SAP Standard for Custom Code Management
SAP Solution Manager 7.2

Document version 1.0 – 2017-03-31
LEGAL DISCLAIMER

The information in this presentation is confidential and proprietary to SAP and may not be disclosed without the permission of SAP. This presentation is not subject to your license agreement or any other service or subscription agreement with SAP. SAP has no obligation to pursue any course of business outlined in this document or any related presentation, or to develop or release any functionality mentioned therein. This document, or any related presentation and SAP’s strategy and possible future developments, products and or platforms directions and functionality are all subject to change and may be changed by SAP at any time for any reason without notice. The information in this document is not a commitment, promise or legal obligation to deliver any material, code or functionality. This document is provided without a warranty of any kind, either express or implied, including but not limited to, the implied warranties of merchantability, fitness for a particular purpose, or non-infringement. This document is for informational purposes and may not be incorporated into a contract. SAP assumes no responsibility for errors or omissions in this document, except if such damages were caused by SAP’s willful misconduct or gross negligence.

All forward-looking statements are subject to various risks and uncertainties that could cause actual results to differ materially from exceptions. Readers are cautioned not to place undue reliance on these forward-looking statements, which speak only as of their dates, and they should not be relied upon in making purchasing decisions.

DOCUMENT HISTORY

<table>
<thead>
<tr>
<th>Version</th>
<th>Date</th>
<th>Change</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.0</td>
<td>2017-03-31</td>
<td>First version created</td>
</tr>
</tbody>
</table>
## TYPOGRAPHIC CONVENTIONS

<table>
<thead>
<tr>
<th>Type Style</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Example</strong></td>
<td>Words or characters quoted from the screen. These include field names, screen titles, pushbuttons labels, menu names, menu paths, and menu options. Textual cross-references to other documents.</td>
</tr>
<tr>
<td><strong>EXAMPLE</strong></td>
<td>Emphasized words or expressions.</td>
</tr>
<tr>
<td><strong>Example</strong></td>
<td>Technical names of system objects. These include report names, program names, transaction codes, table names, and key concepts of a programming language when they are surrounded by body text, for example, SELECT and INCLUDE.</td>
</tr>
<tr>
<td><strong>Example</strong></td>
<td>Output on the screen. This includes file and directory names and their paths, messages, names of variables and parameters, source text, and names of installation, upgrade and database tools.</td>
</tr>
<tr>
<td><strong>Example</strong></td>
<td>Exact user entry. These are words or characters that you enter in the system exactly as they appear in the documentation.</td>
</tr>
<tr>
<td><strong>&lt;Example&gt;</strong></td>
<td>Variable user entry. Angle brackets indicate that you replace these words and characters with appropriate entries to make entries in the system.</td>
</tr>
<tr>
<td><strong>EXAMPLE</strong></td>
<td>Keys on the keyboard, for example, <code>F2</code> or <code>ENTER</code>.</td>
</tr>
</tbody>
</table>
# TABLE OF CONTENTS

<table>
<thead>
<tr>
<th>Section</th>
<th>Title</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>SAP STANDARDS FOR END-TO-END SOLUTION OPERATIONS</td>
<td>5</td>
</tr>
<tr>
<td>1.1</td>
<td>Control Center Approach</td>
<td>7</td>
</tr>
<tr>
<td>2</td>
<td>OVERVIEW OF THE SAP STANDARD FOR CUSTOM CODE MANAGEMENT</td>
<td>10</td>
</tr>
<tr>
<td>2.1</td>
<td>Dimensions of Custom Code Management</td>
<td>11</td>
</tr>
<tr>
<td>2.1.1</td>
<td>Quantity</td>
<td>11</td>
</tr>
<tr>
<td>2.1.2</td>
<td>Usage</td>
<td>11</td>
</tr>
<tr>
<td>2.1.3</td>
<td>Quality</td>
<td>12</td>
</tr>
<tr>
<td>2.1.4</td>
<td>Severity</td>
<td>13</td>
</tr>
<tr>
<td>2.1.5</td>
<td>Criticality</td>
<td>14</td>
</tr>
<tr>
<td>2.2</td>
<td>City Model</td>
<td>14</td>
</tr>
<tr>
<td>2.3</td>
<td>Configuration</td>
<td>15</td>
</tr>
<tr>
<td>3</td>
<td>LIFECYCLE OF CUSTOM CODE MANAGEMENT</td>
<td>17</td>
</tr>
<tr>
<td>3.1</td>
<td>Plan phase</td>
<td>18</td>
</tr>
<tr>
<td>3.1.1</td>
<td>Setting the Scope</td>
<td>18</td>
</tr>
<tr>
<td>3.1.2</td>
<td>Checking Alternatives</td>
<td>18</td>
</tr>
<tr>
<td>3.2</td>
<td>Build phase</td>
<td>19</td>
</tr>
<tr>
<td>3.2.1</td>
<td>Designing Custom Code</td>
<td>19</td>
</tr>
<tr>
<td>3.2.2</td>
<td>Building Custom Code</td>
<td>22</td>
</tr>
<tr>
<td>3.3</td>
<td>Run phase</td>
<td>24</td>
</tr>
<tr>
<td>3.4</td>
<td>Optimize phase</td>
<td>25</td>
</tr>
<tr>
<td>3.4.1</td>
<td>Impact Analysis for Custom Code</td>
<td>25</td>
</tr>
<tr>
<td>3.4.2</td>
<td>Quality Improvement</td>
<td>27</td>
</tr>
<tr>
<td>3.4.3</td>
<td>Custom Code Retirement</td>
<td>28</td>
</tr>
<tr>
<td>4</td>
<td>DRIVING CONTINUOUS IMPROVEMENT</td>
<td>30</td>
</tr>
<tr>
<td>4.1</td>
<td>Quality Assurance Tasks</td>
<td>30</td>
</tr>
<tr>
<td>4.2</td>
<td>Quality Targets and KPIs</td>
<td>31</td>
</tr>
<tr>
<td>5</td>
<td>TRAINING</td>
<td>34</td>
</tr>
</tbody>
</table>
1 SAP STANDARDS FOR END-TO-END SOLUTION OPERATIONS

IT organizations face new challenges every day as they attempt to remain effective and future safe while also keeping costs for day-to-day operations as low as possible. They are also being challenged more than ever to demonstrate their value to the business. Therefore, it is important to optimize the day-to-day tasks that appear to have less obvious business value and to use KPI and benchmark-based reporting to make IT processes more visible, demonstrating the real value that IT can provide.

In order to minimize the costs of IT, it is necessary to standardize and automate the end-to-end IT processes without reducing the SLAs required by the business, such as stability, availability, performance, process and data transparency, data consistency, IT process compliance, and so on.

Based on the experience gained by SAP Digital Business Services (DBS) while serving more than 36,000 customers, SAP has defined process standards and best practices to help customers set up and run end-to-end solution operations for their SAP-centric solutions.

The Build phase of SAP best practices supports a Build SAP Like a Factory approach, consisting of the following processes:
- Custom code management
- Change, test, and release management
- Incident, problem, and request management
- Solution documentation

During the Run phase of a solution, adapting your IT infrastructure to a Run SAP Like a Factory operation impacts both application operations and business process operations. Therefore, operations processes, such as end-to-end root-cause analysis, system monitoring, system administration, and data volume management need to be optimized to achieve state-of-the-art application operations. In business process operations, the same applies to business process and interface monitoring (including performance optimization), data consistency management, and job management.

Quality management processes and tasks need to be established throughout the lifecycle to guarantee continuous improvement of the end-to-end solution operations processes while simultaneously ensuring the flexibility needed to react to changing requirements.
Figure 1 shows an organizational model for solution operations that aligns SAP best practice topics and SAP standards for End-to-End Solution Operations with SAP’s control center approach. The Operations Control Center executes and controls the Run SAP Like a Factory processes, while the Innovation Control Center ensures optimal custom code management and a smooth transition to production with integration validation procedures. SAP connects to these control centers from the Mission Control Center to ensure that professional support is available to the customer. The following Application Lifecycle Management (ALM) functions are not provided directly in one of the control centers because they must be handled across different areas:

- Change, test, and release management
- Incident, problem, and request management
- Solution documentation

The quality management methodologies are an essential part of SAP’s Advanced Customer Center of Expertise (Advanced CCoE) concept and ensure that the KPI-driven processes are continuously improved across all processes and teams. In addition, the quality manager roles ensure consistent and value-centric reporting to the business and management. This unified reporting platform is known as the Single Source of Truth.
1.1 Control Center Approach

The control center approach consists of three components:

- Mission Control Center (MCC)
- Innovation Control Center (ICC)
- Operations Control Center (OCC)

Both the ICC and OCC are made available at your IT facility, while the MCC is located at regional SAP sites. All three approaches are linked together through the SAP Solution Manager application management solution.

![Figure 2: Interaction Between ICC, OCC, and MCC](image)

Mission Control Center (MCC)

The purpose of SAP Mission Control Centers (MCCs) is to support the ICCs and OCCs at customer locations, enabling proactive identification and fast resolution on critical issues operating the SAP solutions and helping to apply standard SAP software functionality that addresses business requirements. The MCCs are serving as the central inbound channels for all complex and business critical request of our customers. MCCs connecting customers to experts from SAP that are ready to provide support across all solution areas and phases of the application lifecycle.

SAP MCCs are located on North America, Latin America, Europe and Asia regions. All MCC’s are networked, use a common infrastructure and service management system, providing 24x7 year around coverage for critical customer situations.
Innovation Control Center (ICC)

SAP’s Innovation Control Center (ICC) is the delivery framework to deliver mid to long term innovation programs. The ICC combines a set of experts, services, tools and templates and represents a lean front office at the customer location that is connected to all offerings of a very strong back office, called the Mission Control Center (MCC). This ensures access to the expertise of the entire SAP ecosystem in a structured way.

The Innovation Framework is the foundation of an ICC and is led by a certified ICC Lead. The Lead delivers against a long-term, value based roadmap, sets-up collaboration tools and dashboards for the connection to the MCC and creates innovation service plans for the underlying projects. ICC services are available for all phase of innovation projects:

- Discover/Prepare: e.g. Prototyping
- Explore/Design: e.g. Gap Validation or Design Review
- Realization/Deploy: e.g. Integration Validation (Safeguarding)
- Run: Transition to Operations

The overall concept of ICC/MCC establish a long-term relationship to SAP and helps saving implementation costs and time for our customers.

Operations Control Center (OCC)

The Operations Control Center (OCC) is the physical manifestation of the Run SAP Like a Factory philosophy. The Operation Control Center (OCC) is a service of an IT organization that:

- creates the relevant transparency to business and other stakeholders along the IT aspects of the seamless execution of E2E critical or core business processes
- provides the relevant transparency on health of the end to end IT landscape and underlying software components
- manages critical exceptions and continuous improvement on the above aspects based on data driven insights
- is supported by standardized IT processes

![Figure 3: SAP Mission Control Centers – Customer Innovation Control Center collaboration model](image-url)
An Operation Control Center is sitting as a layer across / above typical IT departments (who are responsible for the day to day IT operations). It is the job of the OCC to immerse itself in the landscape and processes to fully understand the operational challenges facing the business. Centralized tools and standardized monitoring procedures provide much-needed transparency into these challenges. Meanwhile, a focus on continuous improvement and optimization can improve operations over the long term. As a result, IT departments can realize reduced costs and better capitalize on new opportunities for innovation. To achieve these goals, the OCC relies on a close interaction with both the Innovation Control Center (ICC) and the SAP Mission Control Center (MCC).

The OCC is typically equipped with large screens that display the status of business processes, IT landscape components, as well as exceptions and alerts. If problems occur, a video link can be used to obtain live support from SAP and partners. The customer is responsible for managing the OCC.

The OCC is most effective when closely integrated with other IT processes, such as IT Service Management (ITSM) and Change Management. Central monitors and dashboards based on application and business process operations display the current status of business and IT-related processes. This data can also be used to drive continuous improvement.

An effective system monitoring and alerting infrastructure is fundamental to the success of an OCC and feeding the OCC. The OCC is safeguarding all relevant IT aspects, and the execution of the end to end business processes in scope. The OCC reacts and manages on exception along this critical business processes according to predefined error-resolution activities. The OCC manages follow-up activities for error handling if the relevant tasks are not completed within a certain timeframe.
2 OVERVIEW OF THE SAP STANDARD FOR CUSTOM CODE MANAGEMENT

Custom developments, enhancements, and modifications to SAP software are commonplace at many companies. Implementing custom code enables you to extend the functionality of SAP standard software and adapt it to your company-specific needs. However, custom code does have several disadvantages. For example, over time, custom code is costly to maintain and can cause performance bottlenecks. In addition, due to a lack of understanding of SAP software, companies often create custom code to fill functional gaps even though SAP already provides the desired function in the standard release. A lack of transparency also means that companies often maintain custom code long after it has become obsolete. Poor quality custom code can create countless problems in the productive environment, which drives up the cost of operations. All of these problems can be avoided by implementing sustainable custom code management processes within your organization.

The SAP Standard for Custom Code Management involves constant assessment and optimization of the four dimensions of custom code (quantity and usage, quality, technical severity, technical criticality) as well as preparing and adopting custom developments for daily operations and support. It also provides a holistic view of custom developments from an operational perspective.

Note

The topics and methodology described in this document is relate purely to ABAP.

As long as your IT system landscape and the most important business processes are running smoothly and costs remain within an acceptable range, it can be easy to forget about custom enhancements and developments. However, it is important to ask the following questions about your custom code:

- How many custom code objects are there in the different systems?
- Are all of the objects required or are some of them no longer used?
- What is the quality of the custom code? Was it developed according to defined standards?
- How different is the custom development from the SAP standard?
- How well have customer-specific programs been implemented?
- Will the development continue to run smoothly in the future?
- Who will take care of custom developments when the responsible developers are no longer available?
- What unforeseeable maintenance costs are hidden in the custom development?
- What optimization potential can be realistically attained?
- Will the custom developments still be needed after SAP standard software is updated?
- How will the custom developments be affected by an SAP upgrade?
- How will the custom development be affected by implementing new innovations?
Custom code can improve SAP standard software. Different custom solutions exist, primary developed in ABAP, which interact and integrate with each other. This can be a very complex implementation. As the focus is usually on meeting functional requirements, other important factors, such as documentation, impact on core business processes, quality, operability, supportability, and maintainability often do not receive proper attention. This SAP Standard for Custom Code Management provides advice on how to avoid this situation through sustainable optimization and how to manage important factors from an operational and maintenance perspective. Custom code management can be described as a process of permanent optimization for custom developments based on an "as-is" analysis and preparing for integration into solution operations and support.

The key to successful custom code management is transparency and constant optimization of the quantity and quality of custom developments, their impact on business processes, their technical severity, and their usage. The aim is to achieve optimal operability, maintainability, and supportability of custom developments. This should be considered during the implementation project and all phases of the custom code lifecycle.

In this document, the term "custom code" refers to modifications, enhancements, and pure custom code objects.

### 2.1 Dimensions of Custom Code Management

Custom code that exists in a company’s SAP system landscape can be evaluated in several dimensions. These dimensions are quantity, severity, criticality, quality and usage. The goal is to use these dimensions to establish a constant evaluation process and optimize your custom code based on the results of the analysis. This section describes these dimensions in detail.

#### 2.1.1 Quantity

It is essential to be aware of the quantity of your custom objects. In the ABAP environment, a custom object can be, for example, a program, a customer-specific transaction, or a function. However, it can also be a self-implemented SAP user exit or SAP enhancement interface. In the Java environment, objects can be modified archives of SAP standard scenarios, different types of development components, or portal content.

Each group of objects has its attributes and a technical realization type. It is important to know which objects you have in order to evaluate which are still in use and what custom code is probably obsolete. Reducing the number of obsolete objects can significantly lower the total cost of ownership (TCO). The main tool that can support this optimization in the ABAP environment is Custom Code Lifecycle Management (CCLM) in SAP Solution Manager.

#### 2.1.2 Usage

During the lifecycle of custom code, you have to be aware of your custom object usage statistics. Unused objects still consume resources, which not only increases the TCO, but also increases the security risk to the respective systems. Therefore, it is important to identify objects that are no longer in use so that you can decommission and retire them. In SAP Solution Manager, the Custom Code Lifecycle Management (CCLM) tool collects your custom code objects and uses workload statistics as well as Usage and Procedure Logging (UPL) or ABAP Call Monitor (SCMON) data to help you with such tasks.
2.1.3 Quality

Poor quality custom code often causes unforeseen failures in applications and core business processes. This interrupts business continuity and can become expensive. Companies often only consider functional quality of their custom code, but the non-functional factors are equally as important. The ISO Standard 9126 defines the following major categories for determining the quality of software products:

- Functionality
- Reliability
- Usability
- Efficiency
- Maintainability
- Portability

Each of these categories is further divided into subcategories as follows:

![ISO Standard 9126 Quality Model for External and Internal Quality](image)

Maintaining a clear overview of the quality of your custom code is a prerequisite for driving continuous improvement. Basing your company-specific quality model and quality measurement system on the ISO 9126 standard helps you to maintain this overview. Ideally, you should use a tool to help you measure the quality of your custom code. The ABAP Test Cockpit (ATC), provided within the ABAP development environment, and Code Inspector are tools for conducting formal analyses of ABAP code quality. You need to differentiate between the formal quality of the code, for example, naming conventions or documentation within the code, and the overall product quality represented by the ISO standard. You typically ensure formal code quality by analyzing the source code only. Analyzing the source code is, of course, an important step. However, this type of analysis cannot provide information about every relevant quality category, for example, usability. To perform a more comprehensive analysis of your custom code, you could complement static ATC checks with UPL / SCMON and the SQL Monitor. Companies rely on their business processes running smoothly and correctly. Therefore, custom solutions should provide the same level of quality as SAP standard solutions. In particular, it is important that your IT staff can still provide adequate support for your core business processes. The SAP standard for IT Service Management provides the best practices regarding the incident and problem management.
2.1.4 Severity

There are several ways to add, extend, or adopt the functionality of ABAP-based software delivered by SAP. SAP programs can be enhanced in different ways, e.g. by using user exits or BAdIs. The enhancement and switch framework simplifies and increases the flexibility when making these changes, ensuring a lower TCO in the future. SAP standard software can also be changed by modification. Modifications are used to make bigger changes than you would with enhancements.

⚠️ Caution

Only perform modifications with the help of the modification assistant and when there is no other option.

You can also extend functionality by using completely custom applications, either standalone or with reference to SAP objects. Generally, all ABAP custom objects can be divided in three main categories:

- Enhancements
  - With standard interfaces, such as BAdIs or customer exits
  - Without interfaces, for example, implicit or explicit enhancements directly in the source code
- Modifications
  - With tools that assist modification
  - Without tool assistance
- Custom developments
  - Independent (standalone and composite applications in customer namespaces)
  - With references to SAP objects. SAP strongly recommends that you only reference SAP objects that are released for customers. Failure to do so can cause problems when updating your SAP software.

The goal of the technical severity dimension is to categorize how custom code deviates technically from the SAP standard. Generally, you should avoid custom developments whenever possible and remain as close as possible to the standard.

Clones of SAP code are easily created but once copied to a customer namespace they are no longer maintained with SAP software updates and changes will not be recognized. You should regularly use the Clone Finder tool to identify clones in your system.

Custom code should always provide a clear business benefit. If a lot of custom code already exists, you need to clear unnecessary and problematic custom developments. In general, SAP recommends that you avoid modifications wherever possible. Enhancements with interfaces should be the preferred way of making custom developments. In addition, you can use custom applications (SAP referenced or composite) wherever requirements are more extensive. It is also important to know what impact the technical severity will have on future changes, like upgrades, for existing custom developments.
2.1.5 Criticality

The goal of the criticality dimension is to provide information about the extent to which your technical systems and business processes are affected by custom code. Using the Custom Code Lifecycle Management (CCLM) tool in SAP Solution Manager, you can measure the criticality of an object based on the following data pools:

- Users
- Frequency
- Runtime
- Data Growth
- Table Utilization

2.2 City Model

The 3D City Model was introduced to provide a graphical overview of your current custom code situation in the selected systems.

![Figure 7: 3D City Model in CCLM](image)
The city model is based on 3 + 2 dimensions: the three dimensions of the graph (quantity, criticality, and severity) plus either quality or usage information displayed in the tables. You can use this overview to monitor your custom developments and as a basis for deciding whether further custom code management activities are necessary.

- The **Quantity** dimension shows the number of custom objects in the managed systems.
- The **Criticality** dimension contains five levels, which indicate how important your custom developments are to your business. This criticality rating is defined by the available pools within the Business Criticality Settings in the Custom Code Management scenario of SAP Solution Manager Configuration.
- The **Severity** dimension indicates how far the objects differ from the SAP standard. The severity levels are Enhancements, Modifications, or independently developed Custom Code.
- The **Quality** dimension displays the total number of objects with each quality rating. This rating is based on the quality criteria in the managed systems as defined in the ABAP Test Cockpit (ATC). You need to set them up as part of the Custom Code Management scenario of SAP Solution Manager Configuration for the managed system(s).
- The **Usage** dimension displays the total number of used and unused objects based on workload statistics and Usage & Procedure Logging (UPL) or ABAP Call Monitor (SCMON).

You can use the city model to make effective decisions and improve the governance model. Your metrics do not have to describe the whole SAP IT landscape of the company; you can even improve effectiveness by applying these practices to smaller systems.

### 2.3 Configuration

After completing the mandatory setup activities in SAP Solution Manager 7.2,

- **System Preparation**
  Checks the current installation and setup of the SAP Solution Manager system, for example, license data, implementation of necessary SAP Notes, and creates users
- **Infrastructure Preparation**
  Set up the synchronization of the system landscape directories (SLDs) and SAP Solution Manager.
- **Managed System Configuration**
  Connects the systems that you want to manage and monitor to SAP Solution Manager.

Additional Custom Code Management configuration is also required and needs to be completed by executing the option “Configuration” through SAP Fiori Launchpad in the area Custom Code Management.
The configuration will cover the setup of the BW infrastructure as well as the necessary extractors and collectors by choosing the relevant custom code scenario via the data collection group.

These setup activities could gather data about the following topics related to Custom Code Management:

- Custom code objects, modifications, enhancements
- Usage information from the managed system
- Similarity to SAP standard (clones)
- Referenced SAP objects
- Criticality Information
- Quality data for custom code from ATC runs
3 LIFECYCLE OF CUSTOM CODE MANAGEMENT

You need to plan the different phases of the custom code lifecycle carefully because each phase influences the others. It is important to optimize each phase so that they all work together. You need to use the strategy developed during the planning phase to collect business requirements from all stakeholders and then ensure business continuity throughout each phase of building, testing, deploying, and usage of your custom code.

The Custom Code Lifecycle Management (CCLM) capability provides transparency regarding your custom code and records how it is used in a complex landscape, which helps you to manage your custom code effectively throughout its entire lifecycle.
3.1 Plan phase

You should only create custom code when there is no solution provided by SAP standard software. When creating custom code, your end goal should always be to create a competitive advantage. The decision to make or buy new software should be taken with great care. You need to consider all alternatives, for example SAP standard functions, business process redesign, and reducing the technical severity of the custom implementation during this planning phase.

3.1.1 Setting the Scope

Custom developments should be seen as an investment and an opportunity to adopt customer systems by implementing business requirements and closing functional gaps. There is no single method for integrating business requirements into the custom code lifecycle.

Custom code must create value every time it is used. It should be seen as an asset and, therefore, it must generate a return in a short period of time. Every unused piece of custom code wastes time, money, and resources. For handling multiple large projects, you have to implement Portfolio Management. Portfolio Management handles high-level project management tasks like planning, controlling, resourcing, and reporting for multiple development projects. Strategic multi-project management aims to identify business and IT requirements for new development projects, prioritize them with the enterprise according to business cases, prioritize them with IT services according to business benefits, and manage risks.

3.1.2 Checking Alternatives

SAP recommends checking whether you can use SAP standard functionality or enhancements instead of creating new custom code. SAP provides several tools to help you find products and innovations, such as the Innovation Discovery tool, which simplifies your search for new functions and allows you to identify available functionality from SAP.

Innovation Discovery for SAP Products

SAP’s Business Suite innovations offer several improvements and enhancements. The Innovation Discovery tool for SAP Products is a self-service for customers, which helps them to find innovations for SAP products. For more information, go to http://support.sap.com/innovationdiscovery.

Accelerated Innovation Enablement

The Accelerated Innovation Enablement (AIE) service provides access to SAP solution architects to help you evaluate the innovation capabilities of the latest SAP enhancement packages and how they could fulfill your business process requirements. For more information, see https://support.sap.com/support-programs-services/programs/enterprise-support/academy/delivery/accelerated-innovation-enablement.html.

Innovation Control Center

During the execution of implementation projects for SAP engagement customers, an Innovation Control Center (ICC) focuses on designing and creating solutions as close to the standard as possible. The Innovation Control Center is responsible for managing any deviations from standard SAP software. SAP
experts, both on site and at SAP, work together to show how solutions to perceived functional gaps can be implemented. An innovation control center uses SAP Solution Manager to evaluate functional gaps and monitor project status. For more information, see http://www.sapsupport.info/support-innovations/control-centers/icc/.

SAP Modification Justification Check

Modifications to SAP applications can lead to a significant increase in the total cost of ownership (TCO) when running your SAP solutions. They usually require considerable investment during the implementation phase and continue to generate additional costs when you update your SAP software. The objective of the SAP Modification Justification Check (MJC) is to provide expert advice on how to avoid SAP source-code modifications by using SAP standard functionality whenever possible or by taking advantage of the SAP Enhancement Framework. For more information, see SAP Note 1281532.

Custom Code Documentation (Functional Specification)

Functional specification documentation is the set of documents that describes the behavior of a program, which is necessary to fulfil certain requirements. As documenting custom developments is the only way to connect the technical objects of the implementation with its functional requirements, the documentation needs to reflect the complete development lifecycle from initial functional request to the end-of-life decision. The development is considered to have ended whenever the function is replaced by standard functions or is no longer needed from a functional perspective. Proper custom code documentation is essential. You need to check the level and quality of your custom code documentation continually in order to correct any gaps as quickly as possible.

3.2 Build phase

Application configuration and custom code need to be documented according to best practices and custom code guidelines. Business process owners should always provide test cases to accompany new code. You can save time and money by automating the process of testing. For more information, see the SAP Standard for Test Management.

3.2.1 Designing Custom Code

It is important to keep the technical severity in mind before starting to design your custom code. The following list summarizes each design approach and in order of preference:

1. Try and use SAP standard functionality to fulfill all requirements. Use the Customizing possibilities provided as much as possible.
2. If the SAP standard is not sufficient, consider possible workaround solutions, for example, adapting your business processes.
3. If you cannot avoid making enhancements to SAP software, use interfaces, such as BAdIs or customer exits.
4. Make implicit enhancements in the appropriate places within SAP objects. You use this method to deal with conflict adjustments (transaction SPAU_ENH). Before implementation, it is important to carefully examine the following issues:
   o Versioning: how often has SAP changed the code?
   o Complexity: the shorter the coding is, the more stable the implementation is.
   o Preferred alternative: post enhancements should be preferred
5. Make explicit enhancement enhancements by creating your own enhancement points.

6. If there is no other option than overwriting the SAP source code, create explicit enhancement sections or implicit overwrite enhancements. This option is preferable to classic modifications because you can develop the enhancements in custom packages and they can be transported separately on a different transport layer from the SAP standard.

⚠️ Caution

If you create completely new solutions, SAP cannot provide support or maintenance; you will have to perform these tasks yourself. SAP recommends only creating 100% custom solutions if you are an experience software vendor.

7. As a last resort, in exceptional cases, you can make classic modifications.

Generally, creating good custom code follows the official ABAP Programming Guidelines available at http://help.sap.com/abapdocu_740/en/. These guidelines should be used as the basis for all developments.

Development Guidelines

Development guidelines are typically provided by the customer development organization, which is responsible for creation, maintenance, executions, and publication of the guidelines. Development guidelines and rules cover the complete development process from functional requirement approval, technical evaluation, through to technical implementation and functional approval.

SAP provides more information regarding these topics as part of the Custom Development Guidelines. For more information, see http://scn.sap.com/docs/DOC-56285.

Development guidelines address the requirements for ensuring the maintainability, readability, and stability of coding and define the basis for all custom developments. Development guidelines vary greatly from customer to customer.

In general, guidelines need to address the following areas:

- The necessity for common sense in custom development and commitment on the part of the individual developer to observe the defined rules and regulations.
- Description of the implemented development process, its roles and responsibilities, its approval steps, and its quality assurance and documentation processes.
- References to the guidelines for custom development documentation, as well as detailed descriptions and documents.
- Evaluations of the different possibilities for functional enhancements to the standard and their impact on the solution lifecycle. This should be done as part of the functional requirement phase, which precedes the development implementation phase.
- Guidance for deciding how to implement the functional requirement. For example, customizing only, standalone custom applications, and so on.
- List the enhancement technologies used in existing custom developments, for example, user exit form routines, customer functions (CMOD), business transaction events (BTE), business add-ins (BAdI), or enhancement frameworks. You must state the preferred enhancement technology and the customer policy concerning modifications to standard coding.
- State the programming language to be used for custom development and the language to be used for documentation (including inline documentation in the coding). Take notice of the language requirements of SAP support. In object-oriented environments, the language used for modeling also needs to be defined.
- References to accessing functional and technical requirement documents and their approval processes.
• Outline customer namespaces and their use in custom development (global reserved or strictly defined customer namespace). Provide naming conventions for custom development objects based on the SAP standard recommendations.

• Define and provide programming naming conventions for definitions and declarations inside custom code, such as internal tables or global variables. Some examples for ABAP-development can be found using the Code Inspector check variant for programming guidelines.

• Outline the architecture of the development landscape and the software logistic regulations, including naming conventions for transports.

• Name tools and procedures to be used during modeling and implementation in order to achieve readability, maintainability, stability, and performance, for example, UML, pretty print functionality, inline documentation, runtime analysis, Code Inspector, and ATC. Include documentation on how to use the different tools and at what stage of the implementation process.

• Reference common sources of information about development, for example, the SAP Developer Network, SAP Service Marketplace, SAP Help Portal, and available internal sources.

• Examples for proper use of development language statements and provide tips and tricks. For example, see the Tips & Tricks link in SAP Transaction Runtime Analysis (transaction SE30). Include examples for performance optimized or maintainable programming.

• Provide templates to be used during the development process, for example, development request forms, technical implementation evaluation forms, and acceptance protocols.

Documentation (Technical Design)

Custom developments can be created by customers themselves, by third-party companies, or by SAP Custom Development. Supporting customer developments is often more difficult than supporting standard applications. For SAP applications, documentation is already provided by SAP. For functionality developed by customers or third-party software providers, documentation cannot always be ensured. Technical design documentation provides the necessary details about specific requirements. The risk of losing support experience is high when the people involved in the development project leave the customer or the involved third-party company does not exist anymore. Solution Documentation in SAP Solution Manager allows you to identify, add, or implement new or extended business processes and technical information.

Solution Documentation grows continually and is structured clearly and comprehensibly in SAP Solution Manager. For more information about Solution Documentation, see the SAP Standard for Process Management.

Proper documentation of custom developments should provide answers to the following questions:

• Which parts of the implementation project are impacted by the custom development?

• Why has the custom development been done and what are the business requirements?

• Who is the business owner of the development and who participated in the decision process for the development?

• Which systems are involved?

• Are there dependencies between the custom development and other systems or functions?

• Which business processes and process steps have changed?

• What are the transactions or entry points of the custom development?

• Which interfaces are used to integrate custom development with standard applications?

• Which development objects have to be transferred from development to quality assurance and production system?

• How do I operate the custom development in the future?
• How do I set up and execute integration, volume and regression tests for the custom development?
• How do I ensure handover from the project to the production?

The custom development process needs to ensure that no custom development exists without sufficient documentation, for example, by implementing appropriate quality gates. Proper documentation helps both customer and SAP support organizations to deliver adequate support for custom developments.

Implementing New Developments

To manage all changes across the relevant development systems, you can use Change Control Management functions. Change Control Management and Documentation Management are closely integrated. For more information about managing the implementation of custom code from the perspective of Change Control Management, see the SAP Standard for Change Control Management.

Landscape Architecture

Proper landscape and release management is required to manage the implementation of business requirements in major releases combined with SAP updates, minor releases, and standard or urgent changes. Each customer requires a different landscape because the frequency and amount of custom developments and change management KPIs vary.

The SAP whitepaper Two Value Releases per Year under http://sapsupport.info/support-innovations/sapsolutionmanager/two-value-releases-per-year/ outlines how to design and improve software change management and test processes, how to use the transport landscape, and how SAP Solution Manager can help you automate tasks and control processes.

3.2.2 Building Custom Code

In this phase, you need to introduce quality checks, best supported by quality gates. After creating custom code, you have to perform technical tests and functional tests to ensure stable business processes.

Using Custom Code Lifecycle Management

Custom Code Lifecycle Management (CCLM) was developed specifically to accompany your ABAP enhancements and new developments throughout their whole lifecycle. This cycle begins when you create an object, such as a program, transaction or function. The objective of Custom Code Management is to improve custom code’s quality and technical implementation while reducing its quantity and impact. SAP Solution Manager helps you achieve sustainable cost reductions for the operation and maintenance of your SAP system landscape with CCLM. The application is based on a generic library model were you can classify and manage custom code objects developed by your organization. The application provides instant transparency for your custom code and records its use without any additional manual effort. The generic central library is used by SAP as the central data source for all information on customer objects. Key benefits are the ability to create individual assignments of responsibilities and contracts, consolidation of developments within an organization, and total control over new developments. It is possible to assign any object or list of objects to a contract or other predefined attributes.
Custom Code Quality

You use the SAP Code Inspector (SCI) or the ABAP Test Cockpit (ATC) to test single objects or object sets of programs, function groups, classes, interfaces, and dictionary objects for performance, security, serviceability, error proneness, and statistical information. You can ensure the quality of your third-party developments by using the SCI and ATC to agree on quality standards. Use the ATC to check and track the quality of custom code and for unit testing, as well as for periodic static mass tests. Before the tests can be performed, you must first establish your quality goal, define the object set you want to check, and specify the checks to be applied. You can also specify the weighting of the possible messages during the check according to the seriousness of the error. You manage the code and responsibilities of your custom code by assigning contract and owner to CCLM objects. A Q-gate for this check must ensure that the defined quality goals are met. In addition to contractually aligning the developer's organization and customer IT services, SAP recommends that you define the following:

- Quality governance process with quality gates
- Central responsible quality manager
- Approval check document

Testing Custom Code

Testing is a substantial part of the program development process. The ABAP environment provides a rich suite of testing tools that you can use to verify the formal and functional correctness of your programs. Sophisticated static program checks should be part of each ABAP development process. The ABAP Unit Testing tool provides support when you are writing, running, and organizing unit tests in ABAP. You can test even more intensively with the Code Inspector or the ABAP Test Cockpit (ATC). Solution Documentation in SAP Solution Manager can also serve as the central platform for all test-related documents. For more information relating to testing best practices, see the SAP standard for Test Management.

Using Usage and Procedure Logging

Custom code does not always add an executable program to your implementation. In most cases, it implements minor improvements, such as function modules, methods or modifications. This means that classic measurement techniques are unlikely to be helpful. Usage and Procedure Logging (UPL)/SCMON enables you to log usage of executables like Methods and Functions on a daily basis and provides information to help you assess whether your custom development has been worth the cost of development. Evaluating actual usage after the custom code has been requested enables you to distinguish between good and bad investments and will help you to focus on the right requirements during upcoming custom code development cycles. UPL is a central usage data source used in SAP Custom Code Lifecycle Management (CCLM). Without any additional effort, as soon as UPL/SCMON based functionality is available in the managed system, CCLM will begin to monitor usage information.

Incident Management

For the lifecycle of custom code, you need to develop a concept for logging messages from custom user interfaces. This means customers need to be able to create messages directly from the user interfaces, attach technical documentation, and transfer messages to SAP Solution Manager. SAP delivers templates to be included in custom code when developing in SAP environment. For more information, see the SAP Standard for IT Service Management.
3.3 Run phase

Customer systems usually contain a large number of custom development objects. Transparency about their usage must be available so that you can monitor custom code performance. Avoid testing unused custom code within a lifecycle change event and keep the custom code operational. During the Run phase, transparency about usage and quality of custom code is important and you need to be able to determine the impact of custom code within different lifecycle change events. Additionally, the Custom Code Management analytics dashboard enables the end users to have a detailed visibility to the custom code utilization within their SAP Landscapes through Business Value and Key Performance Indicators perspective.

![Custom Code Dashboard](image1)

**Figure 1: Custom Code Dashboard**

![Business Value and Key Performance Indicators](image2)

**Figure 2: Business Value and Key Performance Indicators**

To use custom solutions, you need to cover at least the following areas:

- Incident Management
- Root Cause Analysis
- System Administration and Monitoring
- Change Management
Roles and rules have to be set up in accordance with the governance model to fulfill defined quality goals.

### 3.4 Optimize phase

The ABAP Test Cockpit is an ABAP check toolset which allows you to run static checks and unit tests for your ABAP programs. A uniform quality measurement system has to be defined and established based on quality KPIs. You need to automate the measurement process as much as possible and visualize the results. You have to continually measure and review the quality of your custom code according to the ISO 9126. This process should be run in close cooperation and continuous communication with development teams. You can use tools to combine different information in order to determine areas for improvement. For example, you can use SQL Monitoring (SQLM) to measure the performance and identify the executed program, the ABAP Test Cockpit to display the erroneous line of coding, and UPL / SCMON to provide information about the number of executions. By using these capabilities, you can combine information about performance, usage, and quality to gain insight into your custom code.

#### 3.4.1 Impact Analysis for Custom Code

If you plan to upgrade to a higher SAP release or to apply a support package, you must check and adapt every custom development object manually. You must also test each object to ensure that it does not cause problems in the new context. This process involves a considerable amount of time and effort. SAP offers different tools to support you and help you performing an impact analysis during the planned changes.

By optimizing the application and turning optimizations back into requirements, SAP customers can gain a broader overview of the way the software serves business requirements. This facilitates business innovation.

Determining Test Effort Using the Scope and Effort Analyzer

The Scope and Effort Analyzer (SEA) allows you to analyze the impact of SAP software changes caused by support packages (SP) or SAP enhancement package (EHP) deployments before the physical installation is done. A structured report presents the findings regarding testing effort and custom code adjustments. The tool determines the scope and effort for testing during the regression test, as well as necessary code adaptations for the current EHP or SP maintenance projects based on the object list provided by the maintenance planner. For more information on the Scope and Effort Analyzer, see https://wiki.scn.sap.com/wiki/display/SM/SAP+Solution+Manager+WIKI+-+Test+Suite.

Custom Code Development Cockpit and Upgrade Change Impact Analysis

You could use the SAP Custom Development Management Cockpit (CDMC) to identify potential consequences of an upgrade on custom developments as soon as you have an updated physical installation in place. Before you can upgrade a system to a newer SAP version or import a support package, all custom code must be checked and manually adjusted. Using the CDMC, you can determine
how custom code is used based on the call statistics provided by the system and UPL, and which SAP objects are referenced by custom developments. CDMC then evaluates the potential adjustments that need to be made during custom development because SAP has changed the referred objects as part of the lifecycle change event. The result list enables a remote comparison between the "as is" and "to be" state of the analyzed objects that you have to adjust.

Preventing Transport Errors before Go Live with new Custom Code

To ensure the security of custom development during go-live, SAP recommends analyzing transport requests before they are imported into the productive system for potential import and sequence errors. For this, you could use the Guided Self Services in SAP Solution Manager. It also provides the necessary info how these errors can be avoided. As a result the import gets more robust and the stability of the production system is improved. For more information, see the Guided Self Services documentation about the corresponding Transport Execution Service for Projects (TEAP) at https://support.sap.com/support-programs-services/offerings/enterprise-support/academy/delivery/guided-self-services.html.

Migration of Custom Code to SAP S/4/HANA

For the transition to SAP S/4HANA, custom code needs to be examined in detail; SAP has changed data structures and business processes in SAP S/4HANA, and these changes may prevent custom code from running correctly on SAP S/4HANA. With the SAP HANA platform, you can meet high performance and throughput requirements. You should consider performance, scalability, and operational readiness before you start to migrate your custom developments. SAP Solution Manager 7.2 delivers Custom Code Management, which helps to tackle potential problems caused by custom code before the transition to SAP S/4HANA in an efficient manner.

Use CCLM in SAP Solution Manager 7.2 for a stable and solid starting point as a single source of truth for your custom code to capture your custom code footprint based on a city model. Then, in the Decommissioning Cockpit, you can identify and remove unused custom code. The quality cockpit allows you to evaluate your custom code in the SAP S/4HANA context, based on new Code inspector checks evaluating your custom code to be ready for simplification and HANA. Implementing and operating SAP S/4HANA also is supported by well-known functions in SAP Solution Manager such as roadmaps. Roadmaps explain how customers can move from their existing set-ups to the new solutions, by using generic methodologies provided by SAP Activate or product-specific methodologies. The roadmaps also provide an introduction to the architecture of the new products as well as access to best practice experience on how to operate SAP HANA based solutions.

The SAP Solution Manager is able to carry out checks regarding system and landscape requirements for a successful SAPS/4HANA transition. The following link http://jam4.sapjam.com/groups/about_page/elGzElriaSFuwNCeuSPr5q to "SAP S/4HANA Value Assurance Services" shows more details regarding roadmaps, best practices and available services.
3.4.2 Quality Improvement

Experience shows that more than 60% of custom code objects contain code quality issues. To define a quality strategy, it is necessary to achieve transparency about the existing custom code objects and their quality in your system landscape. Additionally, to avoid issues after migration to an SAP S/4HANA system, we recommend that you analyze and fix your custom code proactively—before the migration takes place. For this, new HANA and S/4 checks are being integrated into ATC. A successful quality management project is divided into four main steps:

- Analyze your systems to obtain 'as is' quality information.
- Consolidate the results of the different checks.
- Adjust the custom code according to your check results.
- Monitor the progress of software quality improvements for relevant objects.

The main tools that support this project phases are the ABAP Test Cockpit (ATC) running in the managed system, and the new Quality Cockpit integrated into CCLM in SAP Solution Manager 7.2. The Quality Cockpit provides detailed information about the quality of your custom code objects and serve as basis for your custom specific use cases for improvement projects. You could set up your own quality projects according your needs to monitor and improve the quality of your custom code. The results are quickly presented as seen below:

![Figure 12: Quality Cockpit](image-url)
3.4.3 Custom Code Retirement

Unused custom code can have a severe impact on your system by causing an increase in the following undesired effects:

- Maintenance and adjustment effort during change events, for example, system upgrades, new support package implementations, or system merges
- Maintenance effort and cost during operation, for example, bug fixing, dependency of coding, or coupling complex custom code with standard code
- Security and destabilization risk (no control), which can lead to missing authorization concepts
- Testing effort due to unknown usage behavior, which can lead to initial test plans not representing actual business processes
- Data consistency risk when executing unknown and old code, for example, legacy migration reports or Excel upload reports
- Training effort to build and maintain development and support skills, which means that you have to develop procedures for handing over development responsibility and corresponding training activities
- Complexity and roadblocks preventing a more simplified landscape or new technology landscape, for example, Cloud or SAP HANA.
- Unnecessary documentation
- Inaccurate forecast and estimation of future development activities based on uncertainty or incorrectness
- Legal aspects, for example, providing the ability to audit and certifications for processes such as the Payment Card Industry Data Security Standard (PCI DSS)

The majority of the custom code created remains in the system, even if it is not used currently or will not be used in the future. In order to get rid of unused or obsolete custom code, a well-defined decommissioning strategy must be in place. To define a decommissioning strategy, you need transparency of existing custom code objects, including their usage.

Transparency of Custom Code

You can increase transparency across your custom code footprint by building an accurate repository of custom code objects across your entire SAP landscape. You can then use this central repository to clearly identify for example custom code usage.

The goal of transparency is to provide a sustainable process for driving improved planning and custom code object control at certain times during their lifecycle. A more comprehensive insight helps you to set appropriate management priorities and make informed decisions when necessary. SAP Solution Manager provides many functions to meet these needs. For example, Solution Manager 7.2 allows customers without any solution documentation to populate libraries based on system usage. In SAP Solution Manager 7.2 the concept of reusable libraries is introduced. All technical elements such as SAP transactions, configuration units, or interfaces that are used multiple times in customer’s processes are organized in containers, from which they can be pulled into the process steps wherever needed. For more information, please go to E2E Standard for Process Management. Custom Code Lifecycle Management (CCLM) collects and describes attributes and key figures belonging to your custom code solution and helps to achieve transparency for your custom code objects (including modifications). Within the lifecycle of custom solutions they can either be permanently removed from productive use or can be modified and adjusted in order to meet new requirements.
Decommissioning of Custom Code

Obsolete custom code can be retired after careful analysis. The Decommissioning Cockpit, available from SAP Solution Manager 7.1 SP12, helps you identifying redundant or obsolete custom code. Within CCLM, you create and control your decommissioning projects based on usage data and the input from your business department about custom code objects. In order to evaluate usage data directly from your managed system, make sure that both workload statistics and UPL or SCMON data is available and activated for your system.

Modifications also are analyzed within CCLM but due to their technical and functional categories, e.g. inactive Modification due to a new support package implementation, it has to be analyzed separately. Modification Analysis is described in a best practice document available under http://scn.sap.com/docs/DOC-70271. The purpose of this document is to provide guidance to the customers by reducing the scope of technical modifications in a customer landscape and to help customers make the transition to newer technologies.
4 DRIVING CONTINUOUS IMPROVEMENT

To measure the success of Custom Code Management, you can check the efficiency of your support processes by asking the following questions:

- How much custom code do you have in your system?
- How and how often do you execute clearing analysis of your custom code?
- Are all projects at your company fully compliant with your development standard? How do you ensure this?
- What is your definition for accepting modifications and how do you review it?
- How often do you review documentation of your custom code?

You can check the success of your implementation by installing a set of appropriate KPIs and measuring them before and after the implementation. The Interactive Continuous Improvement (ICI) dashboard is built in to the CCM work center. You can set CCM-related objectives and measure your progress toward these goals using customizable scorecards and KPIs.

The following KPIs are available in the CCM dashboards:

- Percentage of criticality level 4 and 5
- Total number of custom code objects
- Total number of low quality objects
- Total number of severity level 1 objects—enhancements
- Total number of level 3 objects—modifications
- Total number of severity level 5 objects—customer objects
- Total number of not used objects

4.1 Quality Assurance Tasks

From a quality management perspective, the key tasks are as follows:

- Ensure you have sufficient custom code documentation, such as policies and change impact analyses, to assess custom code and establish custom code guidelines and custom code assessments
- Set up a regular custom code review to identify unused objects
- Reduce custom code objects and establish custom code clearing and best practices approach, for example, rapid deployment and ICC.
- Define KPIs to track execution of Q-Gates with PMO and the application team and establish areas for improvement
4.2 Quality Targets and KPIs

To efficiently ensure stability and scalability of solutions as well as speed and performance of business processes, the most important quality targets are as follows:

- Improve scalability of IT solutions by ensuring transparency of custom code and make improvements by replacing custom code with standard code to align to business needs
- Increase stability and performance of business processes by ensuring custom code is developed to defined standards and regularly checked by regularly assessing the need for operations custom code and addressing issues related to custom code management
- Increase efficiency and reduce operational costs by reducing maintenance efforts related to custom code and reducing additional testing effort

The following table describes the main challenges for each of these quality targets and which KPIs can be used to measure the success of your implementation:

<table>
<thead>
<tr>
<th>Quality Targets</th>
<th>Challenges</th>
<th>KPIs</th>
</tr>
</thead>
</table>
| Improve scalability of IT solutions by ensuring transparency of custom code and make improvements by replacing custom code with standard code to align to business needs | • Ensuring visibility and governance of custom code  
• Identifying redundant custom code that can be replaced by standard code  
• Proactively avoiding creating custom code that corresponds to functionality.  
• Reactively reviewing custom code and proactively avoiding custom code to increase transparency, reduce maintenance effort and improve the stability and performance of your custom code  
• Increasing compliance with the custom code strategy by building a baseline for transparency and activities related to custom code management in complex and multiple source environments | • Percentage of development projects reviewed for compliance with development standard compliance  
• Percentage of custom code reduction applying the “back-to-standard” approach  
• Ratio of custom code created and thoroughly tested before implementation in the production environment compared with all custom code implemented in the production system |
<p>| Increase stability and performance of business processes by ensuring custom code is developed to defined standards and regularly checked by regularly assessing the need for operations custom code | • Aligning different IT teams to comply with custom code development standards | • Percentage of performance improvement of critical business processes after Custom Code Management implementation |</p>
<table>
<thead>
<tr>
<th>Quality Targets</th>
<th>Challenges</th>
<th>KPIs</th>
</tr>
</thead>
</table>
| and addressing issues related to custom code management | • Checking custom code and removing unused custom code  
• Reviewing modifications to revert code "back to standard"  
• For this, you need to use a single source of truth as a starting point for analysis and improvement  
• Ensuring transparency of custom code by using support functionalities in SAP Solution Manager.  
• This serves as starting point and allows you to get an overview of usage, utilization, and any impact on related business processes  
• Providing results and overviews of custom code objects in terms of severity, quantity, quality, and criticality and identifying and implementing measures, based on analysis  
• Reducing unused custom code, improving the quality of custom code establishing and assessing compliance with development standards.  
• This requires close cooperation between technical teams. Cooperation is particularly important when onsite and offsite teams are working together.  
• Regularly reviewing thresholds to ensure that custom code is always appropriate and free from redundant code  
• Improving business processes by avoiding poorly developed, time-consuming code thanks to close | • Trend in stability after Custom Code Management implementation  
• Trend in ratio of Incidents related to custom code compared with all incidents  
• Percentage of custom code checked for compliance  
• Percentage of custom code cleansing after identification |
<table>
<thead>
<tr>
<th>Quality Targets</th>
<th>Challenges</th>
<th>KPIs</th>
</tr>
</thead>
<tbody>
<tr>
<td>cooperation and discussion with the business</td>
<td>• Improving security by eliminating obsolete objects that could compromise data if executed</td>
<td></td>
</tr>
<tr>
<td>• Improving security by eliminating obsolete objects that could compromise data if executed</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Increase efficiency and reduce operational costs by reducing maintenance efforts related to custom code and reducing additional testing effort</td>
<td>• Increasing cost efficiency.</td>
<td>• Trend in test effort for business users after Custom Code Management implementation</td>
</tr>
<tr>
<td></td>
<td>• Reducing the complexity of your SAP environment</td>
<td>• Cost savings due to reduction of custom code (custom code objects multiplied by time needed to maintain per year, usually 0.5PD per object)</td>
</tr>
<tr>
<td></td>
<td>• Increased complexity extends upgrade time, increases maintenance effort by creating additional work for the development team and additional test effort by business users, and increases the time it takes for changes to be made because of the increased manual</td>
<td>• Overall time needed for upgrade before and after Custom Code Management implementation</td>
</tr>
<tr>
<td></td>
<td>• Appropriately setting up and integrating enables you to improve your custom code management implementation and reduce the effort required for maintenance.</td>
<td></td>
</tr>
</tbody>
</table>
5 TRAINING

SAP offers knowledge transfer opportunities for an increasing number of topics and provides basic setup topics for all relevant areas. The SAP Enterprise Support Academy offers the following types of training:

**Figure 14: SAP Enterprise Support Academy Delivery Formats**

**Expert Guided Implementation Sessions**

For Enterprise Support customers, SAP offers Expert Guided Implementation Sessions (EGI). Expert Guided Implementation (EGI) sessions are a combination of remote training, live configuration, and on-demand expertise, which allow you to perform complex activities with the help of experienced SAP support engineers. The instructor will demonstrate what to do step by step. Afterwards, you can perform the relevant steps in your own version of SAP Solution Manager. If you have any questions, you can then contact an SAP expert via phone or e-mail.

EGIs are offered via SAP Enterprise Support Academy. For more information see [http://support.sap.com/esacademy](http://support.sap.com/esacademy).

SAP provides the following EGIs, which deal specifically with the setup guide and best practices for specific custom code management topics:

<table>
<thead>
<tr>
<th>EGI</th>
<th>Link</th>
</tr>
</thead>
</table>