release timeline
- Building and testing
- Publishing of final release

Release requirements

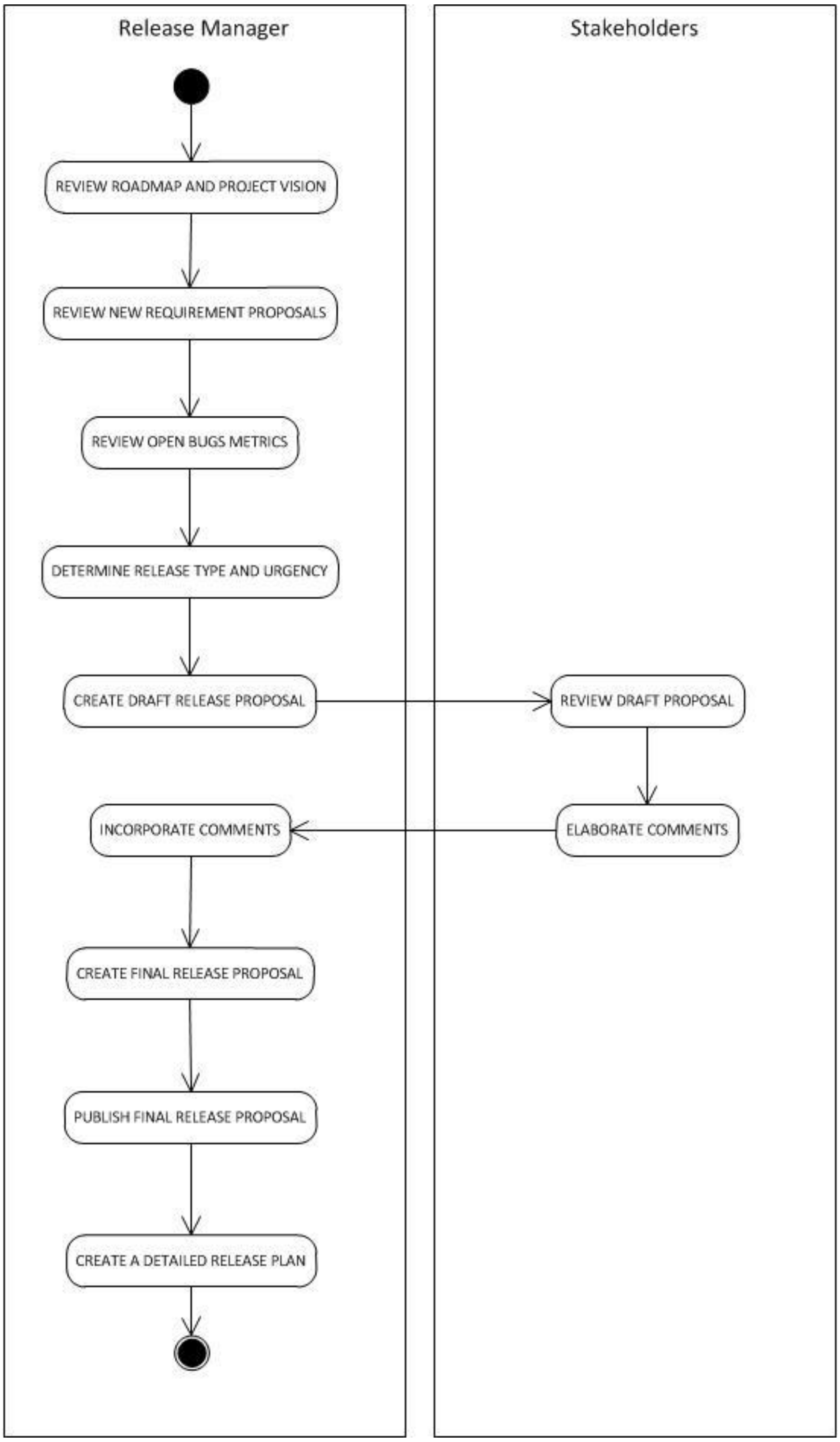
The purpose of the release requirements definition phase is to produce a clear, complete, and testable specification of the technical requirements for the system release.
The Release Manager begins by reviewing project's roadmap and project vision to capture high-level business goals, as well as information from previous releases. Project stakeholders and business users usually participate in the review. This review is further supplemented with the new functionality proposals and information on bugs reported for the previous release of the system.
It is expected that the Release Manager will be confronted with a large number of requirements that can be included in the next incremental release. Implementing them all at once might be impossible, because of limited available resources, time restrictions or conflicts in the wishes formulated by the stakeholders and users.

Users usually prioritize requirements and bugs initially from the perspective of how important each one is to them. Once the development team points out the cost, technical risk, or other trade-offs associated with a specific requirement, the Release Manager might decide it isn’t as essential for the release as they thought. However, the developers might also determine that certain lower priority bugs or changes should be implemented in the release due to their impact on the overall system. In general,

in order to define release priorities, the Release Manager needs to balance the business benefits that each change provides against its costs it has for the system future evolution.
The scope definition resulting from this phase will provide a concise description of all the requests, bug fixes, planned projects, requirements, etc. that are to be in the release, and show their time and resource usage in a high-level release plan. This will allow the various decision makers to make informed choices when deciding where to allocate resources and understand expected schedule for the release.

Finally, the Release Manager formulates an overall concept for the system release, defines scenarios showing how the system will be operated and prepares release proposal document. The finished specification should describe the system in sufficient detail so that software developers can design and build the required release correctly.

At the end of the phase, the Release Manager conducts a release requirements review to demonstrate the completeness and quality of release proposal to project Stakeholders.
When final release proposal is complete, the document is published in OSOR environment.


Release Manager
Stakeholders
REVIEW ROADMAP AND PROJECT VISION
REVIEW NEW REQUIREMENT PROPOSALS
REVIEW OPEN BUGS METRICS
DETERMINE RELEASE TYPE AND URGENCY
CREATE DRAFT RELEASE PROPOSAL
REVIEW DRAFT PROPOSAL
ELABORATE COMMENTS
INCORPORATE COMMENTS
CREATE FINAL RELEASE PROPOSAL
PUBLISH FINAL RELEASE PROPOSAL
CREATE A DETAILED RELEASE PLAN

- **Review roadmap and project vision:** The Release Manager reviews the project’s roadmap and vision in order to establish a general framework for the release.

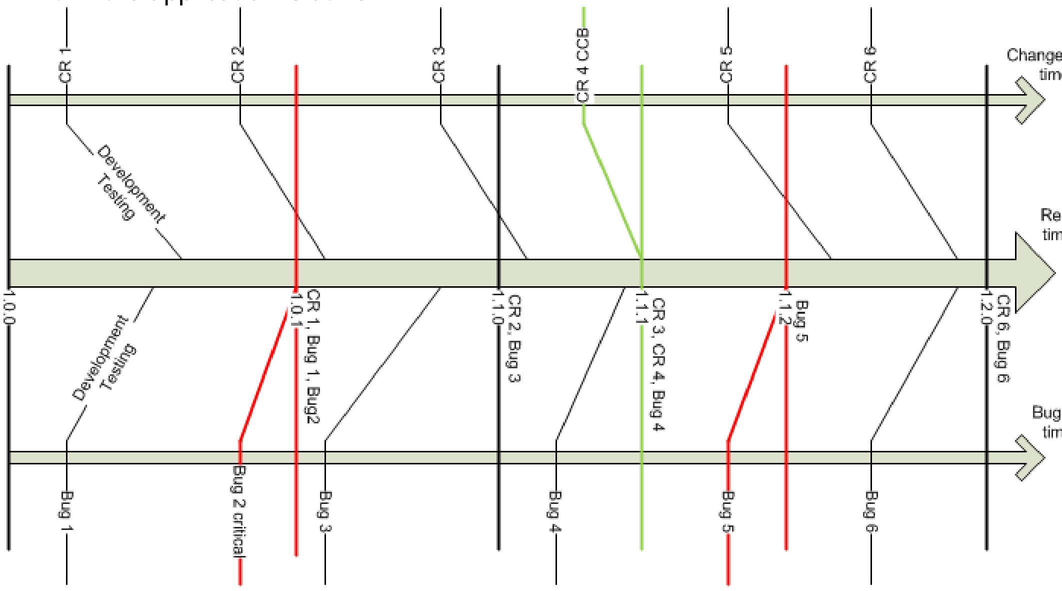
- **Review requirements proposals:** The Release Manager reviews the accepted Change Requests to understand the requested functionality proposals and assess/approve them to subsequent release(s). The assessment and approval process has two parts: the assessment process involves analyzing and recommending specific system change, and the approval process involves evaluating and deciding whether or not to proceed with the change in the scope of current release.

  This early requirements gathering phase is important for ensuring that information and issues from previous releases and projects, and those that have arisen in the interim, are all brought into the release planning process. Examples of this might be:

- Porting base software or applications to new language or platform
- Upgrading software tools
- Upgrading software test tools
- Minor enhancements
- Maintenance work on non-critical projects
- Installation scripts, test scripts, build or make scripts, configuration management, etc.

- **Review bug tracking information:** The Release Manager reviews the reported bugs, which are still opened in order to understand their impact on the current release.

- **Determine release type and urgency:** The Release Manager determines the type (major, minor or micro) and urgency of the release. An urgent micro release might be required if a critical fault was identified during the live operation of the previous system release. However, urgent releases should only be used to resolve critical system errors that directly impacts SA Tool Kit users' activities and where no practical workaround exists or the integrity of data within the application is at risk.



- **Create draft release proposal:** The Release Manager prepares draft release proposal which clearly indicates the scope of new release, including estimates for implementation. The proposal should also ensure an early identification of issues that probably would not be part of the new release.

- **Review and comment draft proposal:** The draft release proposal is reviewed by project Stakeholders. Before a full release proposal is accepted, trade-off decisions will be necessary. Some proposed items may not be allocated in this release but rather be delayed to the next one due to resource issues. Some enhancements may be likewise left out.

- **Incorporate comments:** The Release Manager incorporates the received comments, if any, into the release proposal document.

- **Create final release proposal:** The Release Manager prepares a final release proposal. The release proposal document might be sent to Stakeholders for sign-off.

  The release proposal aim to:

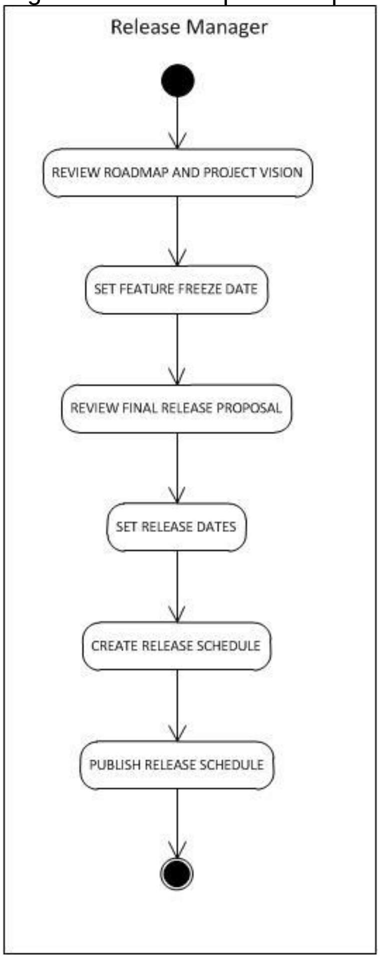
  - Clearly specify the criteria that must be met for each of our public releases (Alpha, Beta, Release Candidate) to publish
  - Document all the requirements for each SA Tool Kit release
  - Establish when a release is ready in terms that all persons involved in a project can understand and in ways that help new people to understand the process and participate.
  - Help others persons who are not directly involved in the release process to

  understand the goals and objectives
  - Focus on the purpose of the release while allocating less time and energy on issues outside the release scope
  - Provide an early warning sign if the release is not on track for shipping on time
  - Make it clear whether a given bug should block the finalization of a release

- **Publish final release proposal**: The final release proposal will be published in the OSOR environment. This assures that users are aware of the forthcoming launch of a new release and are informed of the planned new functionality or the errors the release aims to solve so they can become involved in the process if they wish.

Release timeline

The Release Manager defines the timeline which is based on proposed release scope defined in release proposal. He reviews statuses, gathers feedback from developers, testers and maintainers, and prepares the release schedule. The schedule should have several milestones defined so stakeholders will be able to check progress and catch potential problems earlier.



- **Final release proposal ready:** The final release proposal is accepted and needs to be implemented.
- **Set feature freeze date:** The Release Manager decides when there will be feature freeze dates for the product. Feature freeze date is a date by which all new features need to be integrated into the code base. All features must be completed to ensure that there is enough time to review the new features before the next release and incorporate feedback. After this date, no features will be accepted and integrated for this particular release.

- **Create release schedule:** The Release Manager decides how many candidate releases is necessary and when they will be prepared.

For example, the release's schedule might include:
Development phase

- dd-mm-yyyy – Milestone 1,
- dd-mm-yyyy – Milestone 2,
- dd-mm-yyyy – Milestone 3, Feature Freeze

Alpha/Beta release

- dd-mm-yyyy – Alpha/Beta 1,
- dd-mm-yyyy – Alpha/Beta 2
- dd-mm-yyyy – Code freeze

Release candidate

- dd-mm-yyyy – Release Candidate 1,
  Additional Release candidates as needed.
- dd-mm-yyyy – Final Release.

- **Publish release schedule:** The release schedule is published in OSOR environment

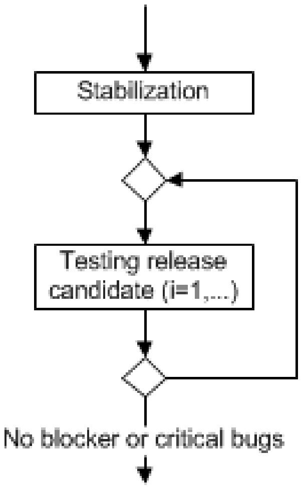
- **Create detailed release plan:** The Release Manager elaborates detailed release plan. The requirements collected during earlier phases of the process should now be translated into an initial high-level plan. The Release Manager must create a picture of the desired upcoming release via a reasonably well-understood set of tasks, deliverables and resource needs.

  In order to plan a new release properly, the following points need to be taken into account:
  - How does the new version affect other areas of the project framework?
  - How should the testing environment be built to reflect the production environment?
  - What are the human and technical resources necessary to ensure the new release is implemented successfully?
  - What is the expected average useful life of the new version?
  - Who will be directly in charge of the various stages of the process?

Building and testing release

The purpose of this phase is to build a complete, high-quality release from the blueprint provided in the final release proposal. The implementation proceeds according to the release plan prepared at the end of previous phase. For each build, developers code and test the system based on the elaborated test cases. When all coding, unit and integration testing for the build are complete, a feature freeze is announced and the development team builds the Alpha release of the system. In the development process, all new feature work should be completed by Feature Freeze date and tested during the Alpha and Beta test releases. At Alpha release milestone, all packages should be testable and feature complete.
At this time, the work concentrates on fixing reported bugs, committing code to source control environment and building daily releases. This is an iterative process and might occur within the community web space.

The transition from alpha stage (i.e. a release which may have serious problems that prohibits its use) to beta stage (i.e. release expected to compile and to perform basic tasks) and from beta stage to the final stage (i.e. release recommended for production) should be a collective decisions of Release Manager, and Testing and Development Managers.
The pre-release tests should be performed by non developer users before every release. This helps to ensure that there is no blocker or critical bug when the release publishes. It is also important that all tests in beta and pre-release phases are done by as many people as possible, to test program using many different computing environments and combinations of preference settings.

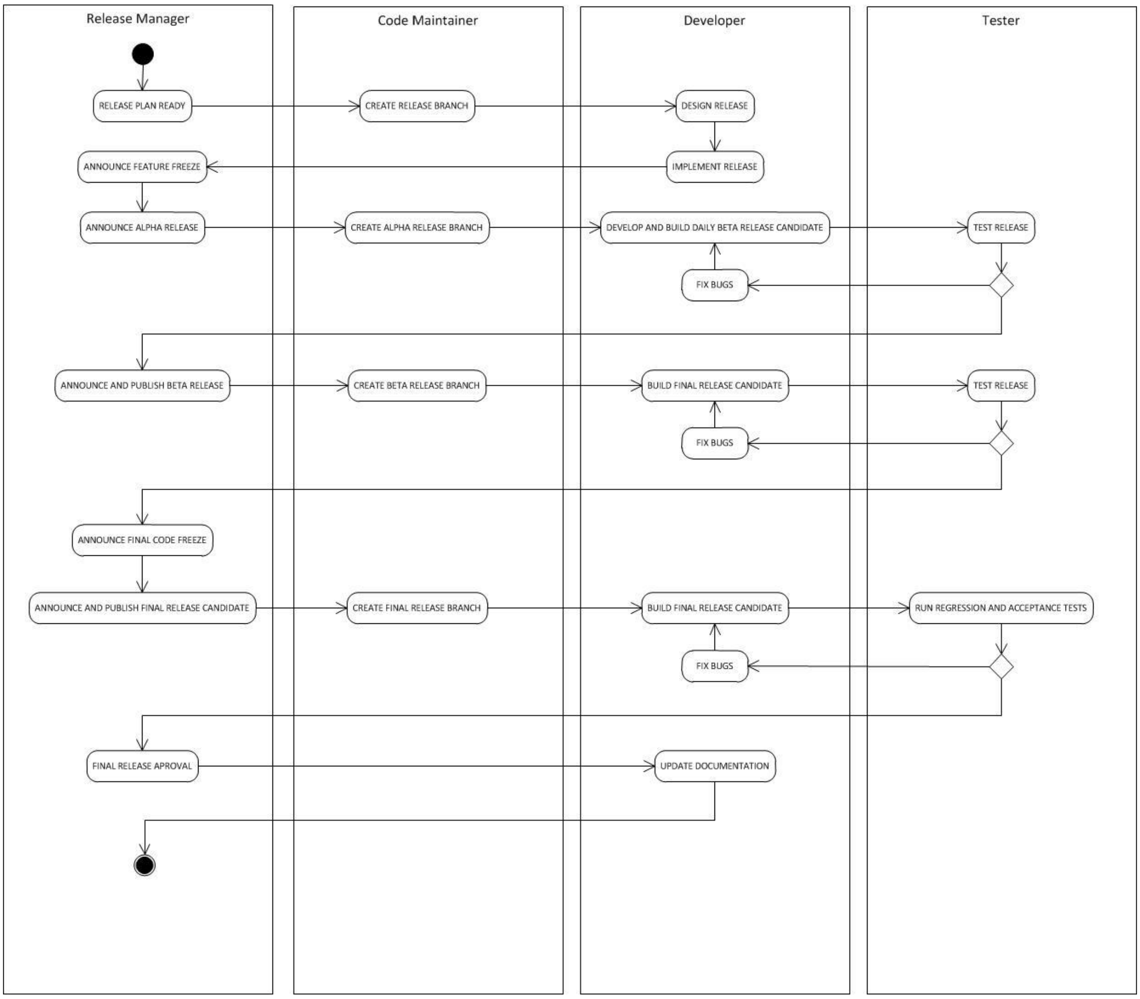

Stabilization
Testing release candidate (i=1,...)
No blocker or critical bugs

Pre-releasing testing.

Finally, every release candidate build is created when there is no know critical bug. If any critical bug is discovered within short period (usually a week) after release candidate is built, the bug has to be fixed and a new release candidate is created. If no critical bug is discovered within this time in release candidate, this build will become the stable release.

During the build and test phase, strict adherence to configuration management procedures is essential. The testers must be able to relate test results to particular modules easily. Therefore, the modules must be under strict configuration control, and the entire set of tests designated for a round of testing should be run on a specific module before any corrections are introduced. Similarly, changes in the test setup or input data must be formally documented and controlled, and all tests should be labeled for the module and environment in which they were run.

Acceptance testing is complete when each test item specified in the acceptance test plan has either been successfully completed or has been waived, and the system has been formally accepted. The phase is concluded when the system source code and final system documentation have been delivered for operational use.


Release Manager
Code Maintainer
Developer
Tester
RELEASE PLAN READY
CREATE RELEASE BRANCH
DESIGN RELEASE
ANNOUNCE FEATURE FREEZE
IMPLEMENT RELEASE
ANNOUNCE ALPHA RELEASE
CREATE ALPHA RELEASE BRANCH
DEVELOP AND BUILD DAILY BETA RELEASE CANDIDATE
TEST RELEASE
FIX BUGS
ANNOUNCE AND PUBLISH BETA RELEASE
CREATE BETA RELEASE BRANCH
BUILD FINAL RELEASE CANDIDATE
TEST RELEASE
FIX BUGS
ANNOUNCE FINAL CODE FREEZE
ANNOUNCE AND PUBLISH FINAL RELEASE CANDIDATE
CREATE FINAL RELEASE BRANCH
BUILD FINAL RELEASE CANDIDATE
RUN REGRESSION AND ACCEPTANCE TESTS
FIX BUGS
FINAL RELEASE APROVAL
UPDATE DOCUMENTATION

- **Release plan ready:** The final release plan is accepted and needs to be implemented.

- **Create release branch:** The Code Maintainer creates a separate branch for the release development.

- **Design and implement release:** The development team incorporates all requirements proposed in the release plan.

- **Announce feature freeze:** Once the Feature Freeze milestone is reached, all new features for the release should be substantially completed and in a testable state.

- **Announce alpha release:** The Release Manager announces the Alpha release of the system that can be packaged up and given out for testing.

  The objectives of the Alpha release are to:
  - Publicly release installable media versions of a feature complete test release
  - Test accepted features of SA Tool Kit
  - Identify as many as possible bugs with severity rating of high or greater and no reasonable workaround
  - Identify as many as possible bugs that hinder execution of required Alpha test plans or dramatically reduces test coverage
  - Identify bugs that relate to an unmet Release Requirement
- **Create Alpha release branch:** The Code Maintainer creates a separate branch for the Alpha release development.

- **Develop and build Beta release candidate:** The development team incorporates changes to the code and builds daily Beta release candidate.

- **Test release:** The testing team executes the tests specified in the system test plans. The results obtained during test execution are documented, and the development team is notified of any discrepancies.

- **Fix bugs:** The development team corrects those software errors that occur because of faults in the design or code. The system bug fixes are handled by development team in accordance with stringent configuration management procedures. Software updates that result from approved corrections are incorporated into a next (possibly daily) release.
  If all SA Tool Kit Alpha release criteria are met, then Beta release might be considered.
- **Announce and publish Beta release:** The Release Manager announces the Beta release. By this date all features must have been fully completed and all code has to be integrated. Only non-invasive changes, user interface work and bug fixes might be incorporated now.

  Test manager has the responsibility of determining whether the criteria for the release have been met through discussion with development team and Release Manager.
  The objectives of the Beta release are to:
  - Publicly release installable media versions of a test release as Beta release is the last widely coordinated test release point in any given release cycle.
  - Finish testing SA Tool Kit features
  - Identify as many as possible bugs with severity rating of high or greater and no reasonable workaround
  - Identify as many as possible bugs that hinder execution of required Beta test plans or dramatically reduces test coverage
- **Create Beta release branch:** The Code Maintainer creates a separate branch for the Beta release development.

- Build daily Final release candidate: The development team incorporates changes to the code and builds daily Final release candidate.

- **Test release:** The testing team executes the tests specified in the system test plans. The results obtained during test execution are documented, and the development team is notified of any discrepancies.

- **Fix bugs:** The development team corrects those software errors that occurred in the last phases of the development. Software updates that result from these corrections are incorporated into a next release.
  If all SA Tool Kit Beta release criteria are met, then the Final release might be considered. If the testing of Beta release results in negative determination, the announcement of final release candidate might be pushed back by one week.
- **Announce final code freeze:** The Release Manager announces the final code freeze in order to prepare for creation the Release Candidate. After the Final Freeze no changes are allowed to any module except for critical release blocking bug fixes.
- **Announce and publish Final release candidate:** The Release Manager announces and publishes in OSOR environment the Final release candidate. No changes to the code are allowed except for the critical issues. If any critical problems are found and fixed, a new release candidate is issued.
  - The objective of the Final release is to provide a final release suitable for meeting the needs of the project audience. In order to be the released to the general public, the Final Candidate (RC) must meet all of the following criteria:
  - All Beta release criteria must be met
  - All critical bugs must be CLOSED
  - The installer must be able to successfully complete an installation from a clean, fully updated default installation of the previous stable SA Tool Kit release
  - All known issues that might have an impact on the system should be documented as Known Issues

- **Create Final release branch:** The Code Maintainer creates a separate branch for the Final release development.

- **Build Final release: The developers build Final release candidate.** If no critical bug is discovered within short period (usually a week) after release candidate is built, the build becomes the Final release

- **Run regression and acceptance tests:** The system is retested by the testing team and regression tests are executed to ensure that previously tested functions have not been adversely affected. Finally the acceptance tests are conducted to demonstrate that system meets its requirements in the operational environment.

- **Fix bugs:** The development team corrects those software errors that occurred in the last phases of the development. Software updates that result from these corrections are incorporated into a next release.
- **Approve Final release**: The Release Manager approves the final release, which is usually the last release candidate that was issued without any show-stopper bugs.

- **Update documentation:** Members of the development team correct any discrepancies arising from the software and finalize the system documentation. They also finalize the user's guide and the system description documents to reflect recent software changes and user comments.

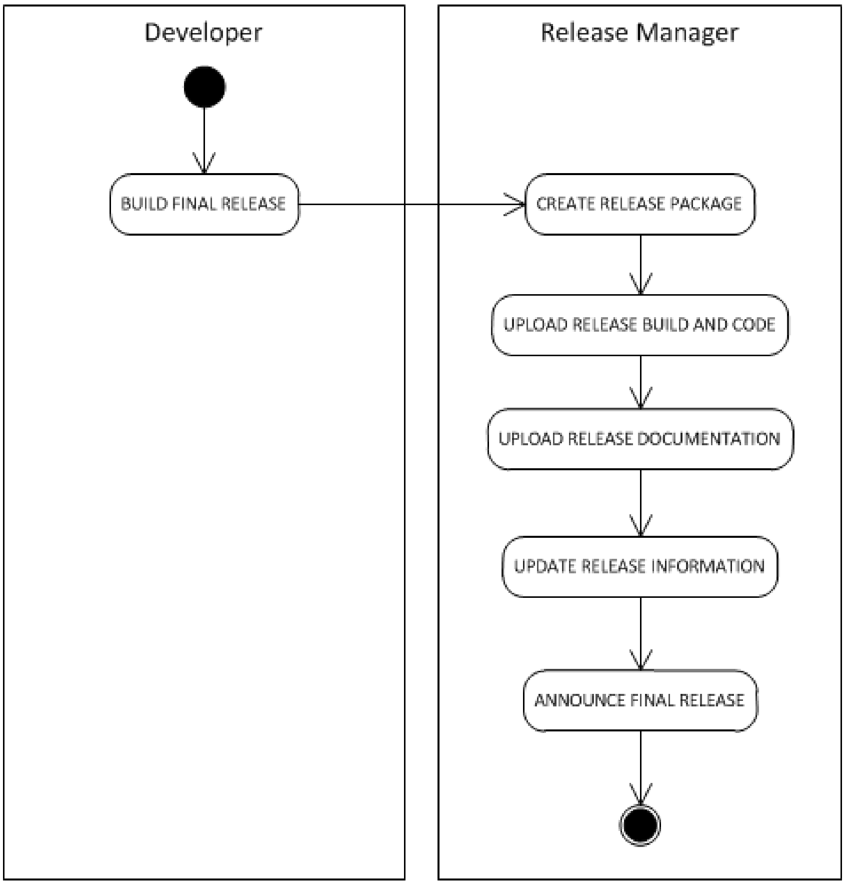
Final release publishing

Upon successful completion of the test and quality assurance plan, including all required actions to certify that the system is fully functioning in accordance with operational and technical parameters, the system will be scheduled for release into production and publishing.

The final release is announced in OSOR environment and emails to users. The announcements should include the relevant information about the changes incorporated, bug fixes and new features, so users can understand easily, why it might be useful to upgrade.

The Release Manager is responsible for preparing the release package and uploading it to OSOR environment.

After the release, Release Management must be informed in a timely way of any comments, complaints, incidents, etc. that the new version may have produced. All this information must be analyzed to ensure that future versions incorporate the suggestions received and that the necessary corrective measures are taken to minimize the negative impact that future changes might have.



Build Final Release: the Developer builds the code for final release using generally accepted numbering rules.
Create Release Package: the Release Manager creates new release package in OSOR environment. Package numbering convention should be same as build numbering.