

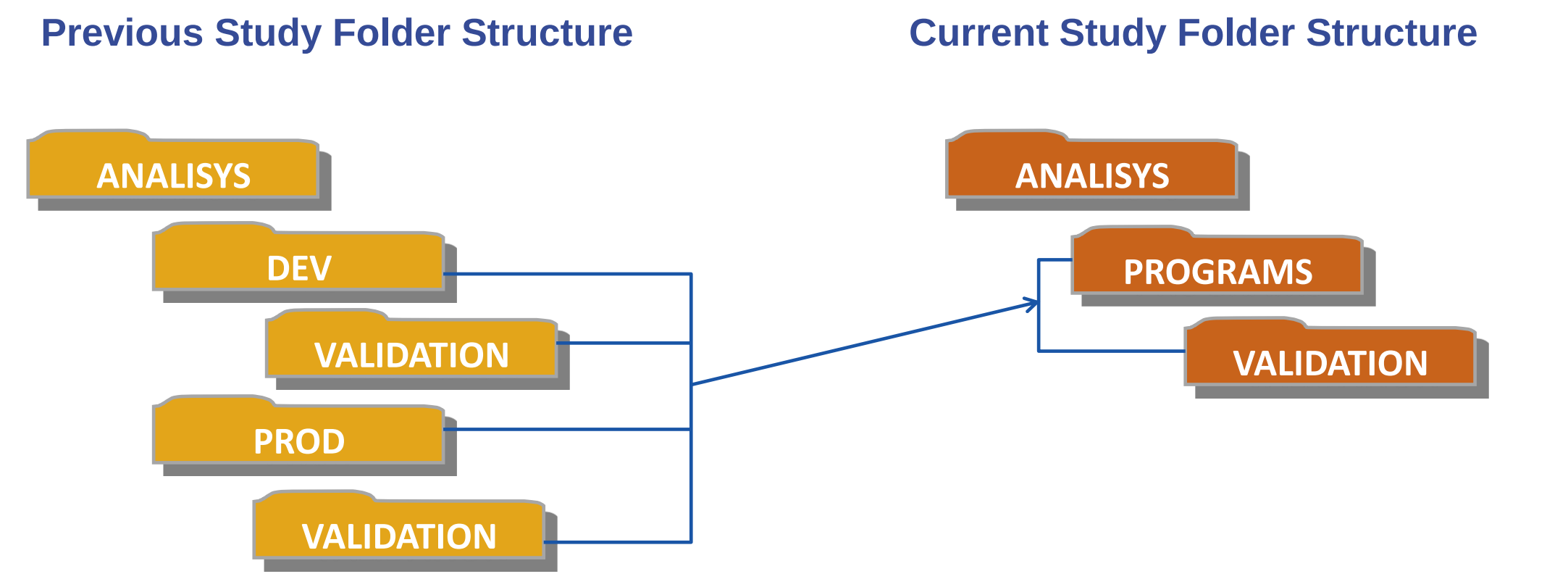
EMM (Environment Macro Management)

Giuseppe Di Monaco, UCB BioSciences GmbH, Monheim, Germany

1. Introduction

Designing a project for the automation of the entire lifecycle of standard macros needs investment, support, hard work and dedication. A good automation of the standard macro lifecycle will ensure that when standard macros are developed they will be successfully validated and deployed to the study programmers in a timely manner. The automation process of the macro lifecycle must be also efficient, reliable, predictable and easy to maintain even when there is a major change in the environment of the standard macros.

I was involved in implementing a major change to the study folder structure (as shown in the below image) and all standard macros, non-standard macros and programs needed to be re-validated.

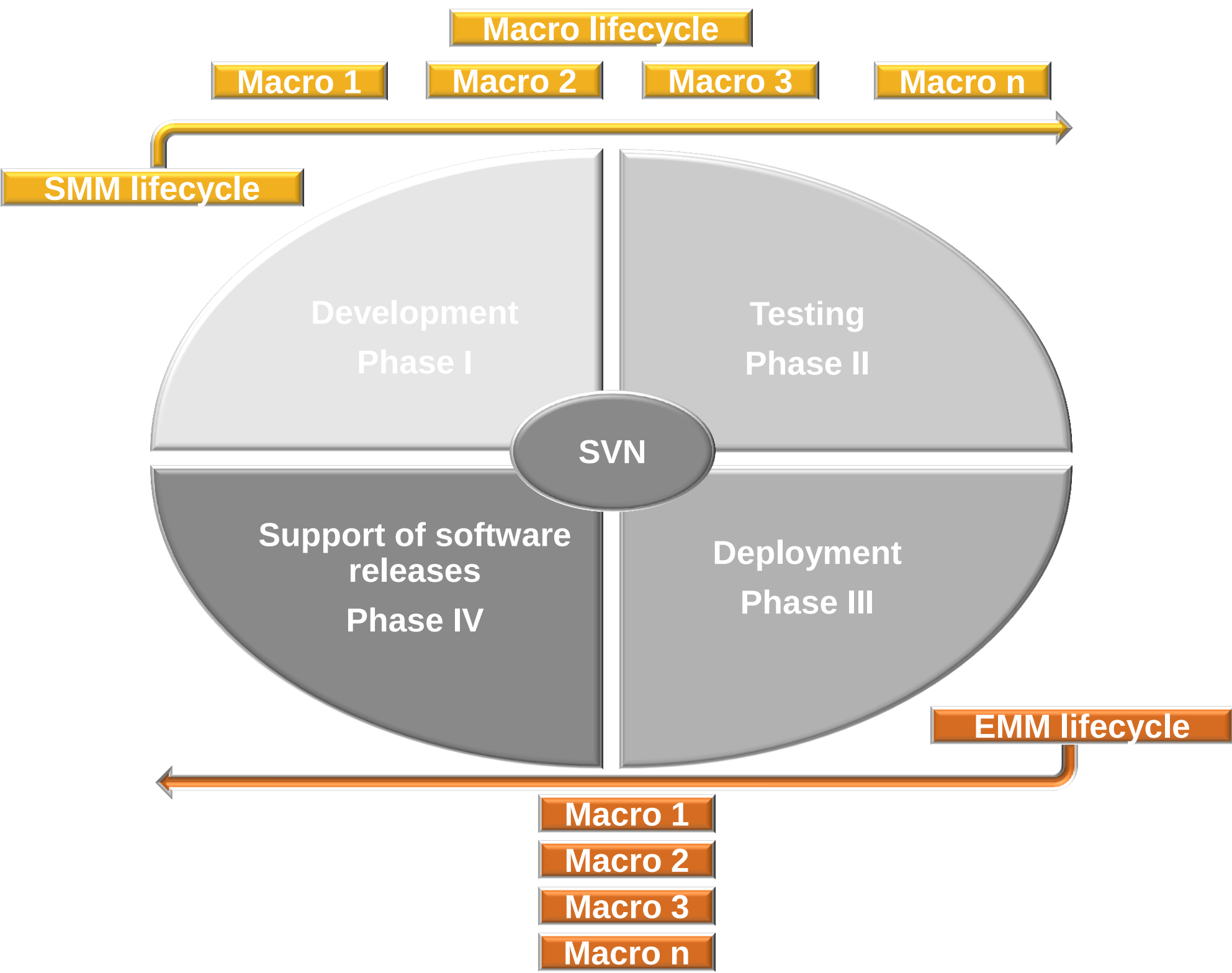


I was chosen as a programmer for this task. As I was very new to the company I started with walk-through sessions with key individuals involved in the standard team. The first step I had to make was to understand what the previous macro management was. From these sessions it was clear that the previous validation process could not be used because the architecture of the macro environment was not reflecting the architecture of the study folder structure.

2. Macro management lifecycle

The previous standard macro management (SMM) was based on a single macro lifecycle. Branches were added for each new release. As a consequence the environment was replicated at each versions and so were the number of files for programs and documents.

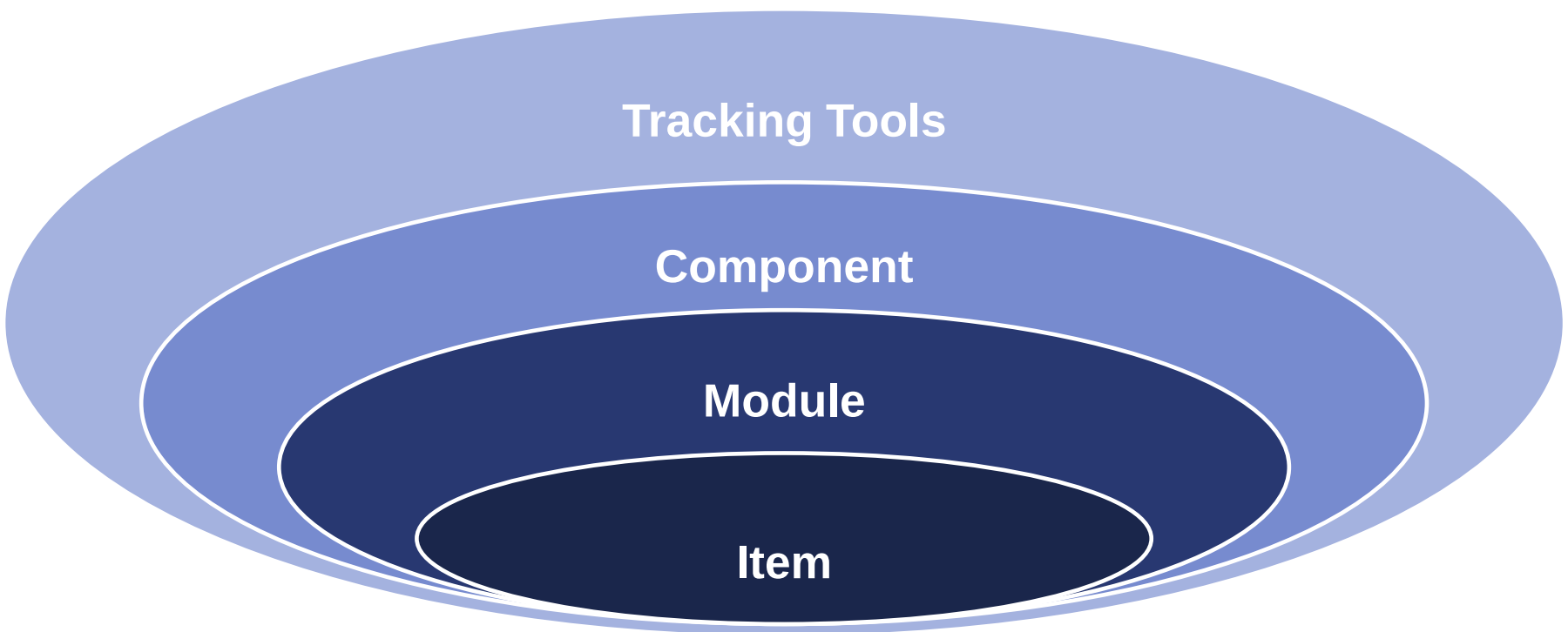
Unlike the SMM, the EMM does not replicate the environments, in fact the entire environment goes only once around the cycle for each version. The image below shows both SMM and EMM lifecycle. The EMM is able to create a release of a full validated package (with all macros, programs included) with **just one click**. This was possible by making the most of Tortoise SVN (freeware application used to control versioning of files), process automation and creation of configuration Management tools. (Using MS-DOS® batch programs, SAS®, Microsoft® Excel®)



3. EMM defined by abstraction

The EMM is composed by environments that are nested in an onion-like layered architecture, each level uses the functionalities offered by its immediate inner layer and is used by its outer layer, so that external levels are more abstract and easier to use.

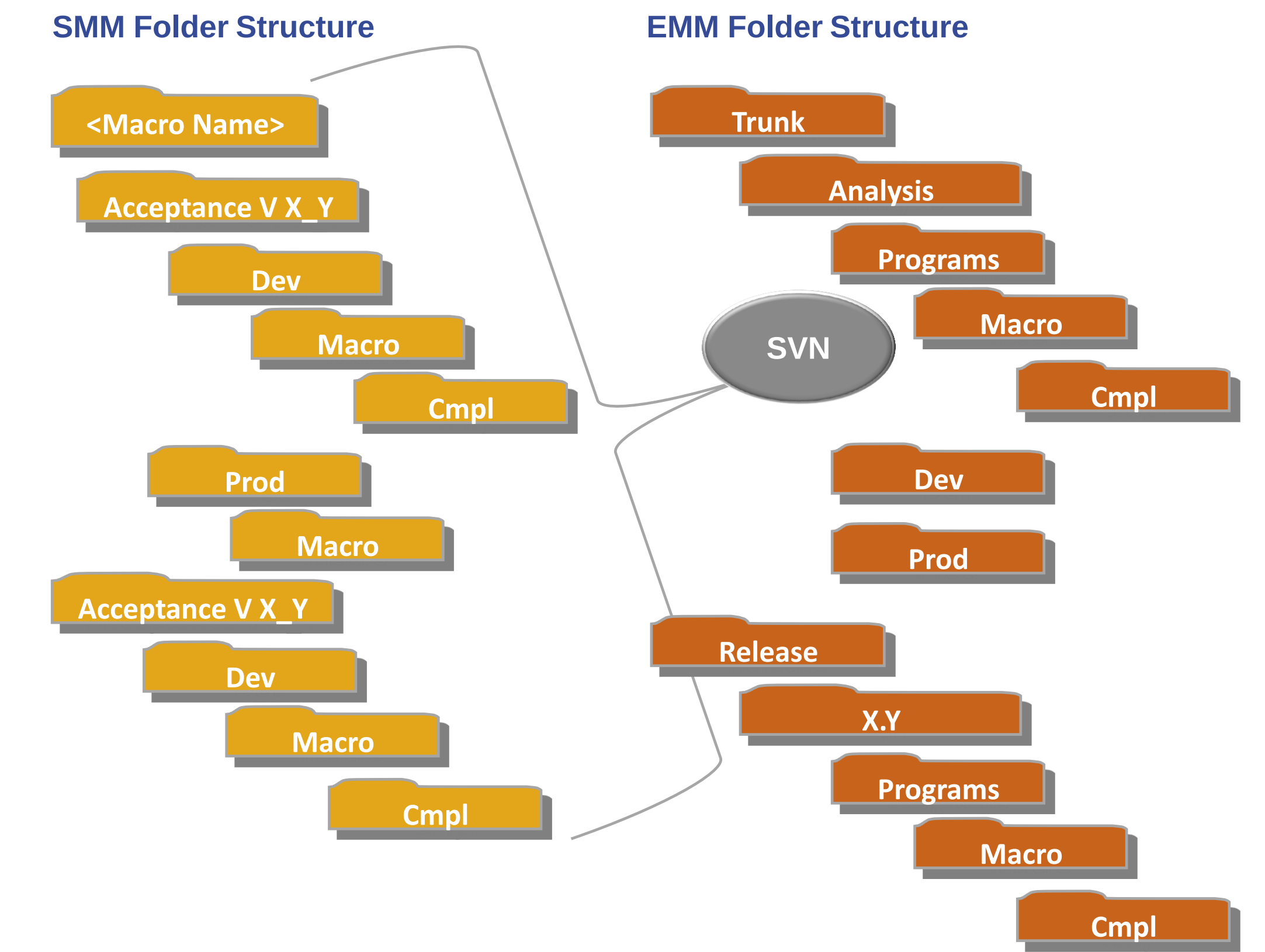
- Tracking tools
 - a visual configuration management map for all our products. It includes but not limited to, bug reporting, features/changes requests, test cases information, documents etc...
- Component
 - a set of modules and Items designed to act together or in sequence
- Module
 - element to be part of delivery that can be tested as an entity
- Item
 - element that cannot be tested separately and is meant to be tested and delivered with a module or a component



4. Configuration management Changes to the Folder Structure

The structure of the SMM affect all phases of the macro lifecycle and it is the cause of the replication of the environment. The name of the macro is the root folder and it contains as many branches as the number of the macro version released.

The structure of the EMM does not cause the replication of the environment at each version. At one point in time there is only one file for each macro and for each document. All previous versions of these files are kept in SVN server. The EMM contains only two main roots, trunk and release.



Naming Conventions

In order to promote consistency and avoid ambiguity, naming conventions were chosen. Having conventions also facilitates the implementation of the automations. For example, all test cases and related documentations were renamed according to the convention definition. For test case names the following convention was chosen: <macro name><test case number>_<running mode>.sas where running mode could be either interactive or production running.

Groups

Having only one root location (Trunk) has the downside of having also unique inputs and outputs location for test cases. In order to overcome the problem of overwriting inputs and outputs, a concept of test case group was introduced.

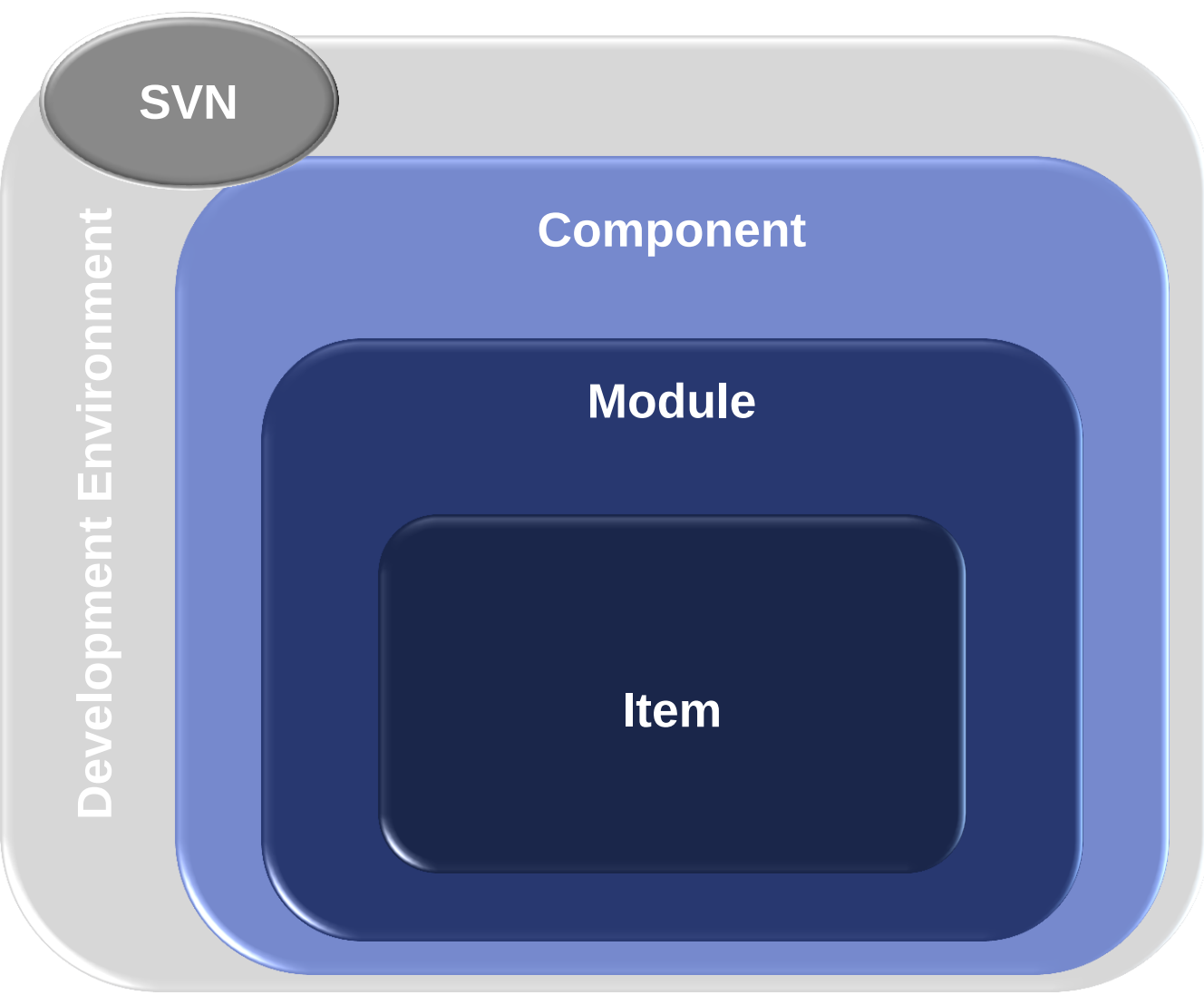
- A test case group consists of:
- Test case program
 - Test case log & reference log
 - Test case listing & reference listing
 - Test case inputs
 - Test case outputs & reference outputs

Each Test Case Group has its unique code. The code has the following structure: TCXXY where XX is the two digit number which uniquely identify the test case, Y is a letter (D or P) indicating whether the test is meant for interactive or production run mode.

5. Phase I - Development Definitions

In EMM all products are developed exclusively under the root trunk and its history is always ensured by Subversion, hence this location always contains the most up to date version of our products, but therefore may also be the most unstable version!

The implementation of the products follow the same onion-like layered architecture of the environment. This architecture facilitates the separation of high-level components and low-level modules and items by layers (interfaces). The main benefit of adopting this infrastructure is the reuse of low-level components/modules/items.

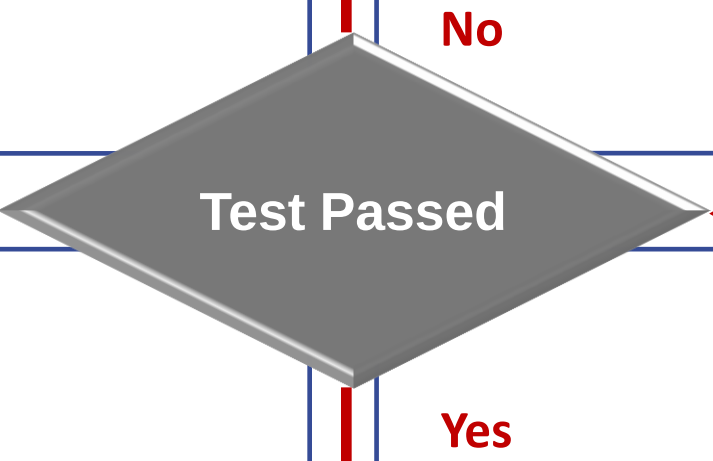
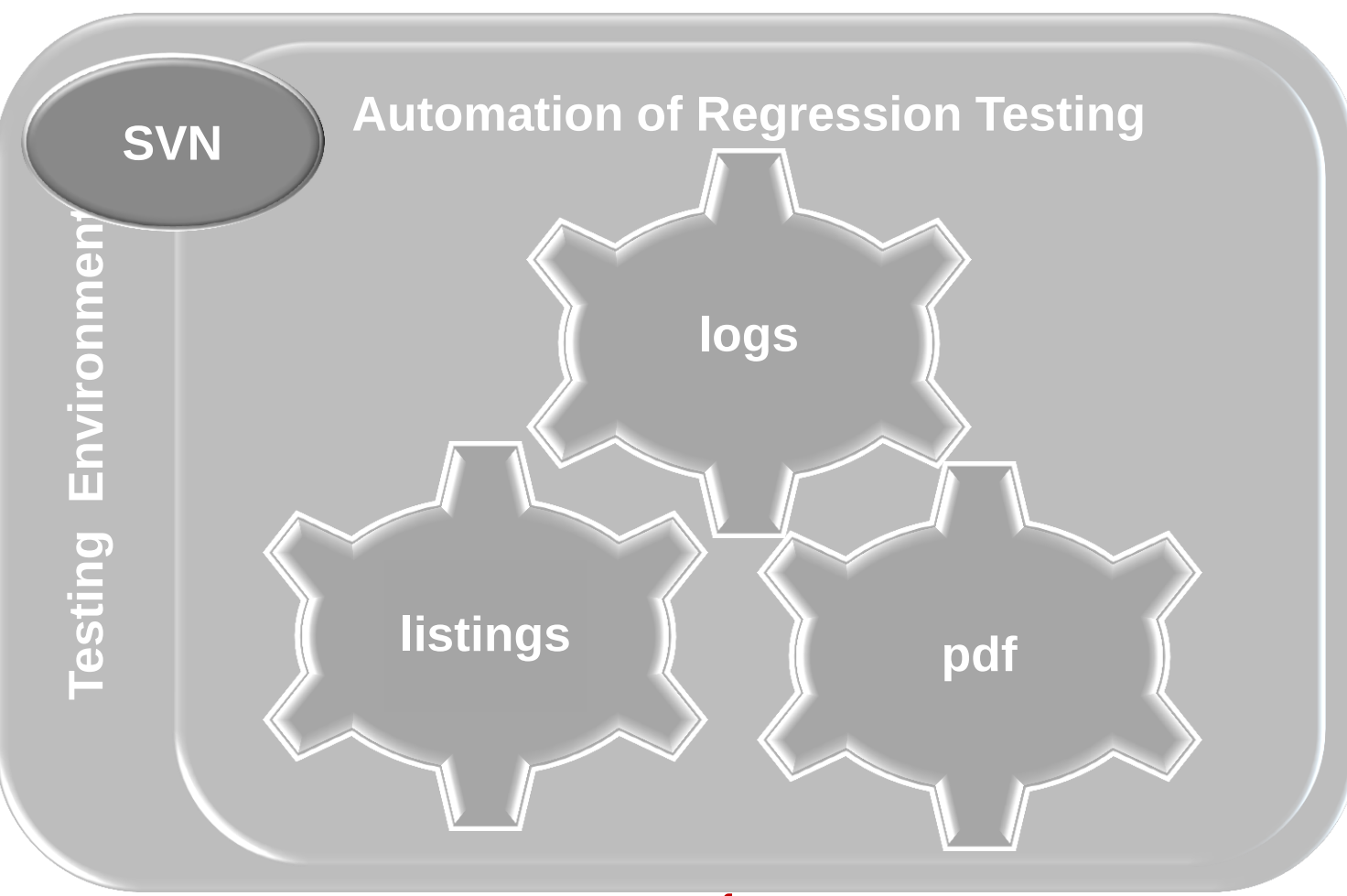


6. Phase II - Testing Automation Process Implemented

In EMM all outputs created by the test cases (logs, listings, pdf etc..) are automatically checked and compared against expected results. The automation will check if the expected results matches the actual results. This is done by importing the expected results in to SAS and comparing them with the actual results.

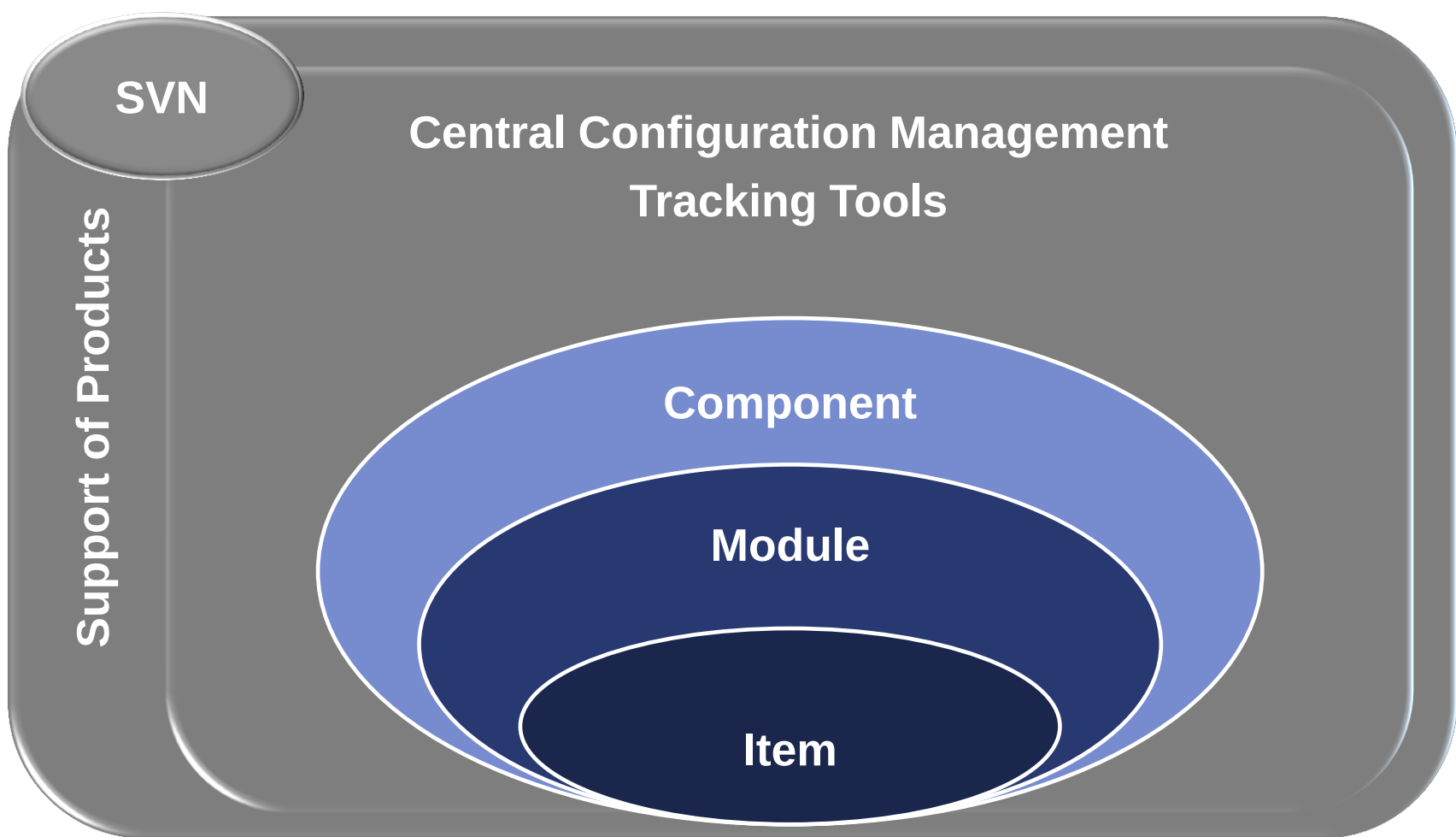
The only manual task is to place the new test cases in to the queue of the regression testing and define the expected result.

Testing Environment is also version controlled and the working version always contains the latest run of the regression testing automation.



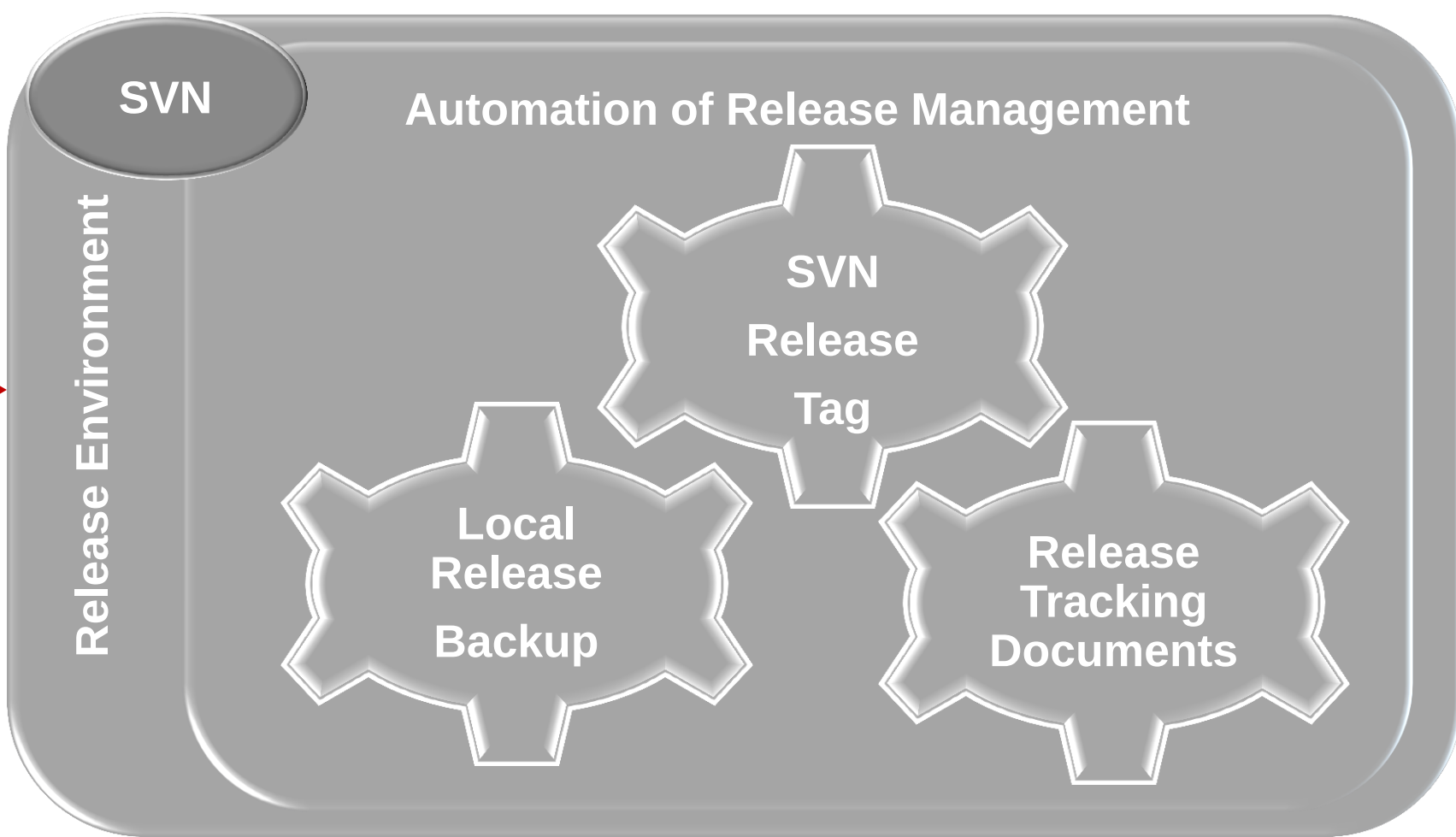
8. Phase IV Support of Release

All Information related to the support of products is centralized within the tracking tool. Its main functions is to provide information regarding the status of each macro, the relationships with other macros and all related information which allows for things like the reconstruction of such macros, at any point in their existence.



7. Phase III Automation of Release

Release management in the SMM did not offer an adequate level of automation and relationships between macros was difficult to maintain since each macro was released separately. In EMM all products are released at once as a package. The package release can be seen as the container to ensure that changes are packaged, released and tested in a repeatable and controlled manner.



- In addition, the automation of the release phase creates:
- release tracking document (keeps track of release across versions)
 - a tag in SVN (backup)
 - export the tagged package to local drives