PUBLIC

Document Version: 2.00 – 2015-11-06

SAP S/4HANA: Custom Code Adaptation

What you need to consider for your custom code when transitioning to SAP S/4HANA

Christoph Nake (SAP SE)

Table of Contents

INTRODUCTION -- 3

STEP 1: GET TRANSPARENCY ON YOUR CUSTOM CODE-- 4

STEP 2: REMOVE WASTE --- 5

STEP 3: ANALYZE THE IMPACT, AND CREATE A PLAN-- 6

STEP 4: DO THE REQUIRED CUSTOM CODE CHANGES---10

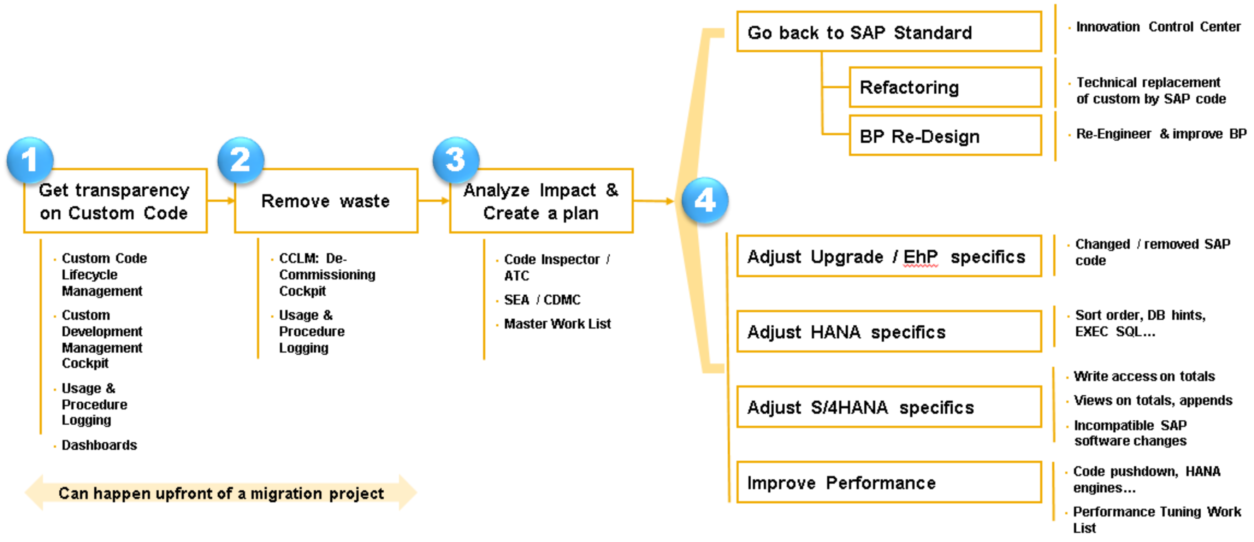
Introduction

First of all: Usually within an SAP system (e.g. ERP, CRM, SCM …) you develop your programs in ABAP – a programming language which interacts with the underlying database by using OPEN SQL. The SAP DB interface translates OPEN SQL commands into native SQL of the underlying database. As a result most of your own ABAP code will run on any data base, including SAP HANA (which is underneath SAP S/4HANA).

However, there are still some things to consider, and this blog would like to make you familiar with the main aspects of adapting custom code for SAP S/4HANA, and some valuable information sources.

Version 1 of the blog discussed the scenario when migrating to SAP S4/HANA Finance – which is technically an Add-On on top of SAP Suite on HANA. The functional changes are mostly limited to Finance. With SAP S/4HANA on-premise edition 1511 planned to be shipped by the end of 2015, things are different: The application layer (SAP_APPL) in SAP will be replaced by a set of simplified functionality. Many SAP applications will change. This blog version 2 also includes the new tools which are required to judge on the additional impact to your custom code.

So, imagine your company would like to upgrade from an old SAP release to SAP S/4HANA. The main things to consider with respect to custom code adaption are shown on the next picture. You will become familiar with a four-step procedure. Steps 1 & 2 require some time, so ideally they are done on a regular basis, and in front of a migration project. Steps 3 & 4 are often done as part of an SAP S/4HANA project.


1
Get transparency on Custom Code
Custom Code Lifecycle Management
Custom Development Management Cockpit
Usage & Procedure Logging
Dashboards
2
Remove waste
CCLM: De-Commissioning Cockpit
Usage & Procedure Logging
3
Analyze Impact & Create a plan
Code Inspector / ATC
SEA / CDMC
Master Work List
Can happen upfront of a migration project
4
Go back to SAP Standard
Innovation Control Center
Refactoring
Technical replacement of custom by SAP code
BP Re-Design
Re-Engineer & improve BP
Adjust Upgrade / EhP specifics
Changed / removed SAP code
Adjust HANA specifics
Sort order, DB hints, EXEC SQL…
Adjust S/4HANA specifics
Write access on totals
Views on totals, appends
Incompatible SAP software changes
Improve Performance
Code pushdown, HANA engines…
Performance Tuning Work List

Step 1: Get transparency on your custom code

You should know about the custom objects in your system. For instance:

- # of custom objects (reports or transactions) including modifications and enhancements.
- SAP objects referenced from your code
- The quality of your custom code
- Custom code documentation and business process documentation
- Support pack level of each SAP component
- Usage for your custom code and Processes
- Complexity or even the criticality of you code
- The relation between your custom code and the corresponding processes
- …

To be honest, you should always have this information handy – independently of any upgrade or transition event. In case you don't, here are some options for you what to do:

- Custom Code Lifecycle Management (CCLM) gives you always the current status of your custom code situation via dashboards, and a lot of tools to manage your code. For more information on CCLM: https://scn.sap.com/docs/DOC-59247
- Custom Development Management Cockpit (CDMC) is an older tool serving a similar purpose.
- In case you don't want to invest setup time, you can also get a snapshot on your custom code situation by ordering a Custom Code Maintainability Check for your SAP ECC system (Enterprise Support):
  - Info sheet
  - Sample report

For example, if you have 10 or 10,000 custom objects in your system, 1 or 100 modifications, and so on, now you know your first indicator on your adjustment effort.

Step 2: Remove waste

You don't need to adjust custom code which is not in use. To be precise, you don't need to maintain unused custom code at all. You can, and you should, decommission unused code. Why? Because custom code will cause additional work for you during each and every major SAP change event (SP installation, EHP installation, Upgrade…). Decommissioned code never needs to be adjusted again.

So if there is still some time until the migration to SAP S/4HANA starts, this is an ideal opportunity to remove such waste. But how do you know what code is actively used in your system, and what is not?

The solution is to switch on a monitoring feature in your productive system over some weeks' time (in case there is no time left, you need to skip this step – but please, start decommissioning unused code right after the SAP S/4HANA migration has finished!). It is called Usage and Procedure Logging (UPL). UPL tracks (with very little performance impact) what code has been executed. Please take care of also tracking special business events like month-end activities). See this link for more information how to set up UPL: https://scn.sap.com/docs/DOC-59249.

When you compare the list of used code, with the custom code inventory list, you know what code is not in use. The Decommissioning Cockpit as part of Custom Code Lifecycle Management in SAP Solution Manager supports you in that exercise, and helps you to find the right custom objects to decommission (http://scn.sap.com/docs/DOC-60525).

Now, all custom code which is still in your system should be subjected for further investigation.

Step 3: Analyze the impact, and create a plan

Now, you need to identify code, which **has to be** adjusted ("Must-Do's"), or which **should be** adjusted ("Should-Do's"). Let us discuss the scenarios:

- Depending on your start release, you may have to upgrade your SAP system first, or you may have to install some additional EHP's. In case you modified an SAP program, and SAP comes with a new version of this program as part of the upgrade or EHP, you will need to adjust your custom code and your modification (that is: keep it, or go back to SAP standard, or mix it somehow together. See how good it was to remove unused waste before?). Once you have done the upgrade or the EHP installation, you will see all modified objects that need to be technically adjusted in the transactions SPDD, SPAU or SPAU_ENH. However, there are two additional interesting options you should be aware of:
    - The Upgrade / Change Impact Analysis as part of CDMC is a way to estimate and manage the technical adjustment effort (Info Link) by comparing systems before & after upgrade, and analyzing the corresponding objects (e.g. referred SAP objects in custom code).
    - The Scope and Effort Analyzer (SEA) in SAP Solution Manager is also estimating the technical adjustment effort caused by EHP installation, by comparing the custom object and modified object list with the Technical Bill of Material (TBOM) of an EHP (thus this is doable in front of the EHP installation), and evaluating the adjustment costs and effort with information provided by SAP's backend. See Info Link and How-To Guide for more information.

- If your current system is not already on SAP HANA, you will need to change your database. SAP HANA behaves under certain conditions different than other databases. For instance, if you used database hints in your coding, those hints may have to be technically adjusted. Also, during DB migration to SAP HANA, cluster tables are converted to transparent tables. The implicit sort order of cluster tables is no longer a given – ABAP coding expects data coming to be automatically sorted from the database, and is subject for inspection (implicitly expecting sorted data is by the way pretty bad coding practice – In case you don't have any, some custom development guidelines can be found here: http://scn.sap.com/docs/DOC-56285).

  Since the tool of choice to find those coding areas, are the SAP Code Inspector (SCI – Info Link), and the ABAP Test Cockpit (ATC – http://scn.sap.com/docs/DOC-31773). SAP continuously ships new tool configurations (called "variants") to detect potential change candidates (see e.g. SAP Note 1912445).

- The first two cases are very common scenarios for each upgrade, and database migrations. But here is something new:
  The installation of the SAP Simple Finance Add-On V2.0 is an "Exchange Innovation". When compared to a common database migration, this SAP Add-On simplifies SAP code and the SAP table design (that means: It changes or deletes SAP code and SAP tables). Custom code, which references the changed or deleted SAP code or deleted SAP tables, is potentially impacted. However, the good thing is: Only a very limited number of SAP code and tables have been changed with this Add-On version. And read access to changed SAP tables still works by using compatibility views. SAP offers a check report, which tells you what custom code is impacted

(see e.g. SAP note 2176077, 2176652, and 1976487). Also, in case you extended SAP tables which are going to be changed during the Add-On Installation, SAP usually takes care of it automatically (see e.g. SAP note 2160045). Since SAP has plans to offer the Exchange Innovation Add-On in future, I keep this paragraph in the blog.

However, with SAP S/4HANA on-premise 1511, situation is different:

- SAP applications have been massively changed (simplified, removed, or replaced).
- The data model – the table structure – has been massively changed as well.

To make it clear: This is not surprising. SAP R/3 has been developed in the Nineties. The applications, and the underlying business data structure just reflected the technical limitations of the Hardware and Database at that time. To take full use of hundreds of CPU cores, and Terabytes of Main Memory which come with SAP HANA, and to create new business value out of it, SAP needs to massively change the software.
SAP Development has documented incompatible software changes, in a so-called "**Simplification List**". Look at this EXAMPLE – The full list will be published together with the new release software. This list is a great text document – here is what you could learn from there:

- The list gives you an indication what has been simplified, and what the benefits.
- Simplification items are categorized:
    - C1 simplification items do not have impact to the business user, or the business process. However, the incompatible changes might impact your custom code. Usually, per C1 item, there is a description how your code needs to be adjusted.
    - C2 simplification items identify consolidated / changed business functionality. The can have large impact on both your business users, and your custom code e.g. in case a certain functionality is completely replaced by a new application. C2 items are discussed at the end of this chapter.
- The Simplification List implies two questions you may have:
    - Do I have custom code which references C1 or C2 items from the list? Obviously that code needs to be adjusted: Most likely simply technical adjustment in case of C1, potentially complete sunset and redesign in case of C2.
      The tool of choice for answering this question is the "**Custom Code Analysis Tool**". This blog describes the tool and how it works in good detail. Currently the Custom Code Analysis Tool has high release requirements. However SAP has plans to offer a cloud based analysis service as an alternative way, so stay tuned in case you cannot fulfill these requirements.
    - Do I have applications in use which are affected by C2 items? Obviously these items need to be addressed before conversion starts. Otherwise, maybe unintentionally, business data and processes are negatively affected during migration.
      The tool of choice is called "**Pre-Check**". Pre-Checks are shipped via SAP Notes, and can be executed long before the migration starts. SAP note 2182725 "S4TC Delivery of the S/4 System Conversion Checks" (currently in Pilot Release) implements the main ABAP report R_S4_PRE_TRANSITION_CHECKS for Pre-Check execution. The note contains additional notes for the application specific pre-checks which must be manually implemented as well.
      As a result, you will get a list of yellow and red items with reference to SAP notes

which need to be cleared before migration starts.
To put your system on the save side: The same checks will be executed by the Software Update Manager SUM in front of a system migration. In case of errors, the migration stops.
    - By looking at the EXAMPLE, you will notice that there are cases where the compatibility view concept does not work. In those cases both read and write access to deprecated tables need to be adjusted.

  To summarize this paragraph on SAP S/4HANA on-premise 1511 – What you should do with respect to custom code:

    - Make yourself familiar with the Simplification List Example and the Custom Code Analysis Tool.
    - Study the Simplification List once fully released. Try to identify which C2 items are relevant for you – they may have large impact on your migration plans.
      Run the pre-checks once officially released to get a detailed list. Follow the SAP notes referenced in the list.
    - Run the Custom Code Analysis Tool once released to identify custom code which is affected by C1 or C2 items. Alternatively wait for the cloud service. SAP strongly recommends to filter the hit list by the UPL usage statistics. Please keep in mind: Custom code which is affected by C2 items potentially cannot / should not be adjusted, but completely decommissioned, because the whole SAP application will change.
      You will get a detailed list of affected custom code. Follow the SAP notes referenced in the list.

- The first three cases from above describe “Must-Do checks” – you have to correct the affected code, otherwise it won’t work properly. The fourth case is a “Should-Do case”, which looks at performance. SAP HANA is an In-Memory & column based database. A lot of SQL processing runs dramatically faster, but in some cases custom code performance might not be ideal. SAP offers tools to get transparency on custom code performance, and to identify custom code. Both of which are good indicators to look at from a performance perspective to keep your business users happy:
    - The SQL Monitor (SQLM) runs in the productive ECC system, and lets you easily identify custom code which puts heavy load on your DB. There are many good information sources out there, e.g. Info Link.
    - The SAP Performance Work List tool (SWLT), runs in the development system, and combines data from ABAP code analysis and runtime data from SQLM to get a ranked list of custom objects. See an Online Tutorial at YouTube.

- Finally, there is the growing area of truly simplified and optimized business processes, where SAP has completely changed the way of processing business data, or where the whole SAP functionality has been completely simplified and changed (including a move from SAP GUI layout to FIORI design). These are the C2 items in the simplification list. The EXAMPLE contains some cases already: “Foreign Trade replaced by GTS”, or “Revenue Recognition replaced by New Revenue Recognition Functionality”. Here, a simple adjustment of some SQL statements within a related custom program definitely falls too short. C2 items belonging to functionality which is in

productive use today lead to a functional implementation project as part of the overall migration project to SAP S/4HANA – including all related custom code items.
This is an ideal opportunity for you to think about:

- A decommissioning the old custom code in general by going back to the SAP standard. SAP is currently developing new services and tools offers around "**Back to Standard**". An interesting blog which describes a new approach – by focusing on complex custom code items which have caused high maintenance efforts in the past - can be found here.
- A re-design of your custom code in this space, leveraging the new business functionality offered by SAP S/4HANA. Inspire yourself by looking at the various information sources on SAP S/4HANA, e.g. at SCN, or SAP.com.
- A professional gap management. Obviously custom code adaption causes efforts on your end. Better managing gap requests from your business properly right from the beginning – before custom coding starts. SAP Best Practice in that space is the Innovation Control Center (ICC) (Info Link).

By running through all cases, you will create a work list of custom objects which must or should be changed. There are multiple ways to prioritize the work items (e.g. Must Do's first, business critical code first, custom objects with high error / change rate first…).

Step 4: Do the required custom code changes

You will need to execute the custom code adaption as part of the migration project. The adaption itself will take place in the migrated DEV system, and is transported to QAS and PRD once those systems have been migrated to SAP S/4HANA.

Finally, I would like to make you aware of the following information sources:

- Having full control of your custom code becomes more important in future, because it enables you to easily consume SAP simplifications. Custom Code Management has to become essential in your Application Life Cycle. At SCN you will find all relevant information on this topic here (e.g. the new SAP Support Standard on Custom Code Management (Info Link)).
- The SAP Enterprise Support Academy offers remote training sessions on custom code management (so-called "Expert Guided Implementation (EGI)"), which is available for Enterprise Support customers. One EGI where you learn how to set up and use the tools mentioned in this blog : Make your Custom Code Ready
- A Best Practice Guide on custom code management, which more focuses on a structured usage of the local expert tools is: Considerations for Custom ABAP Code during a Migration to SAP HANA.

These examples will always help you have your custom code under control, not just in case of a migration to SAP S/4HANA.

The Best-Run Businesses Run SAP®