abat consult Space for the business of tomorrow.

WHITEpaper

Quality assurance for ABAB developments with abat

This white paper provides an overview of the standard inspection tools and their configuration options, presents possible integration scenarios and approval processes using these tools, and discusses advanced QA processes.

AUTHOR

Jens Hollwedel
Senior Consultant

jens.hollwedel@abat.de

Executive summary

When developing custom software, it is helpful to use quality assurance measures and processes to limit the number of tickets during the lifecycle and to enable quick additions or adaptations to changing requirements.

To ensure a rapid return on investment, security, stability, and maintainability must be considered in all phases of the development process. Quality assurance (QA) tools and processes should be integrated into all phases.

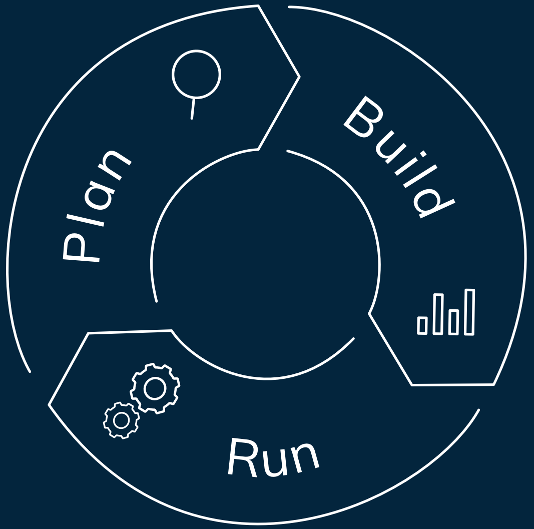
Well-configured testing tools are essential for identifying common problems and monitoring compliance with development guidelines. Processes should be in place to approve critical findings, and developments should be approved through code reviews based on the principle of dual control.

Integrate quality assurance into all phases of the product development process to

- Reduces the number of tickets in the product lifecycle
- Respond more quickly to changing requirements and needs
- Shorten release cycles
- Integrate new technologies more easily.

Plan

- Developer coaching
- Identification of appropriate design patterns
- Visualize the design using UML diagrams
- Test the architecture for security, reusability, and maintainability



Build

- ABAP Test Cockpit (ATC), integration into the Charm process
- Integrate unit tests into ATC
- Development Guidelines and Best Practices
- Adherence to current security guidelines
- Ensuring data integrity
- Use of future-oriented new technologies
- Modern development approaches

Run

- Customize and update ATC test variants
- Accelerate change requests through test automation
- Baseline checks to prepare for further development
- Enhance ATC testing with innovative software metrics
- Monitor software in use with coverage analyzer

Fig. 1 Integrating quality assurance into different phases of the product development process

Preface

In dynamic development projects, constant change is an ongoing challenge. Quality management tools and methods are designed to help control this change and ensure the performance, security, reusability, and maintainability of the software. This means shorter development times, faster testing, less rework in development projects, and a higher return on investment.

Quality management for ABAP development

Sustainable software development is characterized by a holistic approach to quality management throughout the entire product lifecycle. This begins in the conceptual phase, where appropriate design patterns serve as the basis for design and reusable algorithms are conceived as modules. The development process is accompanied by both tool-based code checks and code reviews.

A holistic view of quality assurance in ABAP development encompasses several aspects, from development guidelines and test tools to release processes and manual design and code reviews. These elements are described in detail below.

The quality assurance of ABAP development starts with the concrete solution design and only in exceptional cases with the functional design.

Acceptance of quality assurance measures

Experience has shown that it is very important to involve all project members in the successful use of quality assurance measures in ABAP development. For example, feedback from ABAP developers should be taken into account in the design of the standard test variants used. Business departments, consultants, and testers should also plan adequate time for quality assurance and rework from the beginning.

In addition to appropriate QA processes and test steps, there should be regular coordination between QA and development coordinators.

Development guidelines

Just as the software changes and adapts to new requirements, the development guidelines must also be adapted to the changing state of the art. Ideally, the revision and continuous improvement of the development guidelines should draw on several sources: lessons learned from development projects, and best practices from SAP and DSAG documents.

However, the development guidelines are not intended to replace a developer manual, but rather to provide clear prohibitions and requirements with regard to ABAP development. They should provide specifications for naming Repository objects and cover both design and implementation aspects, with the latter being statically testable as far as possible (although this is not always possible).

New versions of the development guidelines should be presented to the developers before they replace the previous version as the binding guideline. It may also be possible to present the guidelines in workshops to gather feedback and incorporate it as needed.

Test tools

To ensure the quality of ABAP development, SAP provides various tools that are integrated into the system. There are even scenarios for testing older systems using a remote ATC system.

Standard test tools in the system landscape

The testing tools provided by SAP are fully integrated into the development environments, i.e. the ABAP Workbench and Eclipse with the ABAP development tools.

The extended syntax check (SLIN) is the oldest of the tools described here, which is integrated into the newer check tools and therefore does not need to be run separately. Later, the SAP Code Inspector (SCI) was used, which is still used in modern QA scenarios for certain aspects (see Inspection Variants). Based on this, SAP provided the current test tool, the ABAP Test Cockpit (ATC).

For the SCI and, based on it, the ATC, SAP offers various preconfigured check scopes, which can be scheduled in the form of check variants for the execution of SCI or ATC checks. It is also possible to configure your own test variants, consisting of any number of standard tests, which can be parameterized and prioritized according to the rules defined in the development guidelines.

Integration of test tools

All test tools, i.e. the extended syntax check, the SAP Code Inspector and the ABAP Test Cockpit, can be called directly from the development environment, e.g. to check a class that has just been developed or modified.

If the developer does not make a special selection, the standard test variant set in the system is used. However, the developer can also select other validation variants as needed.

In addition to integration with the development tools, regular test runs can also be run as background jobs, allowing special test scopes to be scheduled in addition to standard tests. All results of the standard checks are displayed in the worklist of the responsible developer.

Another integration option is to integrate the checks into the transport release (see Transport Request Release and Approval Workflow) to ensure that all developments have passed this check before they are imported into QSystems and that no unapproved Prio 1 findings remain.

Check variants

The maintenance of test variants is done for both the SAP Code Inspector (SCI) and the ABAP Test Cockpit (ATC) in the SAP interface of the SCI. From a set

of predefined test variants, you can choose which variant to use for which test run. It is also possible to define which variant should be the default test variant. For the SCI, this is entered in the SCICHKV_ALTER[1] table; for the ATC, it can be stored in the ATC settings.

You can also define your own check variants in the SCI. In this case, all checks to be added to this check variant are selected from a set of check categories. The prioritization of the tests can also be customized in three priority levels. Some tests can be controlled by entering additional parameters.

[1] or you name your own test variant

Developing your own ATC tests

In addition to the predefined checks, it is also possible to add your own checks. SAP provides an API for this purpose, so that customer-specific checks can be easily integrated into the check tools as add-ons.

Experience has shown that this is a better way to cover development guidelines that take into account complicated naming conventions and package hierarchies, or to control different conventions for different packages.

In addition, errors or problems that occur frequently[2] can be identified directly with your own checks, which can save a lot of time in manual code reviews, for example.

Introduction of mandatory ATC audits

The introduction of regular and mandatory ATC testing ideally takes place in several steps.

As a first step, a new specific test suite, initially containing only standard tests, should be made available to all developers for voluntary use - this is defined as the standard test suite. In the same phase, feedback from developers should be collected in order to adapt the test-suite accordingly, e.g. to add tests or to adjust the priority of findings. Although this check variant should be included in the transport release[3], it should initially only report Priority 1 findings.

After this introductory phase, regular background checks will be scheduled, but initially without email notification of Prio 1 findings. At the same time, Prio 1 findings should now be considered as preventing the transport during transport release, so that the release (see Release of Transport Requests and Approval Workflow) can only take place after an approval step has been passed.

As a final step, the developers responsible should also be informed by e-mail if Prio 1 findings occur during regular test runs.

At the same time, a test variant should be made available that contains additional test scopes and project-specific add-on tests. This test variant can then be used optionally by the developers and become the new mandatory test variant after an evaluation phase.

abat best practices for ATC testing

- A project-specific test variant should be set up in the system as a standard test variant with adapted prioritizations in order to cover the development guidelines as comprehensively as possible.
- A regular ATC run with the project-specific standard check variant should be performed once a week for the entire system.
- In the case of Priority 1 findings, the system should automatically notify the responsible developers by e-mail after an initial phase.
- After an initial phase, further test variants should be set up for regular test runs, which include the determination of key figures in addition to the tests of the standard variant.
- In a later phase, custom naming convention and package checks, as well as a check for common code review issues, should be added as add-on checks

[2] e.g. a missing extension category for structures or database tables
[3] The SCI test driver should be deactivated, only the ATC should be integrated.

The SCI test driver should be deactivated, only the ATC should be integrated.

In addition to integrating the test tools, especially ATC, into the development environment and performing regular test runs, ATC should also be integrated into the release of SAP Workbench requests in Transport Management.

This can be used, for example, to prevent transport releases if there are still Priority 1 findings in the ABAP code of the associated Workbench objects. Ideally, pragmas or pseudo comments set by the developer are not released at this point without further checks. The individual control of the transport integration of the ATC can be fine-tuned using an appropriate BAdI.

For example, if a transport is to be released with a Prio 1 finding, the system must automatically send a notification to a QA mailbox that is accessible to all quality managers. The developer would justify an approval request in the ATC, which would then be reviewed and possibly approved by a quality manager. The ATC already provides a working environment for approving exceptions and communicating with developers.

However, this quality gate, through which developments must pass by default, should also provide a way to handle emergency transports without delay.

abat Best Practices for ATC transport integration

- Setting up a QA mailbox
- Preventing Transport Releases for Priority 1 Findings
- Ignoring Finding Suppression
- Approval of exceptions during transport release by the quality manager (4-eyes principle)

Solution Manager integration

Solution Manager is often used to control development and manage SAP transports. In such a scenario, in addition to the SAP transports and the results of the ATC checks, the results of the ATC checks can also be displayed and integrated into the SAP transports.

Design and Code Reviews

Beyond the automated checks performed by static code analysis tools, design and code reviews should be established as an integral part of the software development process. In an agile project approach (other project methodologies apply accordingly), the design and code review step could be structured as follows:

In the agile implementation phase, there are two points in each sprint where quality assurance steps make sense: In sprint planning, user stories (i.e., the agile concepts) that meet the „DoR“ (Definition of Ready) criteria are scheduled for implementation. At the end of the sprint, at the Sprint Review, implemented user stories that meet the DoD (Definition of Done) are approved.

For such a story, a technical design should first be created, ideally in the form of UML diagrams. This design is then approved in a design review. Only then does the implementation take place. Tools such as the SAP Code Inspector and the ABAP Test Cockpit, unit tests, and manual code reviews are used to verify that the finished story meets the DoD criteria. The code review step should be performed by a dual review of the source code. The individual quality assurance steps can either be performed by the Scrum teams themselves or, even better, by an independent quality assurance team.

If code reviews are performed by quality managers who are not part of the development team, it should be possible to request reviews through a QA mailbox. The reviewer then reports any findings to the developer and approves the development or implementation. The reviewer documents the status in a project management tool, such as JIRA or the SAP Solution Manager add-on Focused Build.

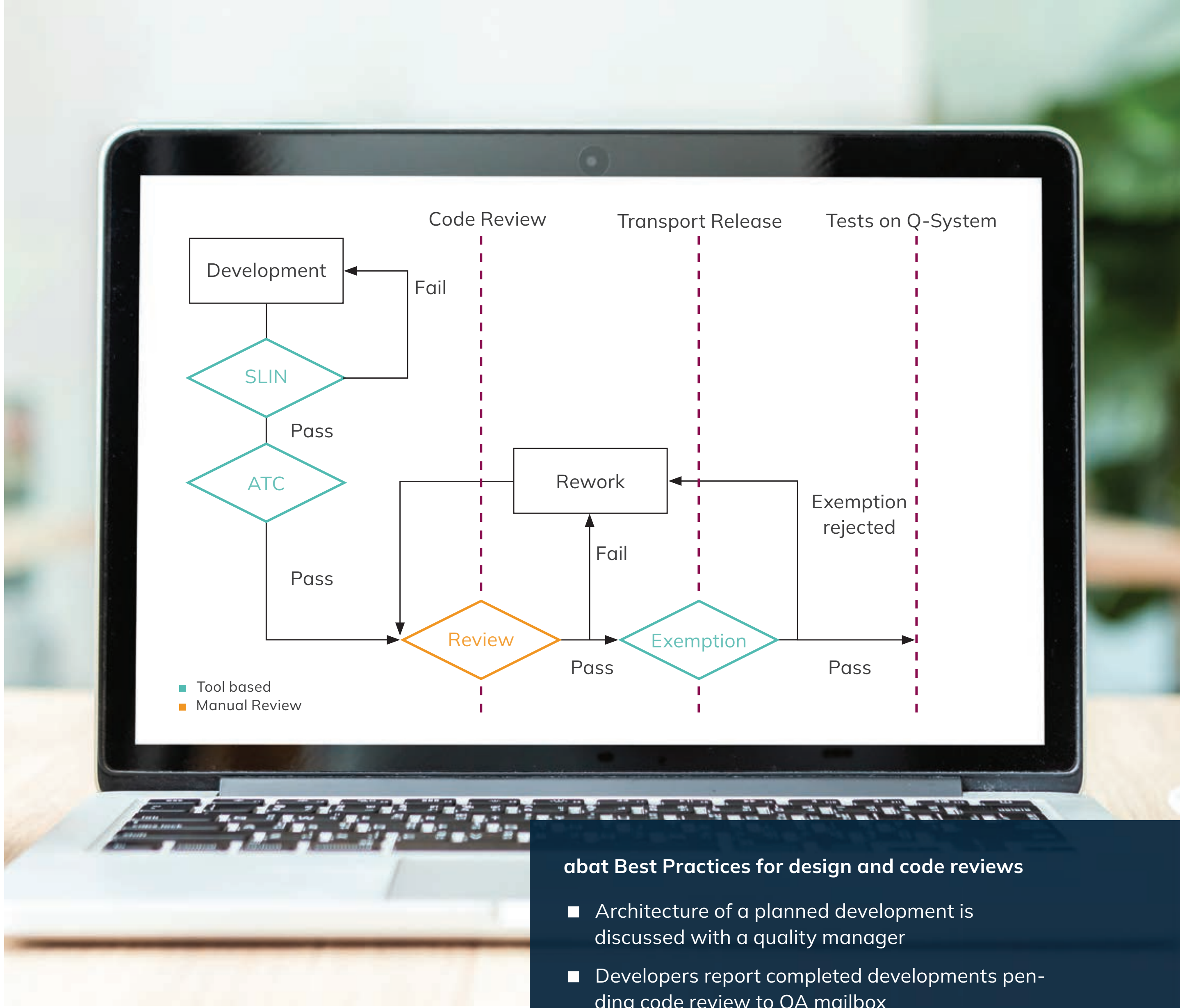


Fig. 2 Quality gates (red dashed lines) during the product development process

abat Best Practices for design and code reviews

- Architecture of a planned development is discussed with a quality manager
- Developers report completed developments pending code review to QA mailbox
- Quality manager sends report with necessary rework to developer
- Status documented in appropriate project management tool
- Regular Jour Fixes are held with lead developers and quality managers

Legacy, technical debt and ATC baseline

In some systems, a large number of ABAP Test Cockpit findings occur over time. These can „mask“ new findings, so that there is no overview of the problems that currently need to be resolved.

Therefore, a baseline for standard tests should be established, in which old defects are included and thus disappear from the relevant set of defects to be resolved. Any issues that remain after a go-live should also be included in this baseline.

To remediate problematic findings in legacy software, test runs should be performed across the entire code base to identify relevant objects.

The ABAP Call Monitor should be enabled on the production system to identify legacy programs that are no longer needed. In addition, the SQL Monitor should be used to log database accesses on the production system over a period of several weeks. The results obtained in this way can be combined with the performance checks of the ABAP Test Cockpit to identify objects that are particularly critical to performance.

CONTACT

Jens Hollwedel
Senior Consultant

jens.hollwedel@abat.de

ABOUT US

The abat Group, founded in 1998, is an SAP service provider, innovative software developer and provider of complete solutions for software-supported process optimization – primarily in the core industries of automotive and discrete manufacturing as well as in logistics processes and production control. With our six service areas, we give companies the freedom they need for new ideas, efficient processes, and future-oriented solutions.

In the **consulting** service area, we advise and support you in all phases of an SAP project – from conception to implementation to operation of your SAP system. With abat **manufacture**, you receive digital, high-availability solutions for production control in the complex manufacturing industry. With abat **transform** we offer you innovative and unique solutions that make you special: from AI to cloud to X-Reality. The **PLM** area offers comprehensive process consulting with the goal of achieving a continuous data flow across PLM, ERP and MES. Offerings from the **protect** area help customers secure information and maintain the confidentiality, availability, and integrity of business relationships. Finally, our **sustain** experts advise on how sustainability and CSR reporting can be strategically and structurally anchored in the company.

abat | An der Reeperbahn 10 | 28217 Bremen | Germany | +49 421 43 04 60 | www.abat.de