Codemanagement

Entwicklung von Standardsoftware mit Varianten

Claudia Fritsch

claudia.fritsch@gmx.net

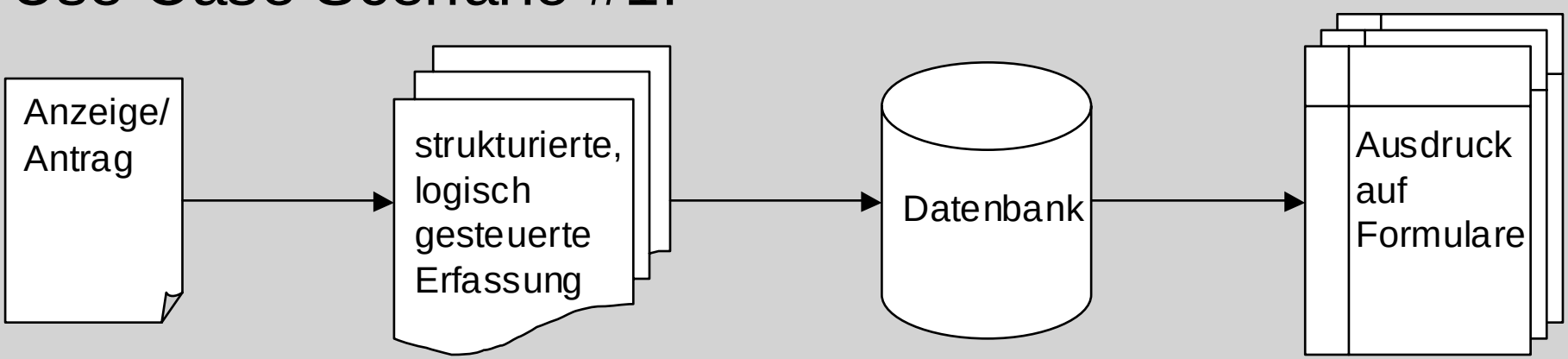
Übersicht

- Die Software
- Architektur – Stable Design Centers
- Gemeinsamkeiten und Unterschiede
- MKS Source Integrity
- Codemanagement
- Zusammenfassung und Lessons Learned

Software

- Standardsoftware für PCs und PC-Netzwerke
- Verarbeitung personenbezogener Daten
- Expertensystem

Use Case Scenario #1:



Programme

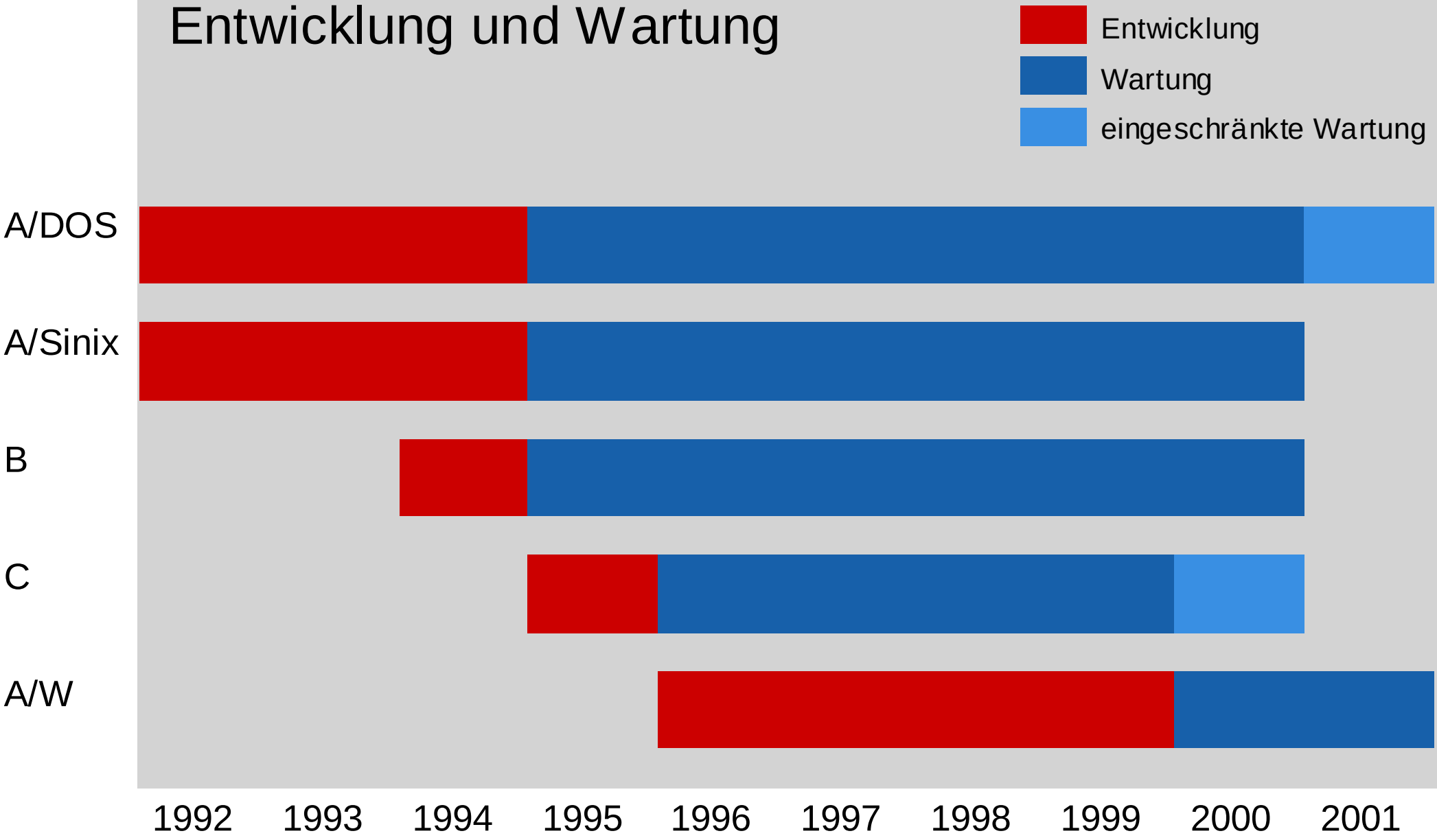
- A/DOS Programm A, entwickelt unter DOS
- A/Sinix Programm A, portiert auf Sinix RM
- B Programm B, 1. Variante von A
- C Programm C, 2. Variante von A

  DOS-Programm für PCs und PC-Netze (Novell),
  Raima Database Manager,
  entwickelt unter DOS, objektorientiert in C, MS C/C++ 7.0-Compiler,
  MKS Toolkit und rcs/Source Integrity, make.

- A/W Programm A für Windows

  32 bit Windows Client/Server-Lösung für Windows NT/95/98/2000,
  Sybase SQL Anywhere, Oracle, MS SQL Server via ODBC,
  entwickelt unter Windows NT mit C++, STL, MFC, MS VC++-Compiler,
  MKS Toolkit und Source Integrity, make.

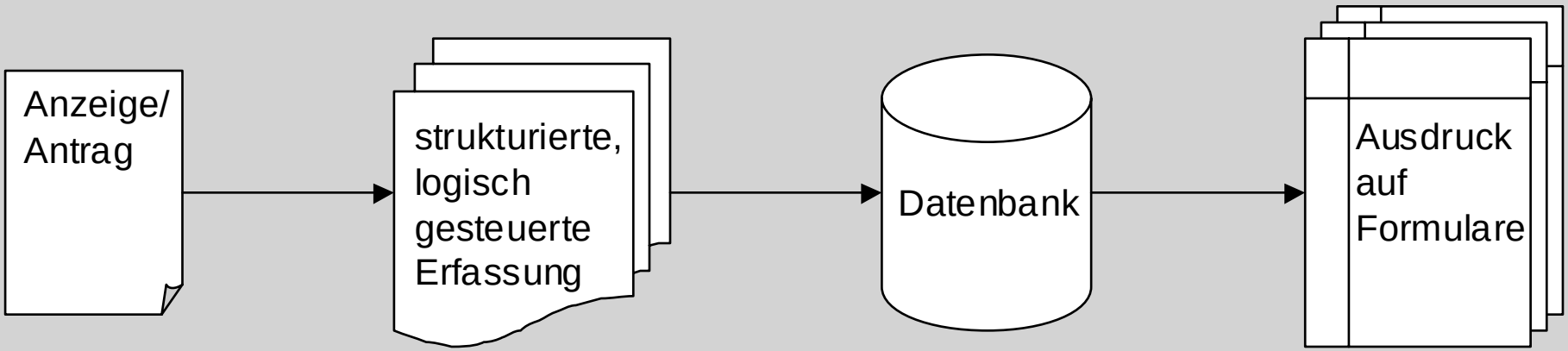
Entwicklung und Wartung



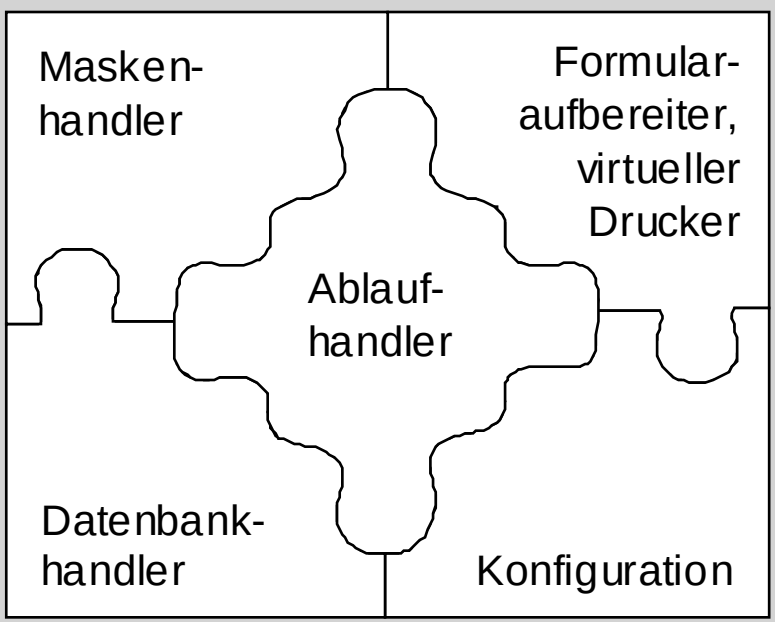
Architekturtreiber

- plattformunabhängig
  - DOS Einzelplatz und Netzwerk
  - Sinix
  - Windows und andere Datenbanken absehbar
  - diverse Drucker
- fachlich leicht adaptierbar
  - fachliche Anforderungen ändern sich ständig
  - rechtliche Korrektheit vertraglich zugesichert
  - häufige Updates
- hochkonfigurierbare Standardsoftware
  - anpassbar an örtliche Gegebenheiten
- Abbildung des gewohnten Arbeitsablaufs

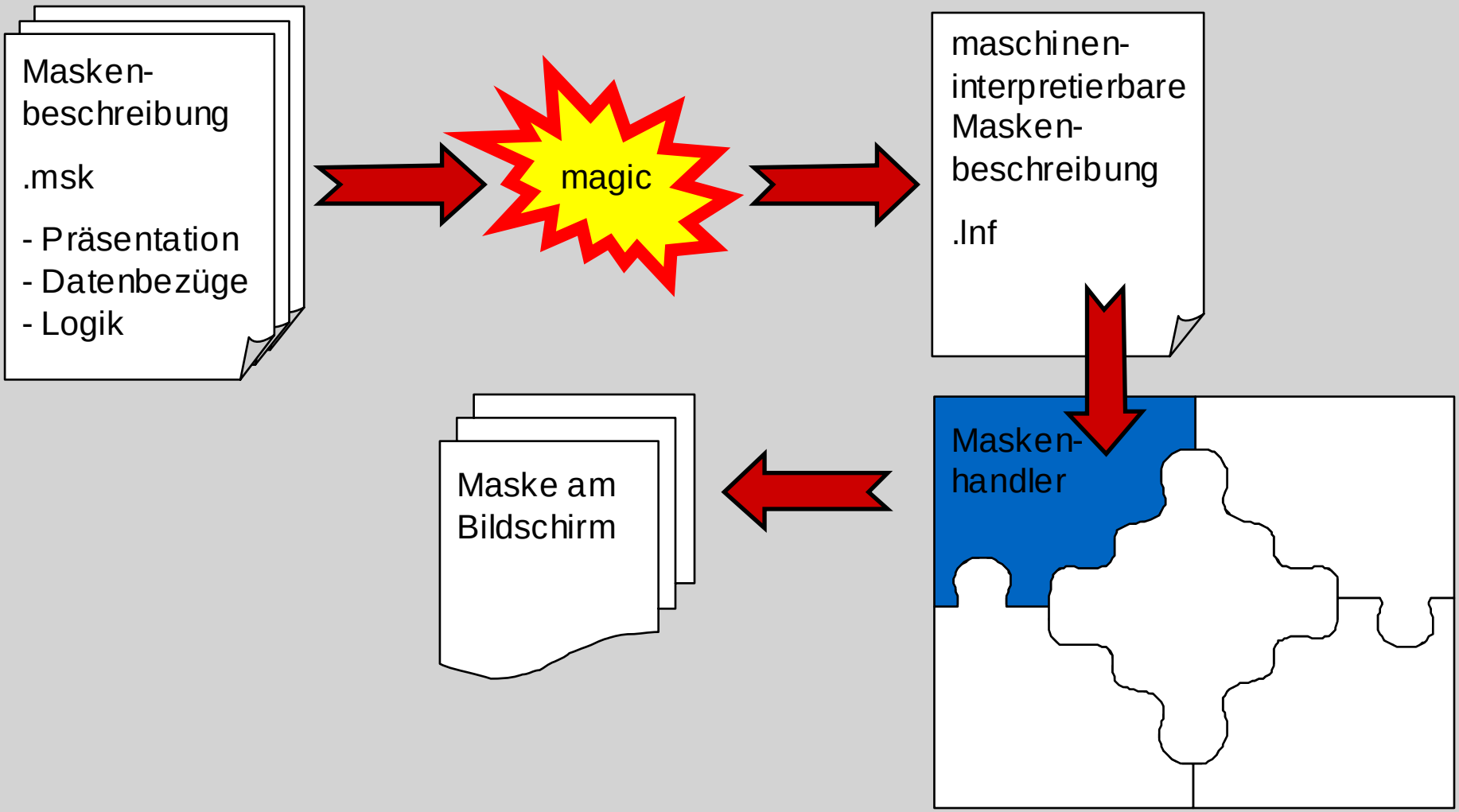
Use Case Scenario #1:



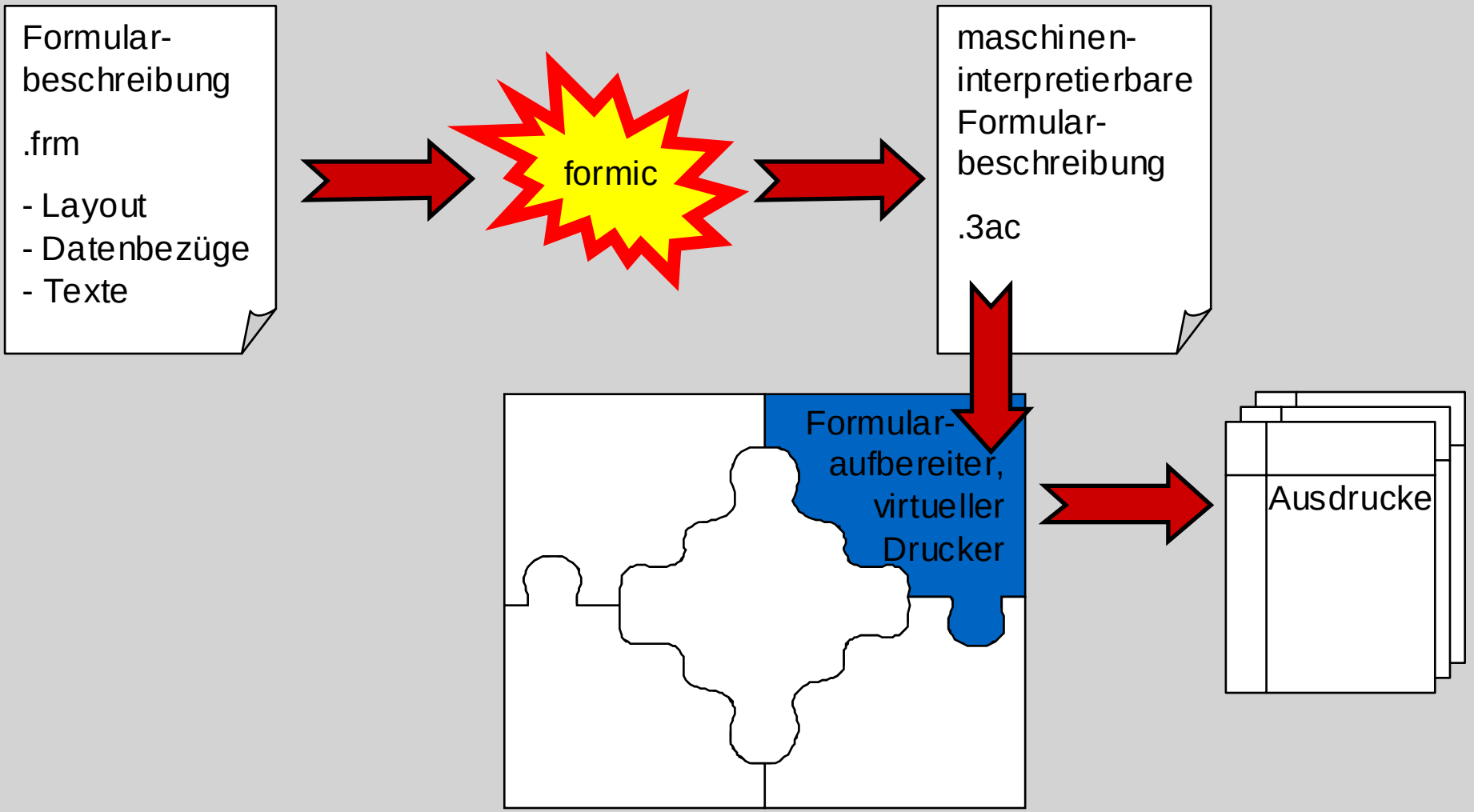
Verarbeitung im Programm:



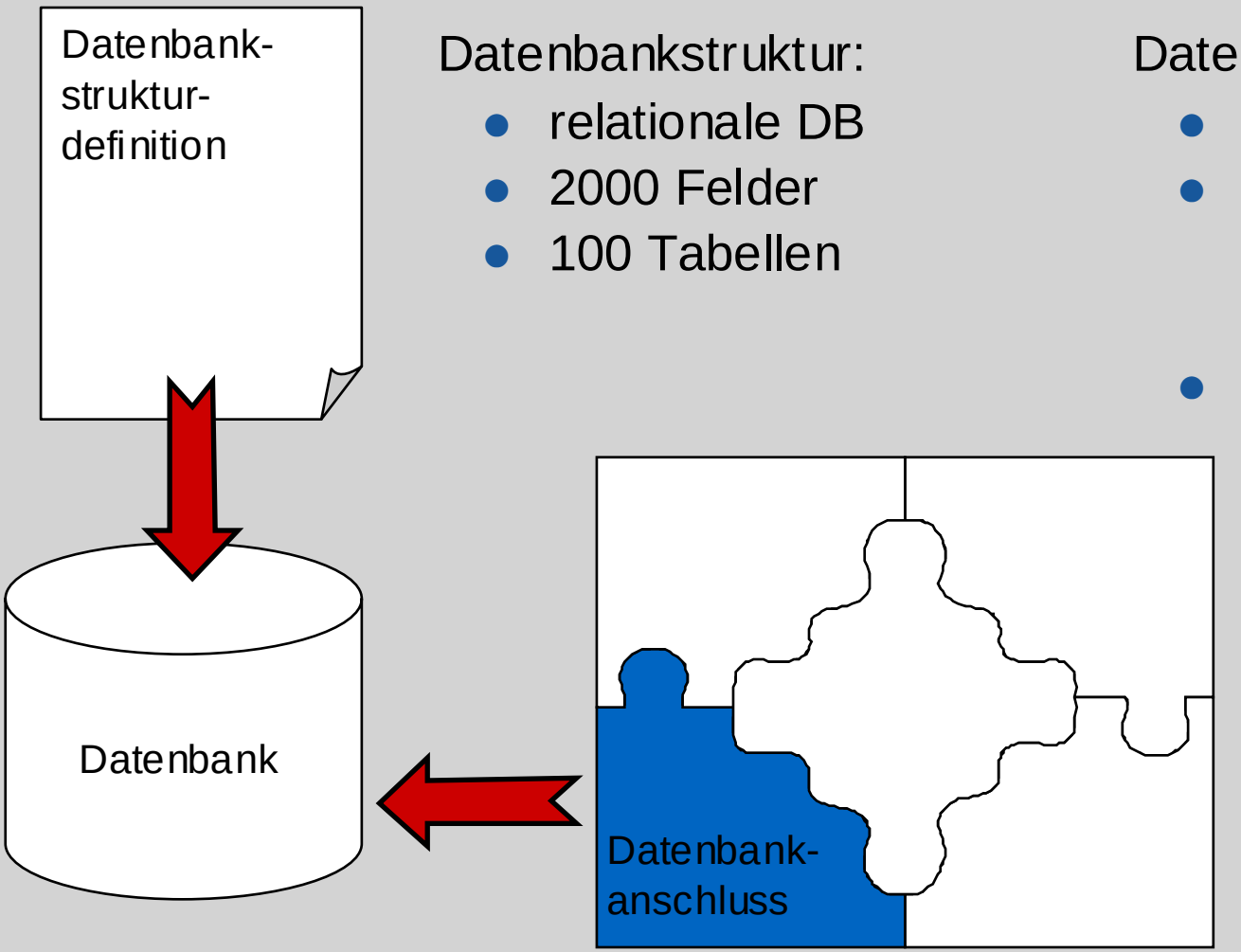
Maskenhandler



Formularaufbereiter



Datenbankanschluss



Datenbankstruktur:

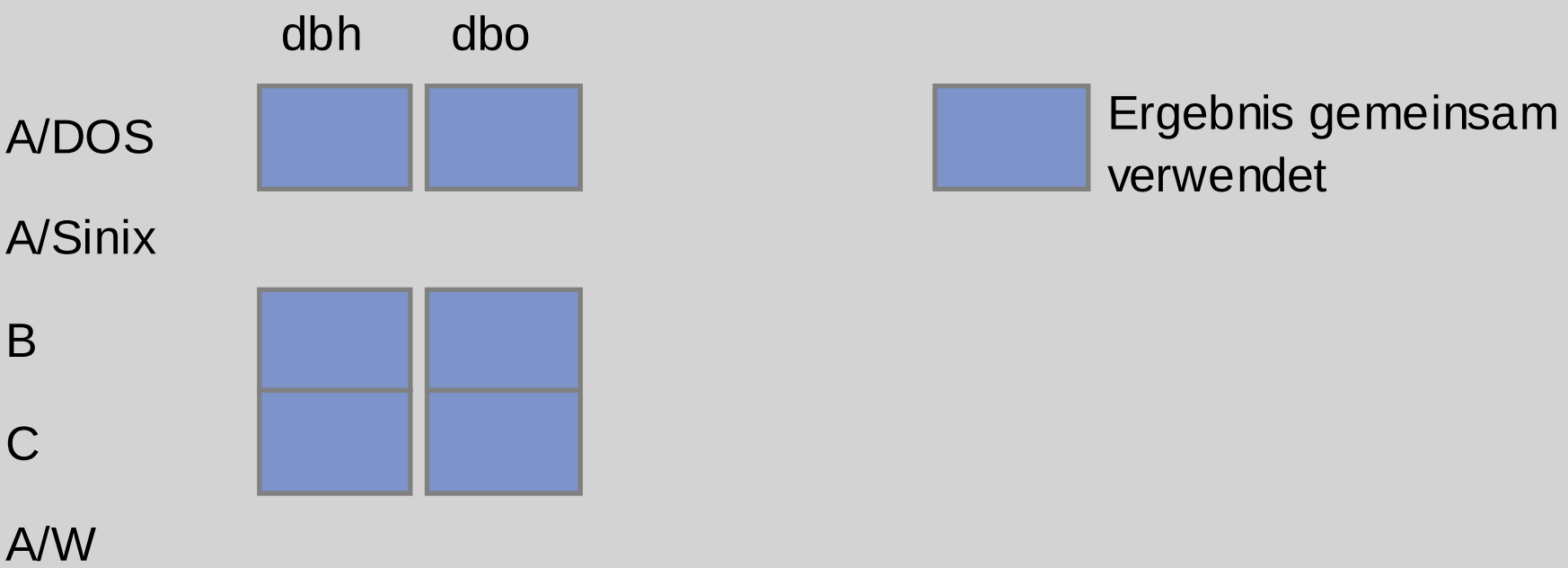
- relationale DB
- 2000 Felder
- 100 Tabellen

Datenbankanschluss:

- zweistufig, oo
- dbh: Datenbankhandler, Progammierschnittstelle oder ODBC
- dbo: Datenbankobjekte für Zeilen, Vorgänge, Listen

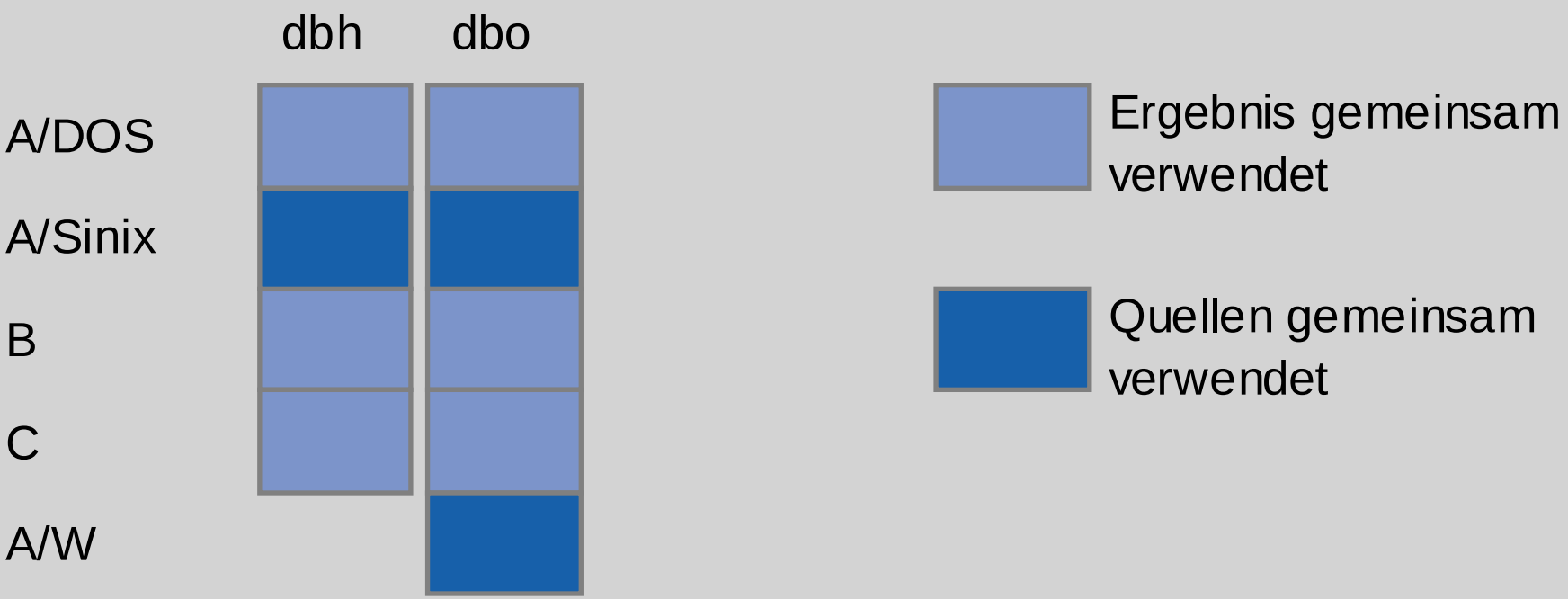
A) Projekte verwenden gemeinsame Ressourcen oder Ergebnisse

Beispiel: Datenbankanschluss



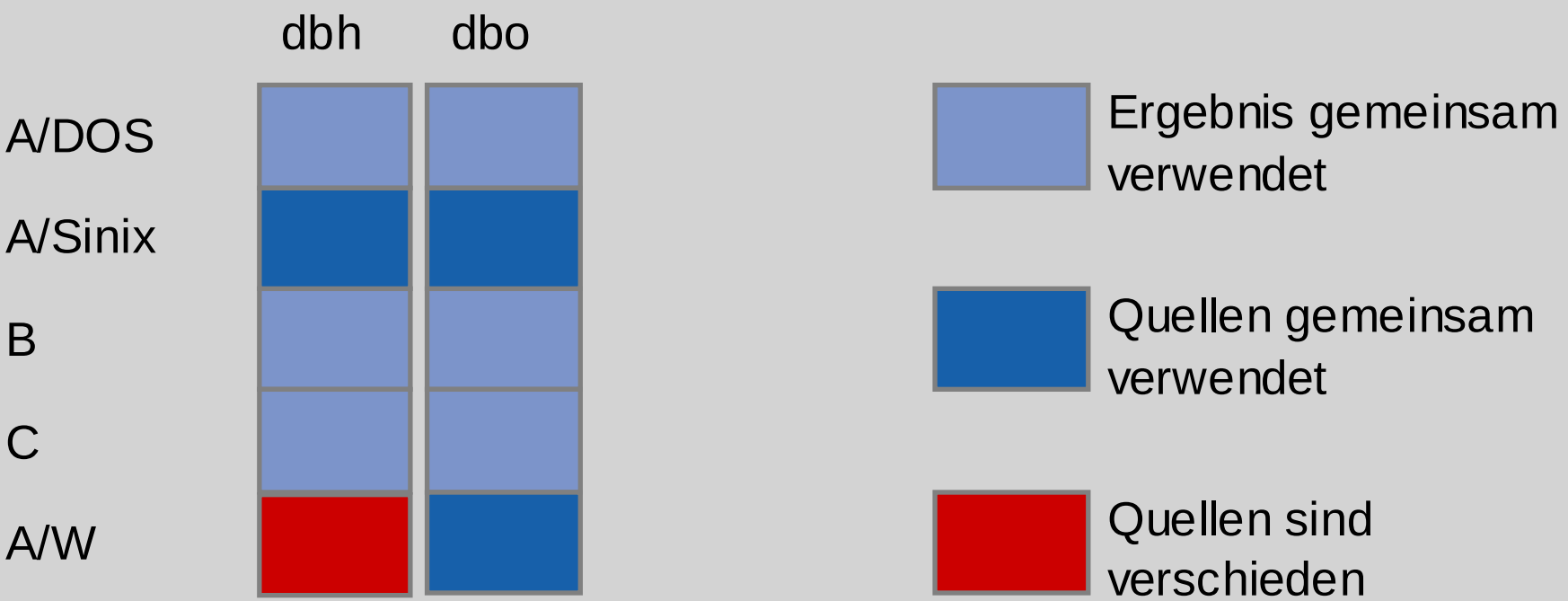
B) Plattformunterschiede: gemeinsame Quellen, verschiedene Compiler

Beispiel: Datenbankanschluss



C) Plattformunterschiede: Differenzen in Quellen

Beispiel: Datenbankanschluss



D) Unterschied in Daten, Ressourcen teilweise dieselben

Beispiel: Masken

Zusammenfassung

- Projekte verwenden gemeinsame Ergebnisse oder Ressourcen
- Plattformunterschiede: gemeinsame Quellen, verschiedene Compiler
- Plattformunterschiede: Differenzen in Quellen
- Unterschied in Daten, Ressourcen teilweise dieselben

Wie regelt man das?

Wie behält man das im Griff?

... mit so wenig Aufwand wie möglich?

Konfigurationsmanagement

Versionsverwaltung

- Check-in Check-out
- Welche Dateien gehören in welcher Version zu welchem Produkt?
- Paralleles Arbeiten

Build- und Release-Management

- Automatische Builds (Make)
- Build Management für jeden Entwickler
- Release Management für Kompletttests und Auslieferungen
- Pflege der Abhängigkeiten

Konventionen

- alle verwenden dieselbe Entwicklungsumgebung

Struktur

- „alles ist überall gleich“

Konzepte

Archiv

- Speichert Dateien in mehreren Versionen
- Dateien werden eingecheckt und zum Bearbeiten ausgecheckt

Projekt

- Gruppiert archivierte Dateien zu logischen o. funktionalen Einheiten
- Zugriff auf das Archiv erfolgt über ein Projekt
- Eine Datei kann mehreren Projekten zugeordnet sein

Master Project Tree (mpt)

- Ein hierarchisch aufgebautes Archiv
- Enthält alle Projekte und die zu ihnen gehörenden Dateien

Sandbox

- Abbild eines Projekts aus dem mpt, identischer Verzeichnisbaum
- Individuelle Arbeitsumgebung

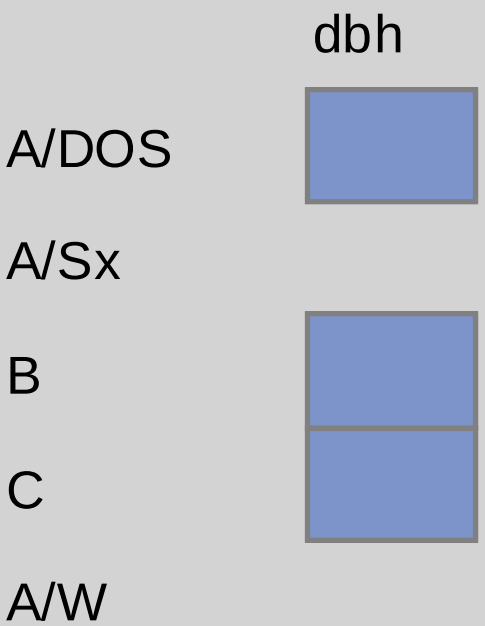
Anwendung der Konzepte

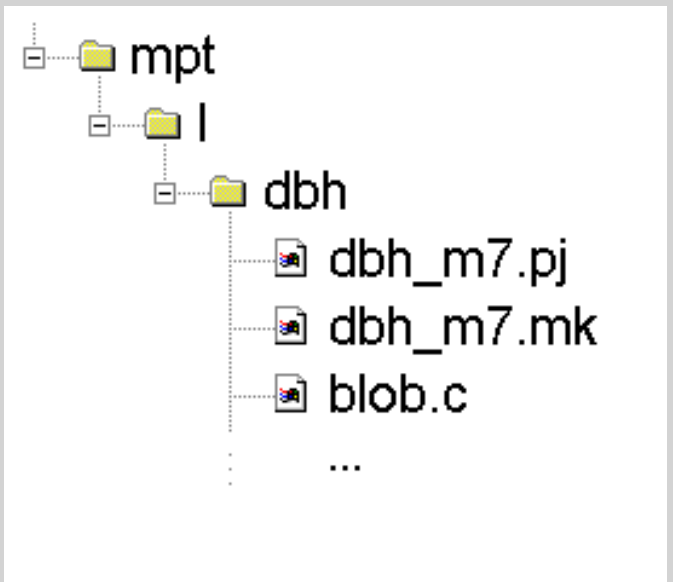
Jedes Projektverzeichnis enthält Projektfiles und Makefiles für alle plattformabhängigen Produkte, die erzeugt werden können.

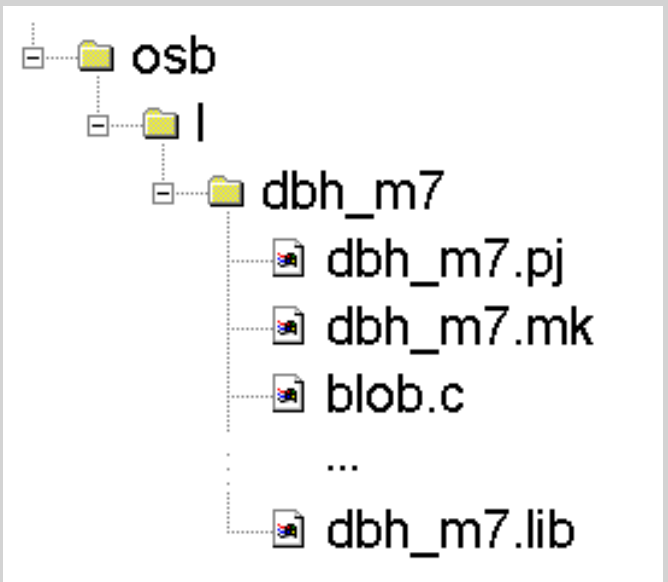
osb: offical sandbox, nur für release build

isb´s: individual sandboxes für die Entwicklung

A) Projekte verwenden gemeinsame Ressourcen oder Ergebnisse



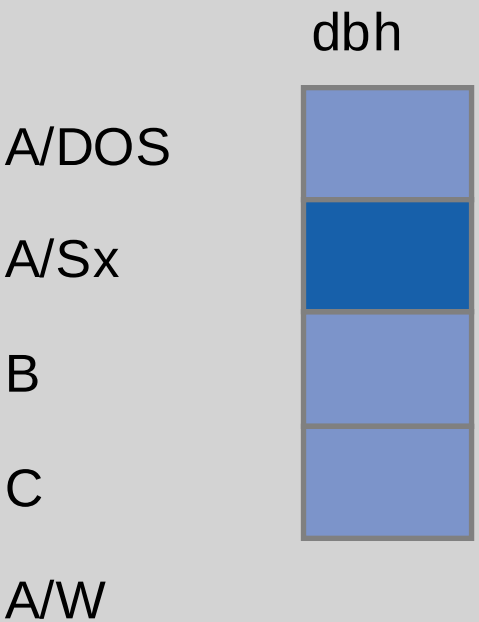


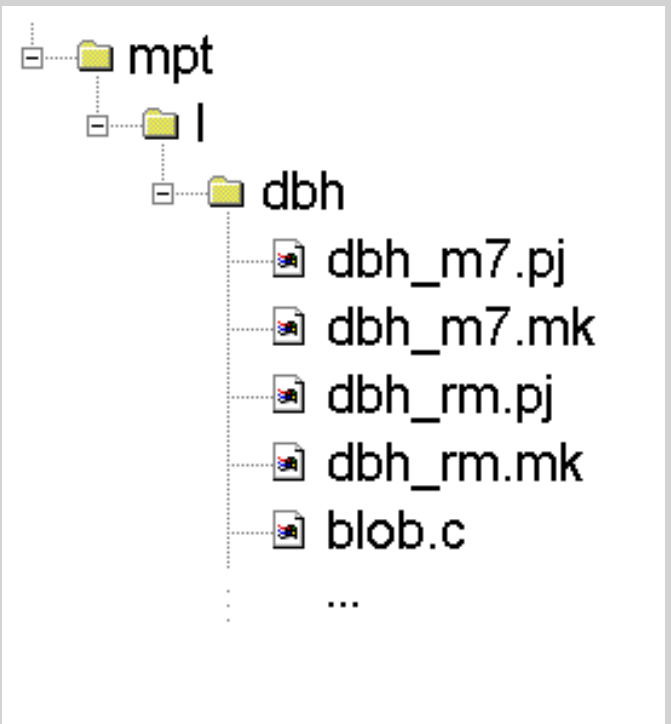


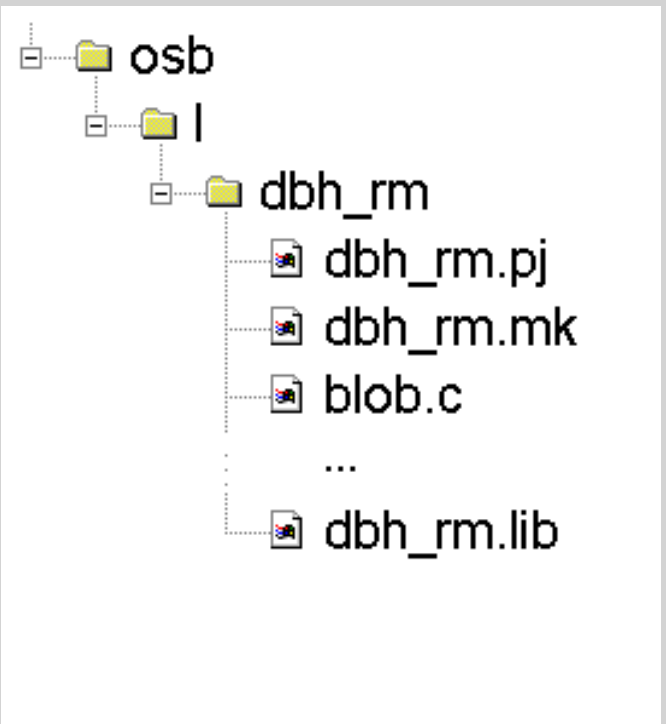
dbh_m7.pj:

p:/mpt/l/dbh/dbh_m7.pj
p:/mpt/l/dbh/dbh_m7.mk
p:/mpt/l/dbh/blob.c

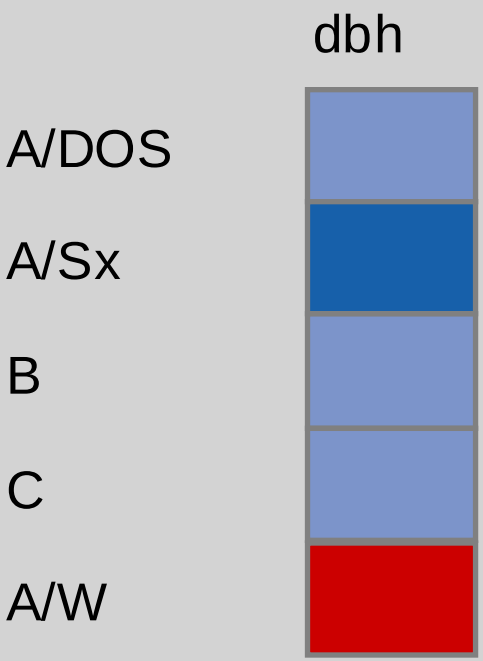
B) Plattformunterschiede: gemeinsame Quellen, verschiedene Compiler

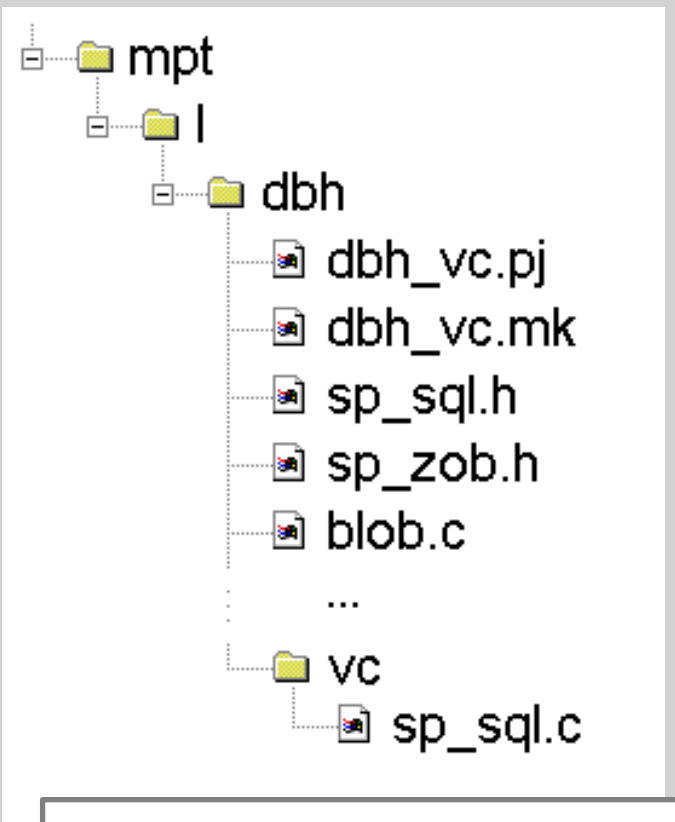


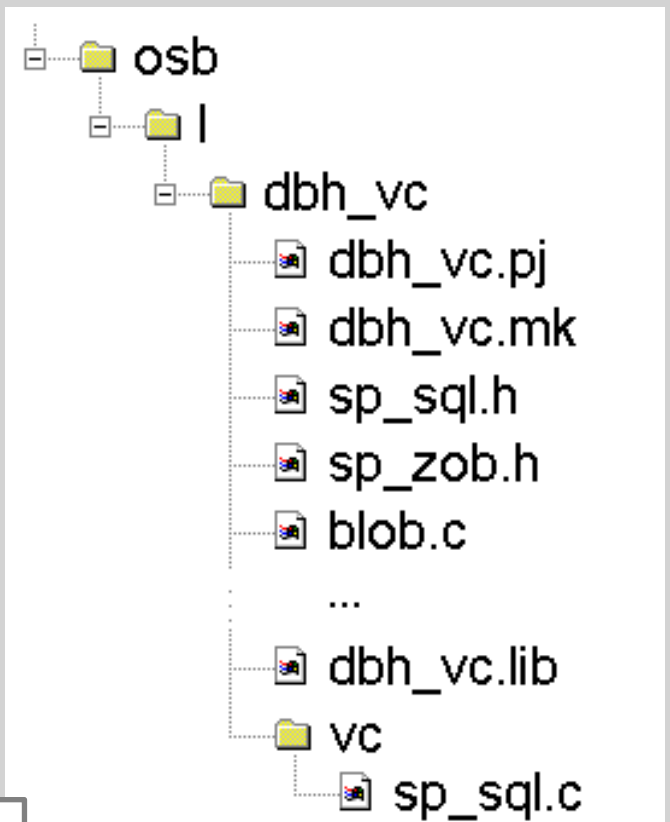




C) Plattformunterschiede: Differenzen in Quellen



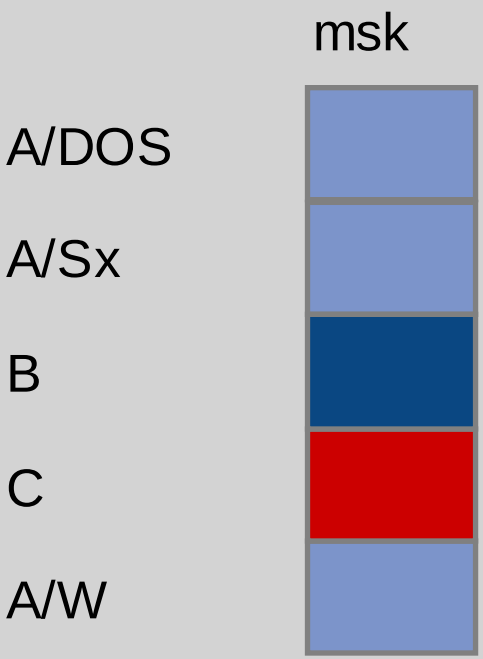


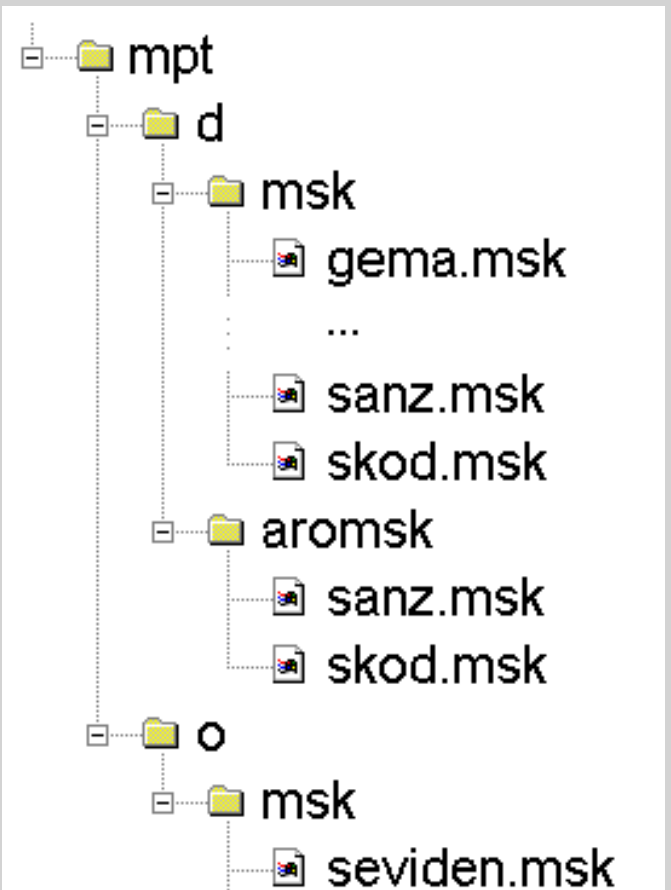


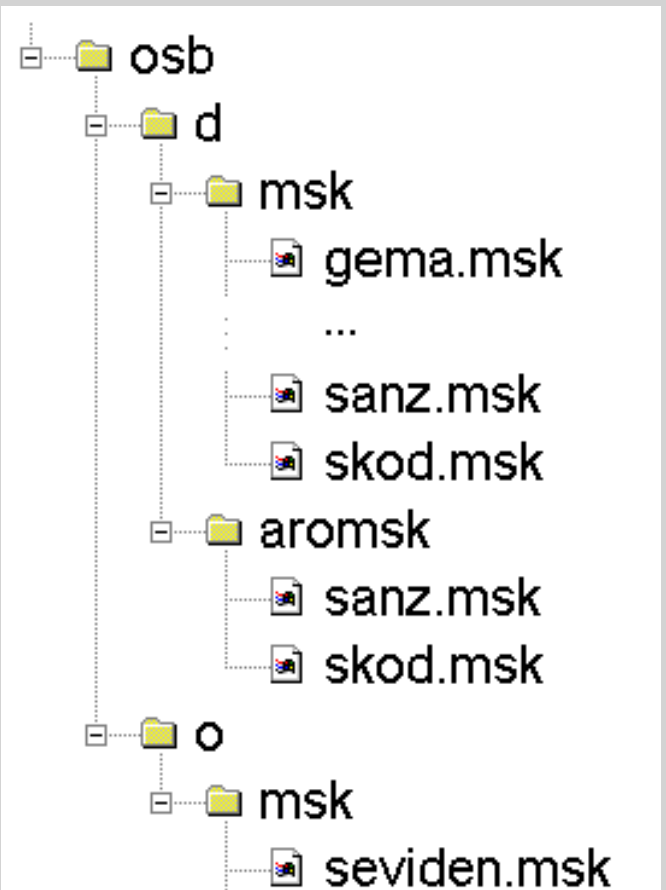
sp_zob.h:

```
#if defined( SPVC )
   #include "spsql.h"
#endif
```

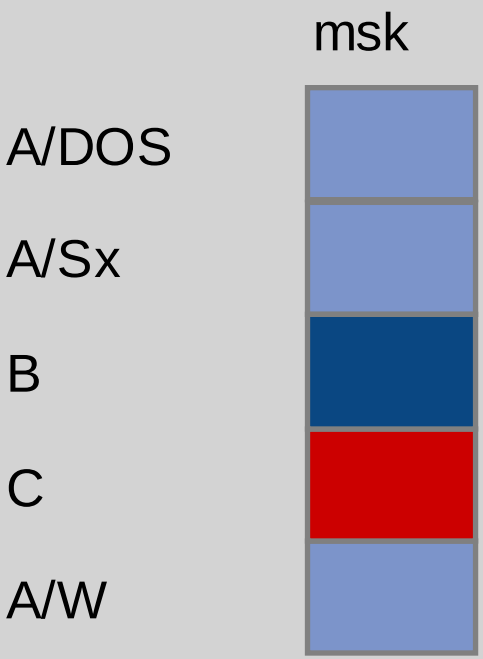
D) Unterschied in Daten, Ressourcen teilweise dieselben

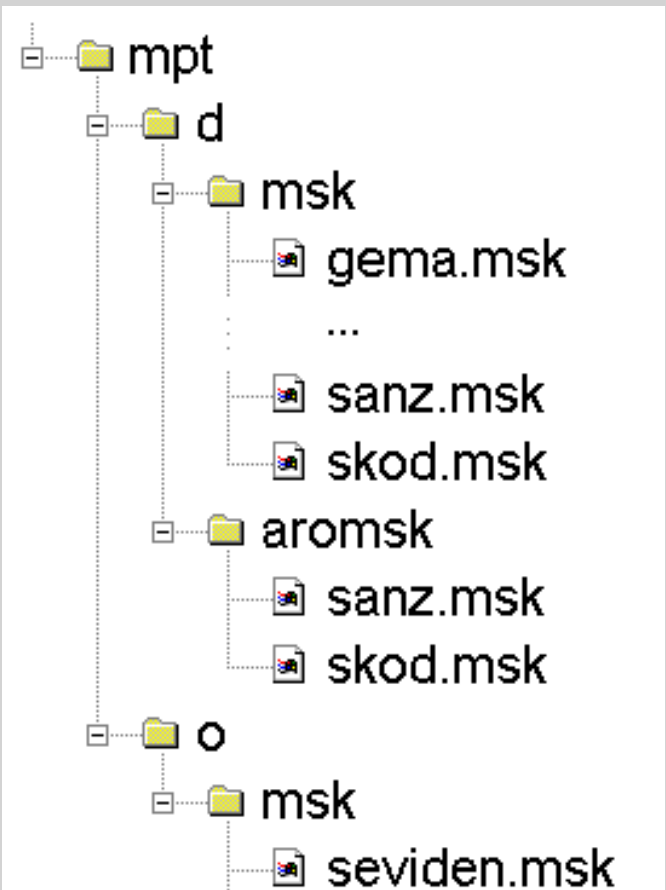






D) Unterschied in Daten, Ressourcen teilweise dieselben





msk.pj (A/DOS):

p:/mpt/d/msk/gema.msk

p:/mpt/d/msk/sanz.msk

p:/mpt/d/msk/skod.msk

msk.pj (B):

$(mpt_d_msk)/sanz.msk

p:/mpt/d/skod.msk

msk.pj (C):

p:/mpt/o/msk/seviden.msk

Auslieferung

Builds in den osb´s

- pj refresh
- make
- halbautomatisch

Export in die Auslieferungsverzeichnisse

- Abbild der Verzeichnisstruktur beim Kunden
- skriptgesteuert, halbautomatisch

Installationsprogramm

- eigenes Projekt

Media Build

- skriptgesteuert

Warum hat das funktioniert?

Architektur

- Layers + Interpreter
- Struktur durch Verzeichnisbäume
- Variabilität beherrschen

Reproduzierbare Verfahren

- pj refresh + make
- halbautomatisch + bewusst
- from scratch + verstanden

Soziale Kompetenz

- sich einig sein
- bewusst den Konventionen folgen
- keine rein technische Lösung
- Qualitätsbewusstsein

Literatur

- Kevin Jameson: Multi-Platform Code Management, O'Reilly 1994
- MKS Source Integrity User Guide, Mortice Kern Systems, Canada 1999