

**A Modern Software Design Principle Applied To SAS Macro
Programming:**

The Inversion Of Control Concept

Company

- **HMS Analytical Software is a specialist for Information Technology in the field of Data Analysis and Business Intelligence Systems**
- **Profile**
 - 40 employees in Heidelberg, Germany
 - SAS Institute Partner for 15 years
 - Doing data oriented software projects for more than 20 years
 - Focus on life science industry
- **Technologies**
 - Analytics and Data Management: SAS, JMP, R, Microsoft SQL Server
 - Application Development: Microsoft .NET, Java



Our IT Services for the Life Science Industry (SAS, JMP, R and Microsoft)

- Independent Consulting
- Programming
- Data Management
- Data Mining / Analysis
- Training and Individual Coaching
- Application Development and Integration
- Software Validation



Two everlasting questions..

.. in software development:

- „Are we building the right product?“
(-> Validation)
- „Are we building the product right?“
(-> Verification)

Are we done, if our
code is free of
errors?

No!
Software quality is
more than lack of
errors!

Overview

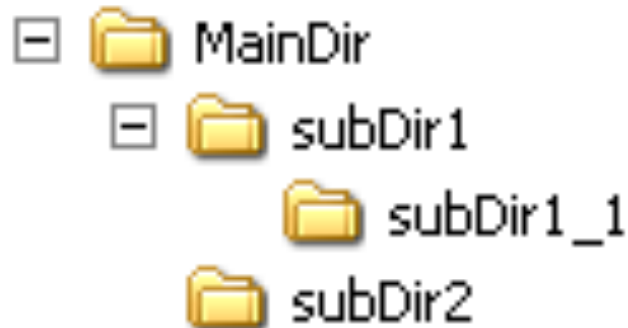
- **Inversion of Control – A Software Design Concept**
- **Inversion of Control applied to SAS programming**
- **Conclusion**

Inversion of Control (Ioc)

- **General principle:**
- **implement high-level source code more independent from low-level code**
- **Dependency relations are inverted or decomposed**
- **Advantages:**
 - more reusable code
 - Decoupled code with less risk of side-effects when code has to be changed

Example Task

Search a directory tree for SAS files...



.. and für every SAS file...



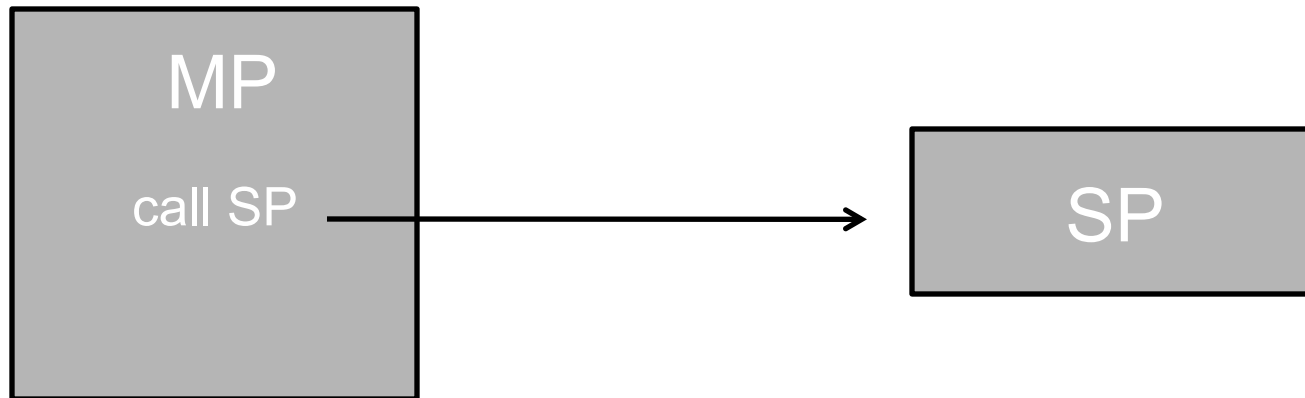
.. generate a PDF report.



First Solution Approach To Example Task

Main Program (MP)
“traverseDirForSasFilesAndDoTask”

Sub Program (SP)
“getSimplePdfReportOfSasFile”



What if there are other tasks to be done for every SAS file in a directory tree?

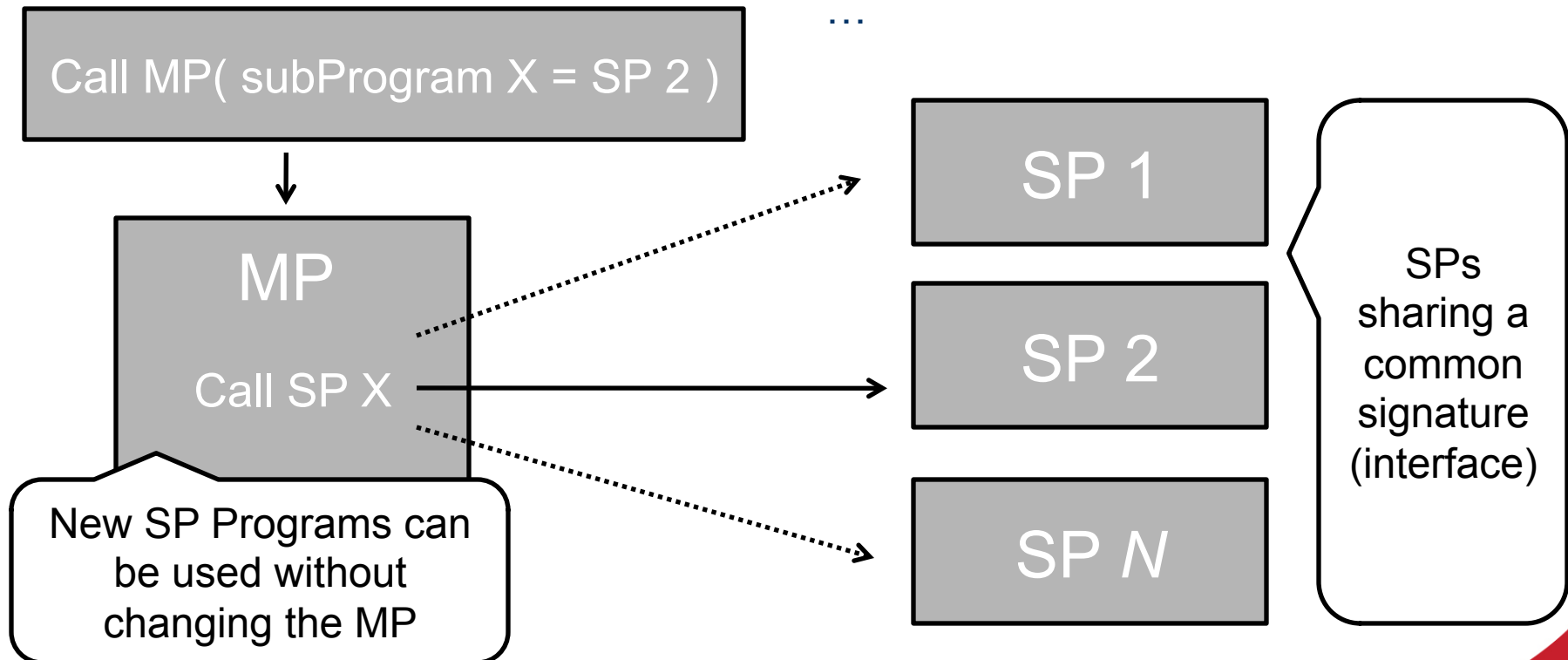
Support several file tasks: Solution approaches

- For every new task, copy and rename main program and call a different Sub Program
-> **bad solution**
- Create a new parameter for the main program for selection of subtask to be executed, conditional call of sub programs
-> **better solution**
- Make your main program independent of actual subprograms, rather program against a certain subprogram-signature (interface); create one or more parameters defining the actual subprogram to be executed
-> **even better solution**

loc Solution Approach To Example Task

Main Program (MP)
“traverseDirForSasFilesAndDoTask”

Sub Programs (SP)
“getSimplePdfReportOfSasFile”
“convertSasToCsv”
...



Inversion of Control applied to SAS programming

- **Simple example macros:**

```
%MACRO callTwice(aString, outputVariant);  
  %DO i=1 %TO 2;  
    %&outputVariant(&aString.);  
  %END;  
%MEND callTwice;
```

```
%MACRO simplePut(stringToOutput);  
  %PUT(&stringToOutput.);  
%MEND simplePut;  
%MACRO upcasePut(stringToOutput);  
  %PUT(%UPCASE(&stringToOutput.));  
%MEND upcasePut;
```

- **Examples for the call of the main macro are:**

```
%callTwice(testString, simplePut)  
%callTwice(testString, upcasePut)
```

Inversion of Control applied to SAS programming

- **Application to the example task, searching a directory tree:**

```
%traverseDirForSasFilesAndDoTask(  
  dir=C:\Temp\TestloC,  
  taskMacro=convertSasToCsv  
)
```

```
%traverseDirForSasFilesAndDoTask(  
  dir=C:\Temp\TestloC,  
  taskMacro=getSimplePdfReportOfSasFile  
)
```

Inversion of Control applied to SAS programming

- **Using a flexible number of parameter for submacros:**

```
%MACRO callTwice2(mNameToCall, mParameters);  
  %DO i=1 %TO 2;  
    %&mNameToCall(p0=a, %UNQUOTE(&mParameters));  
  %END;  
%MEND callTwice2;
```

```
%MACRO m1(p0, p1);  
  %PUT outputM1: &p0 &p1;  
%MEND m1;
```

```
%MACRO m2(p0, p1, p2);  
  %PUT outputM2: &p0 &p1 &p2;  
%MEND m2;
```

```
%callTwice2(mNameToCall=m1, mParameters=%STR(p1=first Call))  
%callTwice2(mNameToCall=m2, mParameters=%STR(p1=second, p2=Call))
```

- **Alternatives:**

- String lists that you explicitly process with %SCAN
- Parameter tables consisting of name-value pairs

Conclusion

- **The IoC can be easily applied to SAS programming**
- **Macros only rely on a certain interface describing a group of sub-macros.**
 - More reusable code that is decoupled from specific sub-macros
- **Caller-macros that are calling sub-macros do not have to be changed if you need to call a new variant of the sub-macro**
 - Less risk of side-effects when you change your code

Thank you for your attention

Dr. Patrick René Warnat

**HMS Analytical Software GmbH
Rohrbacher Str. 26
69115 Heidelberg
Germany**

www.analytical-software.de