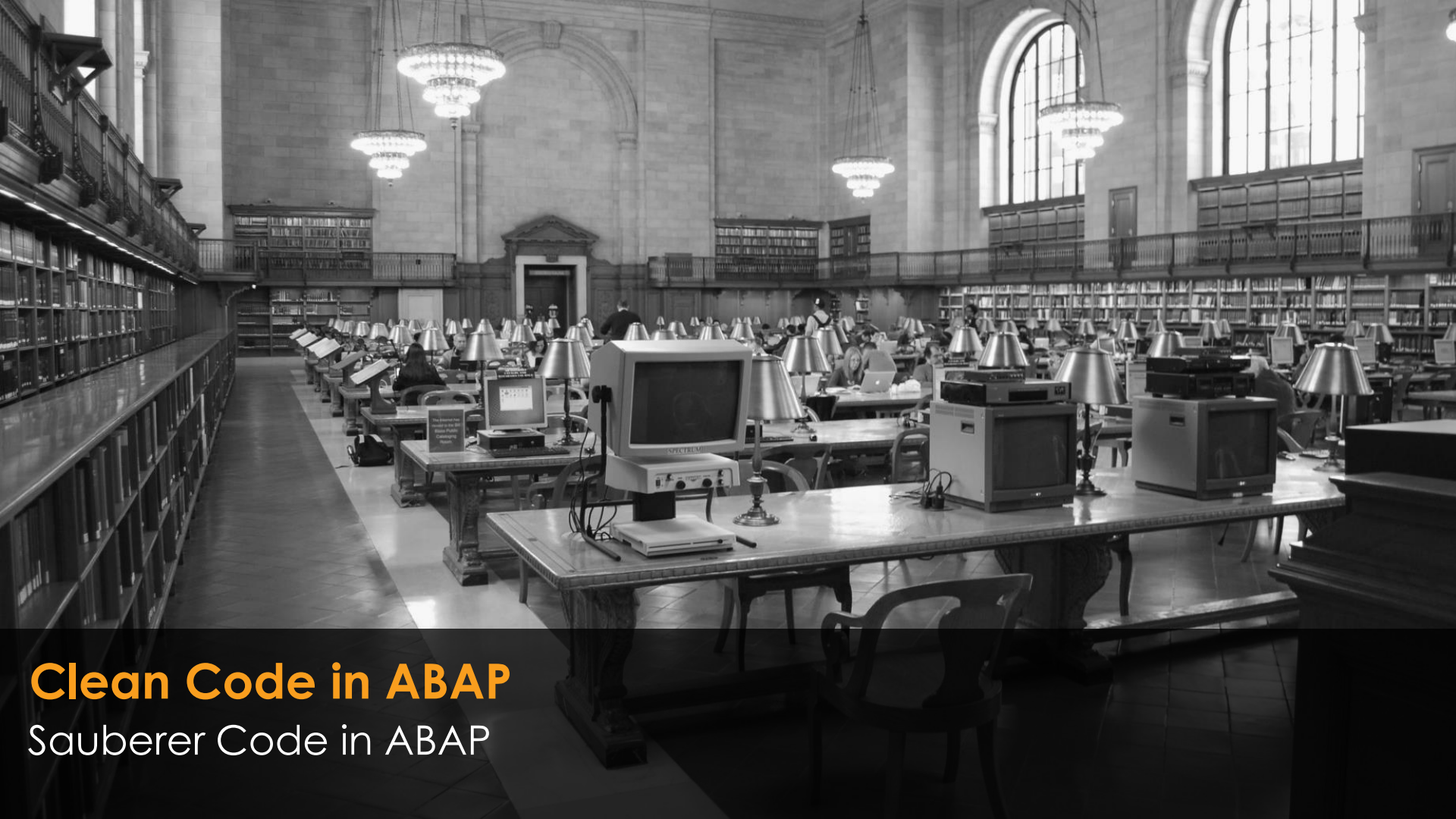


Clean Code in ABAP

Sauberer Code in ABAP

Johann Fößleitner

Senior Consultant at Cadaxo

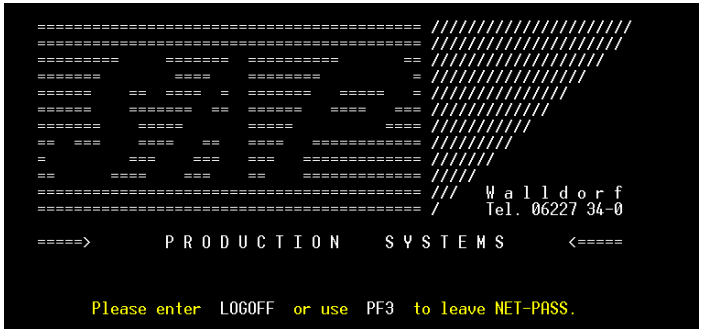
eMail: johann.foessleitner@cadaxo.com
Twitter: @foessleitnerj

Beratungsschwerpunkte

- Konzeption & Management von Entwicklungsprojekten
- Qualitymanagement & Performanceoptimierung
- Webanwendungen auf Basis von SAPUI5, BSP oder Web Dynpro for ABAP
- SAP CRM Entwicklungen (WebUI, Middleware, …)

cadaxo

6. August 1990



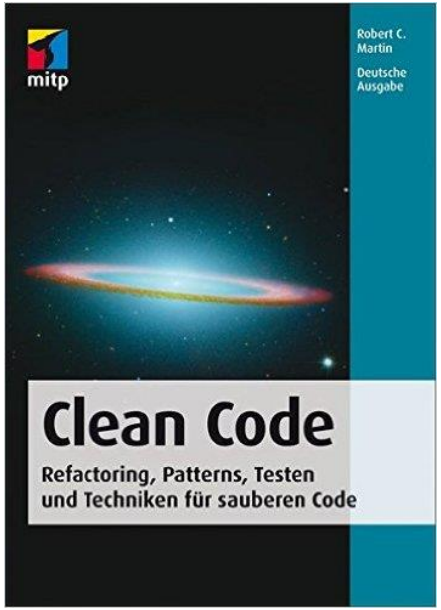
30. April 1993

- **CERN gibt WWW frei**
  - Tim Berners-Lee & Robert Cailliau haben für den internen Dokumentenaustauch kurzerhand das WWW erfunden
  - Sie definieren und entwickelten:
    - Seitenbeschreibungssprache **HTML**
    - Transferprotokoll **HTTP**
    - **URL**
    - Den **ersten Browser**
    - Den **ersten Webserver**

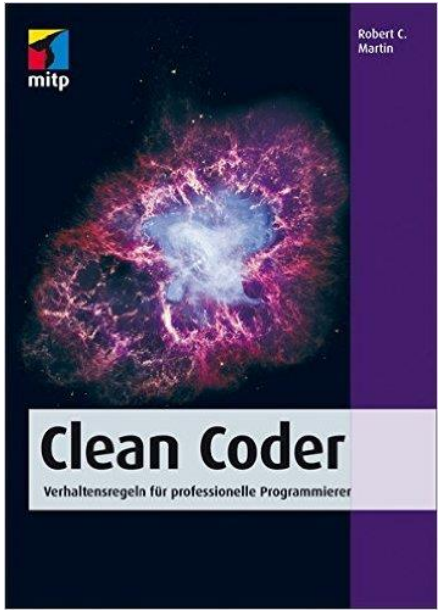
Picture von Coolcaesar aus der englischsprachigen Wikipedia, CC BY-SA 3.0, https://commons.wikimedia.org/w/index.php?curid=395096

cadaxo

Clean Code



ISBN 978-3-8266-5548-7



ISBN 978-3-8266-9695-4

Umfrage Clean Code

cadaxo

Definition von Clean Code

cadaxo

"Clean code is simple and direct. Clean code reads like well-written prose."

Grady Booch, Mitbegründer von UML

cadaxo

"Any fool can write code that a computer can understand. Good programmers write code that humans can understand."

Martin Fowler, Pionier der agilen Softwareentwicklung

cadaxo

"You know you are working on clean code when each routine you read turns out to be pretty much what you expected."

Ward Cunningham, Extreme Programming Pionier und „Erfinder" von Wiki

- Funktioniert **korrekt**
- **Lesbar** wie ein Buch
- **Einfach** und direkt
- **Effizient**
- Einfach **erweiterbar**

cadaxo

Wie immer, geht‘s um

Schlechter Code kostet Geld!

cadaxo

4. Juni 1996

- **Explosion Ariane 5**
  - 36,7 Sekunden nach dem Start
  - Ursache
    - Arithmetischer Überlauf bei der Umwandlung einer 64-Bit-Gleitkomma-Zahl in eine 16-Bit-Ganzzahl – der Wert der horizontalen Geschwindigkeit.
    - Lenksystem brach zusammen und löste eine Selbstzerstörung aus
  - Hintergrund
    - Software stammt von Ariane 4. Ariane 5 flog jedoch schneller
    - Schaden ca. 500 Millionen Dollar für Rakete und 4 Satelliten

cadaxo

TOTER CODE
Völlig unnütz

DUPLICATES
Don‘t repeat yourself

KOMMENTARE
Meist schlecht

KOMPLEXITÄT
Schwierig zu verstehen

ZU GROßE EINHEITEN
Kleiner Einheiten sind besser zu verstehen

NAMENSGEBUNG
Zweckbeschreibende Namen wählen

- **Verständlicher Code ist enorm wichtig**
  - Überwiegende Kosten für Software sind Wartungskosten
  - Code wird daher überwiegend gelesen
  - Je unverständlicher der Code, desto größer die Gefahr, dass er nicht ausreichend verstanden wird

Warum erstellen wir überhaupt schlechten Code?

- **Namensgebung**
  - Aussagekräftige Namen für Felder, Methoden, Klassen, …
  - Lange Namen sind nicht schlecht
  - Namen sollen den Zweck beschreiben (GET_PARTNER)
  - Verwendet aussprechbare & suchbare Namen
  - Und bitte, einheitlich in Englisch
    - **MARA-MTART** oder **BSEG-DMBTR** sind keine Vorbilder!

- **Beispiel: Funktionsname**
  - Was könnte bei folgendem Code ausgeführt werden
    - LR_DATE->ADD(5).
  - Besser wäre:
    - LR_DATE->ADDDAYSTO(5) (addiere 5 Tage)
    - LR_DATE->INCREASEBYDAYS(5) (addiere 5 Tage)
    - LR_DATE->DAYSSINCE (5) (gib datum + 5 Tage zurück)

- **Funktionen**
  - KISS – Keep it simple and stupid
  - Funktionen sollten genau nur eine Aufgabe erledigen
  - Minimale Funktionsargumente (Parameter)
    - Keine FLAGs als Parameter verwenden
  - Einheitlich klassenbasierte Ausnahmen verwenden

- **Kommentare**
  - Niemals kommentieren, was der Code macht
  - Kommentieren warum es so gelöst wurde
    - Hintergrundinfos
  - Auskommentiertes Coding entfernen
    - Jedes SAP System verfügt über eine Versionsverwaltung
  - Keine redundanten Kommentare
  - Keine irreführenden Kommentare

```
PERFORM BUILD_QUERYTAB IN PROGRAM SAPLADCE
   TABLES QUERYPARAMTAB
   USING 'EN'    "only a dummy caused by this bullshit API
         RESULT_FROM
         RESULT_TO.
```

```
DELETE NPLAN INDEX SY-TABIX.
" we have to delete the whole fucky bunch of volumes where we
" were able to insert before (life is terrible)
```

```
* This is terrible basis function module. :-((
```

```
* 24.01.1989 mit    First release
```

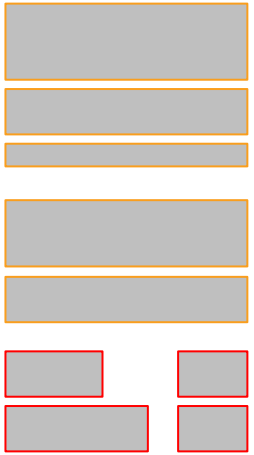
- **Beispiel: Redundante Kommentare**
  - Oft werden redundante Kommentare eingefügt
  - Wenn der Code sich selbst ausreichend erklärt, ist ein edundante Kommentar nur überflüssig, verwirrend, ...

```
* Lesen aller Felder aus der BUT000
SELECT * FROM BUT000 INTO …

* Verlasse Routine, wenn SY-SUBRC > 0
IF SY-SUBRC > 0.
   EXIT.
ENDIF.
```

- **Beispiel: Irreführende Kommentare**
  - Irreführende Kommentare bzw. subtile Fehlinformationen sind schwerer zu lesen als der Code selbst
  - Unbedingt vermeiden

```
* Lesen aller Felder aus der BUT000
SELECT PARTNER, NAME1 FROM BUT000 INTO …

* Verlasse Routine, wenn Fehler oder flag = true.
IF SY-SUBRC > 0 AND FLAG = ABAP_TRUE.
   EXIT.
ENDIF.
```

- **Formatierung**
  - Code muss sauber formatiert sein
  - Was zusammengehört, vertikal zusammenfassen
  - Keine horizontale Ausrichtung
  - Ein unsauber formatierter Code deutet darauf hin, dass der eigentliche Code ebenfalls unsauber ist.



- **Klassenbasierte Ausnahmen sind besser als Fehlercodes**
  - Fehlercodes führen zu tief verschachtelten Strukturen, der Aufrufer muss den Fehler sofort behandeln
  - Der Verwendung von Ausnahmen ermöglicht die Trennung von der Fehlerverarbeitung und dem normalen Coding
  - Catch Bereiche in eigene Methoden auslagern

Erste Anweisung

Ausnahmebehandlung
In eigener Methode

Letzte Anweisung

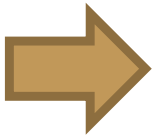
```
METHOD main.

  TRY.

      do_something( ).

      do_more( ).

      do_the_last_thing( ).

    CATCH zcx_an_exception INTO DATA(lr_exception).

      handle_exception( lr_exception ).

  ENDTRY.

ENDMETHOD.
```

- **Klassen & Objekte**
  - SRP - Single-Responsibility-Prinzip – Eine Verantwortlichkeit je Klasse
  - Klein, Klein, Klein – keine Gottklassen erschaffen
  - Objektorientiert entwickeln und damit Erweiterungen ermöglichen
    - Es gibt auch Instanzen, Interfaces, Konstruktoren, …

cadaxo

Was hindert uns daran schlechten Code zu verbessern?

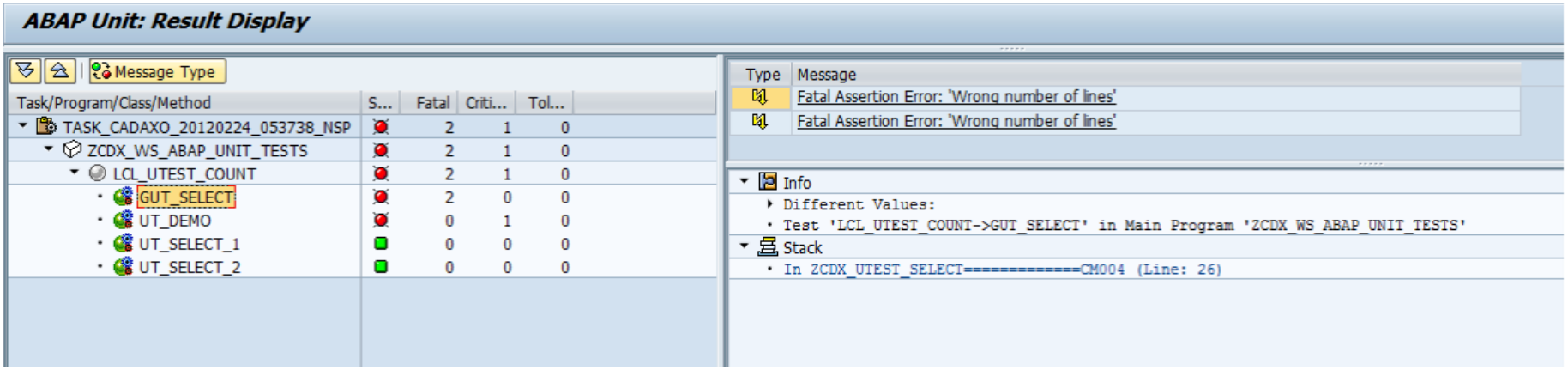
- **Unit Tests**
  - Mit Unit Tests werden einzelne Units (Methoden, Funktionbausteine) automatisiert getestet.
  - Ziel der Unit Tests ist die Isolation aller Parts eines Programmes und die Prüfung ob alles ordnungsgemäß läuft.

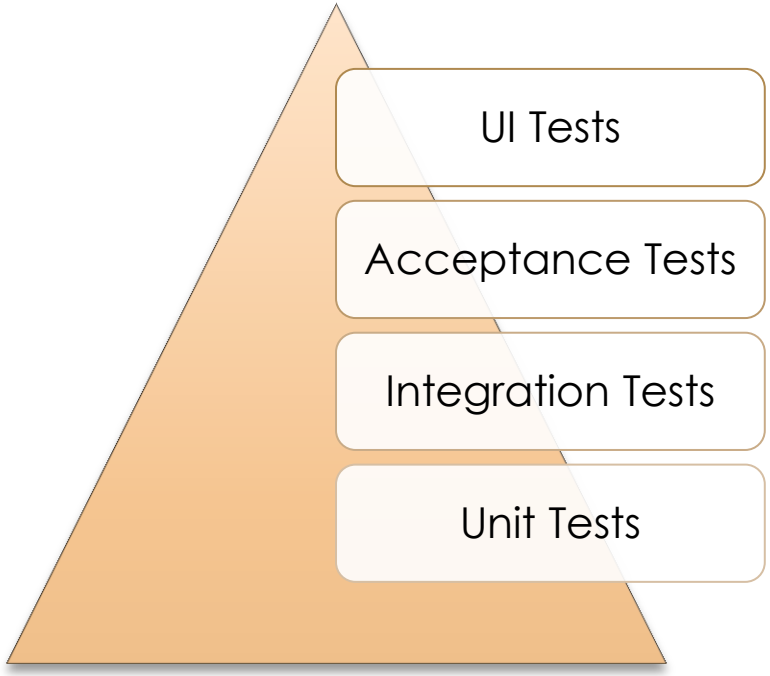


ABAP Unit Tests

- **Unit Tests** – FIRST Prinzip
  - **F**ast – Tests sollen schnell sein
  - **I**ndependent – Tests sollen nicht von einander abhängen
  - **R**epeatable – Jede Umgebung (Dev, QA, Prod, …)
  - **S**elf-Validating – der Test wird bestanden oder er scheitert
  - **T**imely – Tests müssen rechtzeitig geschrieben werden

cadaxo


ABAP Unit: Result Display
Message Type
Task/Program/Class/Method
S...
Fatal
Criti...
Tol...
TASK_CADAXO_20120224_053738_NSP 2 1 0
ZCDX_WS_ABAP_UNIT_TESTS 2 1 0
LCL_UTEST_COUNT 2 1 0
GUT_SELECT 2 0 0
UT_DEMO 0 1 0
UT_SELECT_1 0 0 0
UT_SELECT_2 0 0 0
Type
Message
Fatal Assertion Error: 'Wrong number of lines'
Fatal Assertion Error: 'Wrong number of lines'
Info
Different Values:
Test 'LCL_UTEST_COUNT->GUT_SELECT' in Main Program 'ZCDX_WS_ABAP_UNIT_TESTS'
Stack
In ZCDX_UTEST_SELECT==========CM004 (Line: 26)



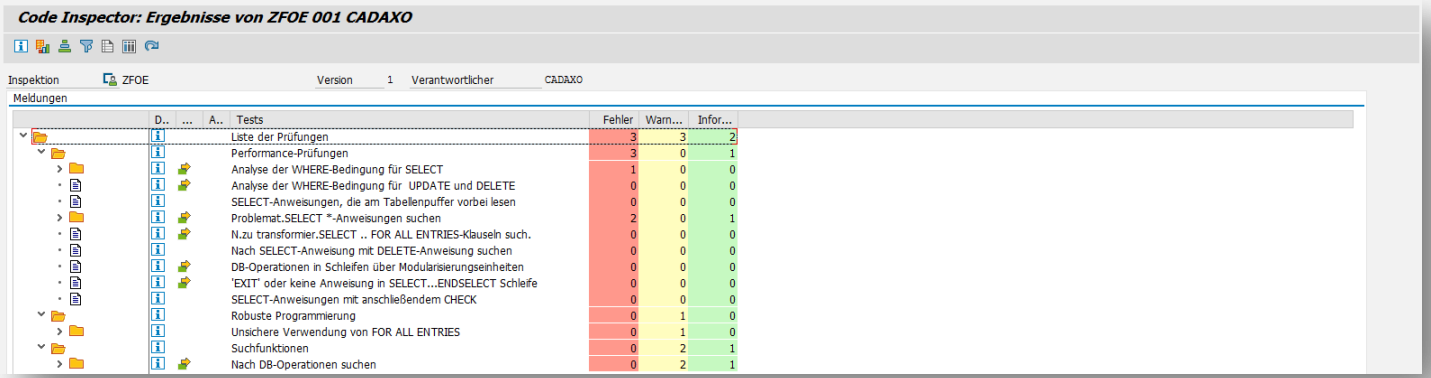
eCATT

ABAP Unit

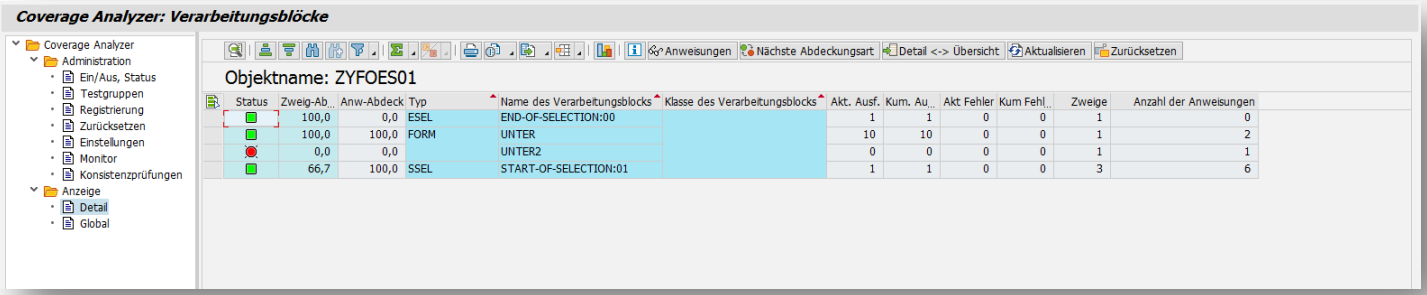
https://www.eventbrite.com/e/tdd-mit-abap-units-tickets-27079167519

Tools im SAP bzw. ABAP

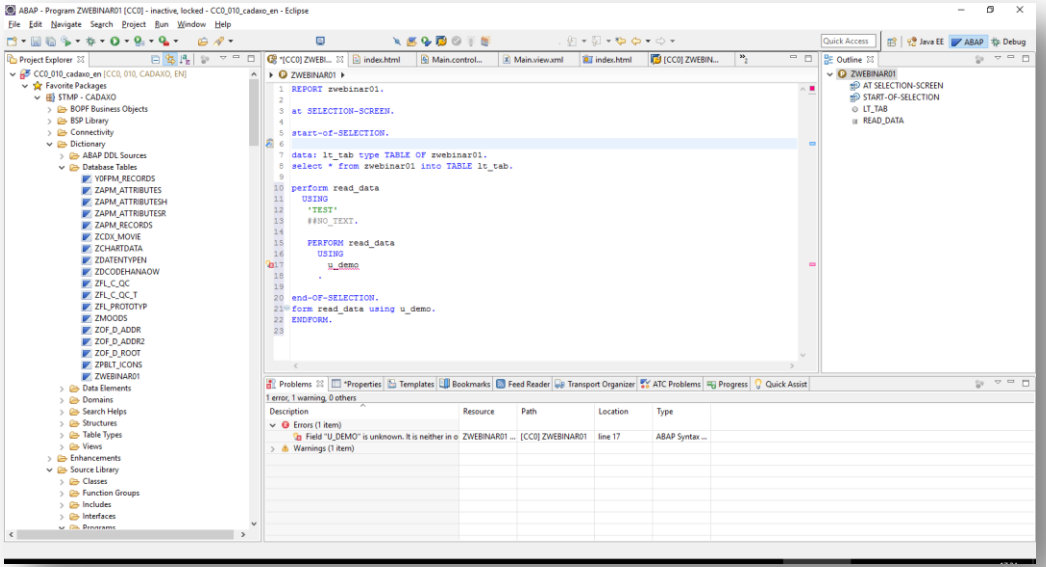
- **Der Code Inspector führt verschiedene statische Prüfungen durch**
  - Es können einzelne Objekte oder viele Objekte (ein Paket, Z*, …) geprüft werden
  - Geprüft wird beispielsweise:
    - Performancekritische Statements, „totes“ Coding, fehlende Indices bei Datenbankzugriffen, Namenskonventionen, …
  - Meldungen können mit Pseudokommentaren (z.B. #EC CL_BYPASS) unterdrückt werden.
  - Transaktionscode: **`SCI`**



- **Der Coverage Analyzer ist ein Tool, mit dem die Verarbeitung von ABAPs systemweit analysiert werden kann.**
  - Ermittlung von Programmteilen welche selten bzw. nie aufgerufen werden
  - Ermittlung von Programmteilen welche sehr oft aufgerufen werden und sich daher für performanceverbessernde Aktivitäten eignen könnten
  - Transaktionscode: **scov**



- **ABAP in Eclipse**
  - Bietet mehr Möglichkeiten im Bereich Refactoring als SE80

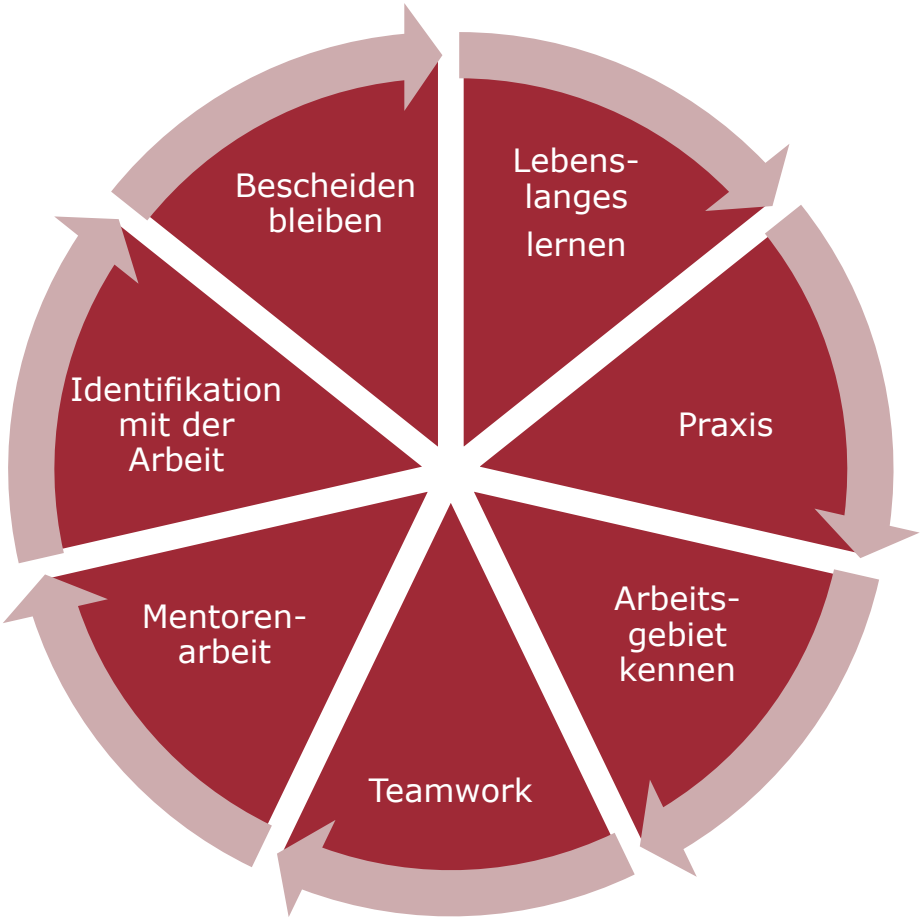


- **Wie schreibt man sauberen Code**
  - Zuerst notieren wir unsere Gedanken, ordnen sie bis sie gut lesbar sind, bis sie logisch und verständlich sind
  - Dann schreiben wir die Funktion. Zuerst meist eine lange Funktion mit vielen verschachtelten Schleifen, schlechten Namen, Duplizierungen, etc. um eine schnelles Ergebnis zu haben
  - Und schließlich verfeinern wir den Code, lagern Funktionen aus, ändern die Namen, eliminieren Duplizierungen
  - Wir säubern unseren Code!

cadaxo

Professionell sein

cadaxo


Bescheiden bleiben
Lebens-langes lernen
Praxis
Arbeits-gebiet kennen
Teamwork
Mentoren-arbeit
Identifikation mit der Arbeit

cadaxo

Adobe Document Services
ABAP for Eclipse
Floorplan Manager
Switch Framework
BRF/BRF+
SAPUI5
JQuery
CSS
Web IDE
SAP Gateway
HTML5
Web Dynpro for ABAP
Open SQL
WebServices
Javascript
BADIs
ABAP
BSP
JSON / XML
ABAP SQL Expressions
CDS Views
SQL Monitor
SAP HANA
OData
BOR
Laufzeitanalyse
BOPF
Code Inspector
AMDP
SQL Script
ABAP Channels
ABAP Expressions
Regex
Shared Objects

Praktizieren & Üben

- **Übung macht den Meister**
  - Damit ein Musiker bei einem Konzert begeistern kann, **muss er üben**.
  - Damit ein Sportler erfolgreich ist, **muss er üben**.
  - Damit ein Softwareentwickler perfekte Lösungen abliefert, **muss er üben**.
    - Die geübten Skills sollten außerhalb des normalen Jobs sein
    - Skills sollen verfeinert bzw. erweitert werden
    - Es geht schlichtweg darum, das Gehirn zu trainieren
    - Beispiele in ABAP
      - Dec/Bin/Hex Umrechnung; Quicksort ausprogrammieren; eMail Validierung mit Regex, …

cadaxo

DOJO

KATA
WAZA
RANDORI

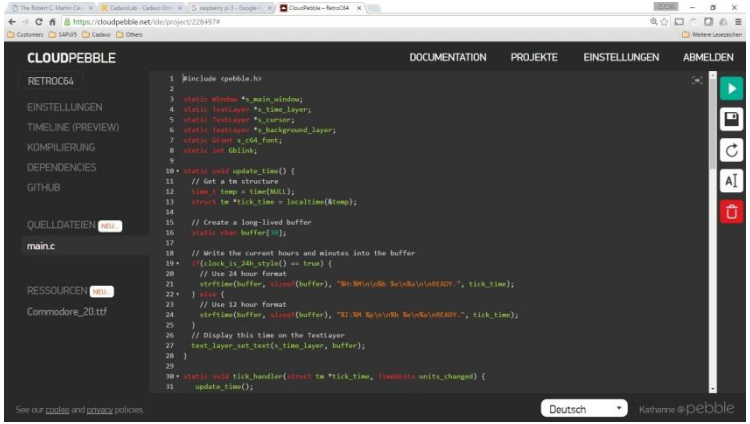
Picture by Wang Ming [GFDL (http://www.gnu.org/copyleft/fdl.html) or CC-BY-SA-3.0 (http://creativecommons.org/licenses/by-sa/3.0/)], via Wikimedia Commons from Wikimedia Commons

- **Kata – Die Perfektion ist das Ziel**
  - Ein Programmier-Kata ist eine festgelegte Gruppe von choreographierten Tasteneingaben/Mausbewegungen
  - Man löst nicht wirklich ein Problem, man kennt die Lösung bereits
  - Man übt die Bewegungen und Entscheidungen die zur Lösung gehören
    - http://codekata.com/
    - http://katas.softwarecraftsmanship.org/

- **Waza – Paarweise üben**
  - Entwickler A schreibt einen ABAP Unit Test
  - Entwickler B implementiert die Methode, bis der Test bestanden wird
  - Entwickler A kann für den Test z.B. auch min. Geschwindigkeit oder den max. Speicherverbrauch vorgeben.

- **Randori – Übungen für‘s Team**
  - Editor wird an die Wand projiziert
  - Entwickler A schreibt einen ABAP Unit Test
  - Entwickler B implementiert die Methode, bis der Test bestanden wird
  - Entwickler B schreibt einen ABAP Unit Test
  - Entwickler C implementiert die Methode, bis der Test bestanden wird
  - …

cadaxo


CLOUDPEBBLE
DOCUMENTATION
PROJEKTE
EINSTELLUNGEN
ABMELDEN
RETROC64
EINSTELLUNGEN
TIMELINE (PREVIEW)
KOMPILIERUNG
DEPENDENCIES
GITHUB
QUELLDATEIEN
main.c
RESSOURCEN
Commodore_20.ttf
Deutsch

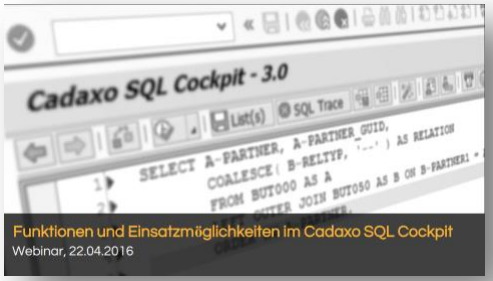

2. Pin Details
Please enter the message text, the title and the subtitle
Title:
Title of the message
Subtitle:
A subtitle
*Message:
The body...
*Icon:
NOTIFICATION_
Weiter
Cancel


05
September
2014
06
Oktober
2015
07
November
2016
12
11
OK
Abbr.

cadaxo

- **Ethisch handeln**
  - Professionelle Entwickler üben in ihrer Freizeit
  - Auf die eine oder andere Weise üben und praktizieren alle Profis
    - Ärzte, Musiker, Sportler, …
  - Da wir in der Freizeit üben, können wir auch in anderen Programmiersprachen üben
  - **Es liegt in unserer eigenen Verantwortung unsere Skills zu schärfen, es ist nicht die Verantwortung unserer Auftrag- bzw. Arbeitgeber**

https://www.eventbrite.com/e/tdd-mit-abap-units-tickets-27079167519



http://www.cadaxo.com/blog/

See you again!

Thank you for participating!

https://twitter.com/foessleitnerj

https://www.linkedin.com/in/johann-fößleitner-a9851b2a

https://www.xing.com/profile/johann_foessleitner

johann.foessleitner@cadaxo.com

If you want to stay in touch …

cadaxo

Cadaxo GmbH
Stubenring 18/5a | 1010 Vienna, Austria

office@cadaxo.com
www.cadaxo.com

https://www.linkedin.com/company/cadaxo-gmbh
https://www.xing.com/companies/cadaxogmbh
https://www.facebook.com/CadaxoGmbH
https://twitter.com/cadaxo
http://com.slideshare.net/cadaxogmbh
http://www.youtube.com/CadaxoGmbH