Softwarequalität – Eine Einführung

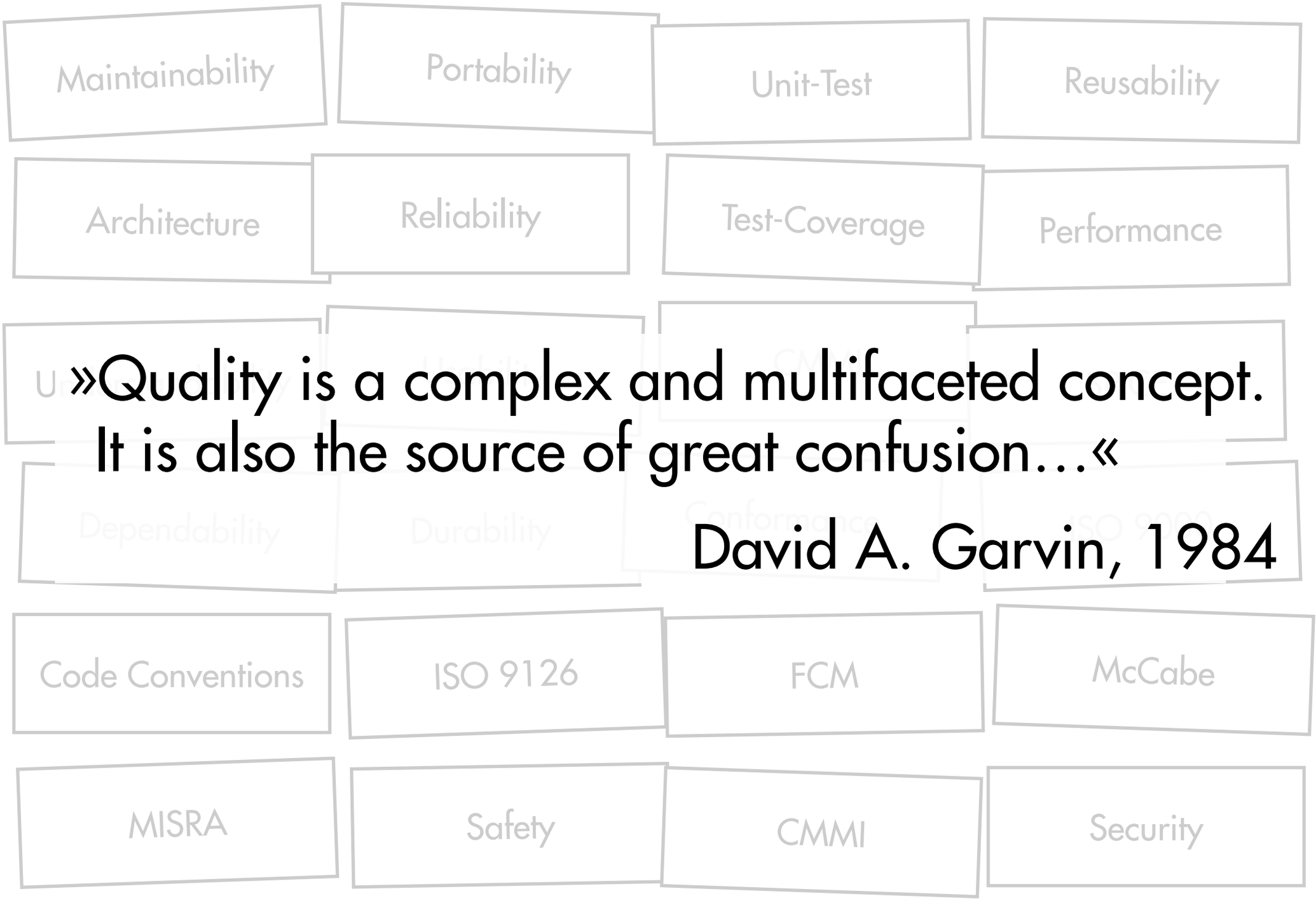
Dr. Florian Deißenböck

Seminar Softwarequalität
4. April 2013

CQSE
Continuous Quality in Software Engineering

Agenda

- Softwarequalität
- Softwarequalitätsmodelle
- Software-Wartung
- »Schlechter Code«
- Qualitätsverfall
- Qualitäts-Controlling
- Werkzeugunterstützung


Maintainability
Portability
Unit-Test
Reusability
Architecture
Reliability
Test-Coverage
Performance
»Quality is a complex and multifaceted concept.
It is also the source of great confusion…«
David A. Garvin, 1984
Code Conventions
ISO 9126
FCM
McCabe
MISRA
Safety
CMMI
Security

Can Clean Pipes Produce Dirty Water?

New views of mature ideas on software quality and productivity.

This issue, Jeff Voas looks at how much we can rely on process quality to assure product quality.
—Shari Lawence Pfleeger

OF COURSE THEY CAN. CLEAN PIPES CAN break, they can be attached to the wrong source, or the original water source may infuse dirty water into the pipeline. The complementary question is "Can dirty pipes produce clean water?" Once again the answer is yes, but this result is much less likely.

The analogy between water flowing from pipes and software flowing from process standards is quite simple. Consider the original set of requirements to be the original water source, and each successive process applied to the developing software is the next link in the pipe. In this analogy, each link in the pipe is a process, and each process is either geared toward developing software or validating it. Certain processes may be reapplied during development, and hence our pipeline may contain distributed, redundant links. Eventually something will exit from the

require you to perform certain product assessment methods, but these standards are more concerned with team infrastructure and interpersonnel communication than with rigorous product measures.

The process movement has given the perception that clean pipes can produce only clean water.

Today's software process movement is a logical advancement that evolved from the unstructured, ad hoc software engineering methods used in the 1970s and early 1980s. The cry for systematic, repeatable methods that would engender more reliable software was natural given the state of the practice then. This call has been answered by dozens of different software development standards, and even by manufacturing standards that have been supposedly refur-

Can Clean Pipes Produce Dirty Water?

New views of mature ideas on software quality and productivity

This issue, Jeff Voas looks at how much we can rely on process quality to assure product quality.

—Shari Lawrence Pfleeger

OF COURSE THEY CAN. CLEAN PIPES CAN break, they can be attached to the wrong source, or the original water source may infuse dirty water into the pipeline. The complementary question is "Can dirty pipes produce clean water?" Once again the answer is yes, but this result is much less likely.

The analogy between water flowing from pipes and software flowing from process standards is quite simple. Consider the original set of requirements to be the original water source, and each successive process applied to the developing software is the next link in the pipe. In this analogy, each link in the pipe is a process, and each process is either geared toward developing software or validating it. Certain processes may be reapplied during development, and hence our pipeline may contain distributed, redundant links. Eventually something will exit from the

require you to perform certain product assessment methods, but these standards are more concerned with team infrastructure and interpersonnel communication than with rigorous product measures.

The process movement has given the perception that clean pipes can produce only clean water.

Today's software process movement is a logical advancement that evolved from the unstructured, ad hoc software engineering methods used in the 1970s and early 1980s. The cry for systematic, repeatable methods that would engender more reliable software was natural given the state of the practice then. This call has been answered by dozens of different software development standards, and even by manufacturing standards that have been supposedly refur-

CMM Level	Min	Avg	Max
1	0,150	0,750	4,500
2	0,120	0,624	3,600
3	0,075	0,473	2,250
4	0,023	0,228	1,200
5	0,002	0,105	0,500

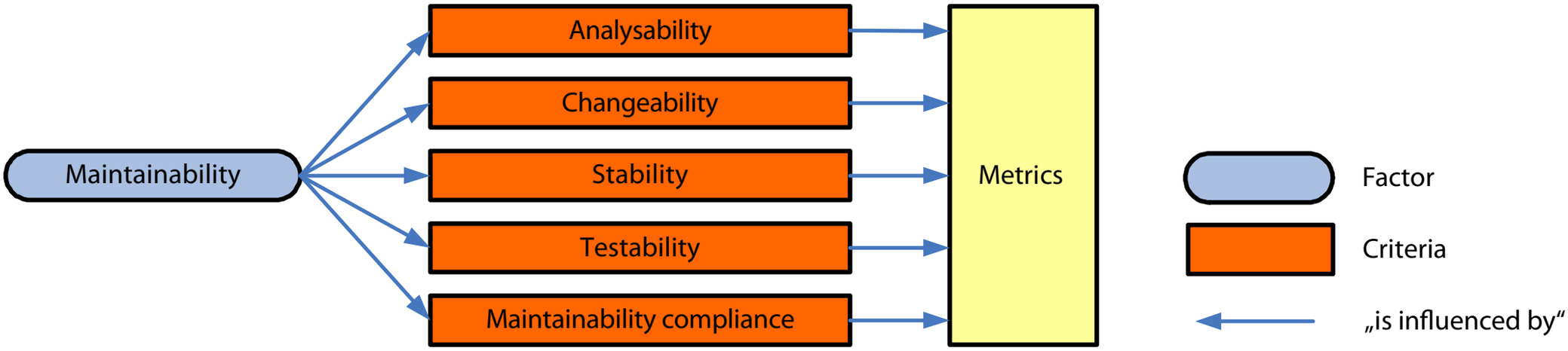
Fehler pro Function Point (Jones 2003)

Qualität – Sichtweisen

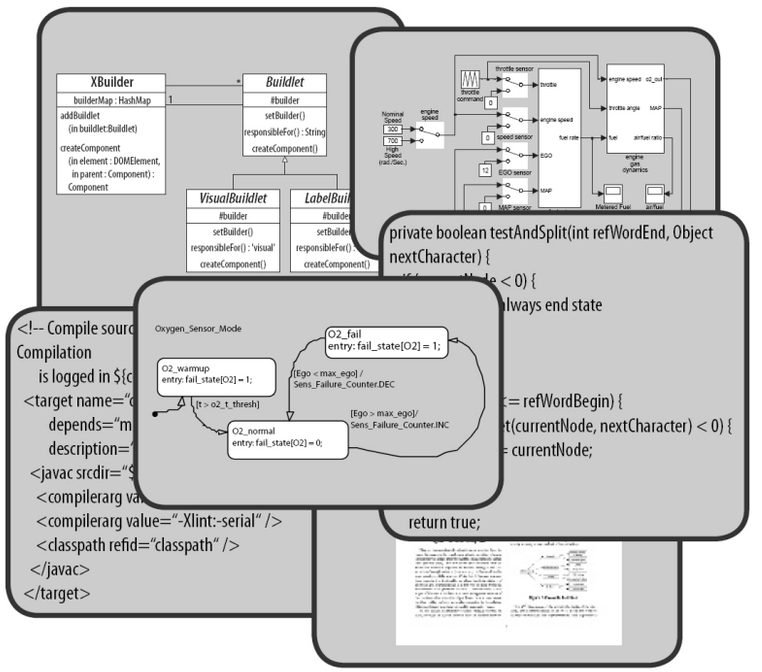
- **Transzendenter Ansatz**
  Qualität ist immanente Güte, sie ist erkennbar aber nicht definierbar
- **Produktorientierter Ansatz**
  Qualitäts-Unterschiede zwischen Produkten spiegeln sich in der unterschiedlichen Ausprägung von Produkt-Attributen wieder
- **Benutzerorientierter Ansatz**
  »Qualität liegt im Auge des Betrachters«
- **Herstellungsorientierter Ansatz**
  Qualität ist definiert durch die Erfüllung der Anforderungen bzw. durch Abweichungen von der Spezifikation
- **Wertorientierter Ansatz**
  Qualität als Tradeoff zwischen Kosten und Nutzen

* David A. Garvin, What does »Product Quality« Really Mean?, 1984


Functionality
Portability
Reliability
ISO 9126
Maintainability
Usability
Efficiency

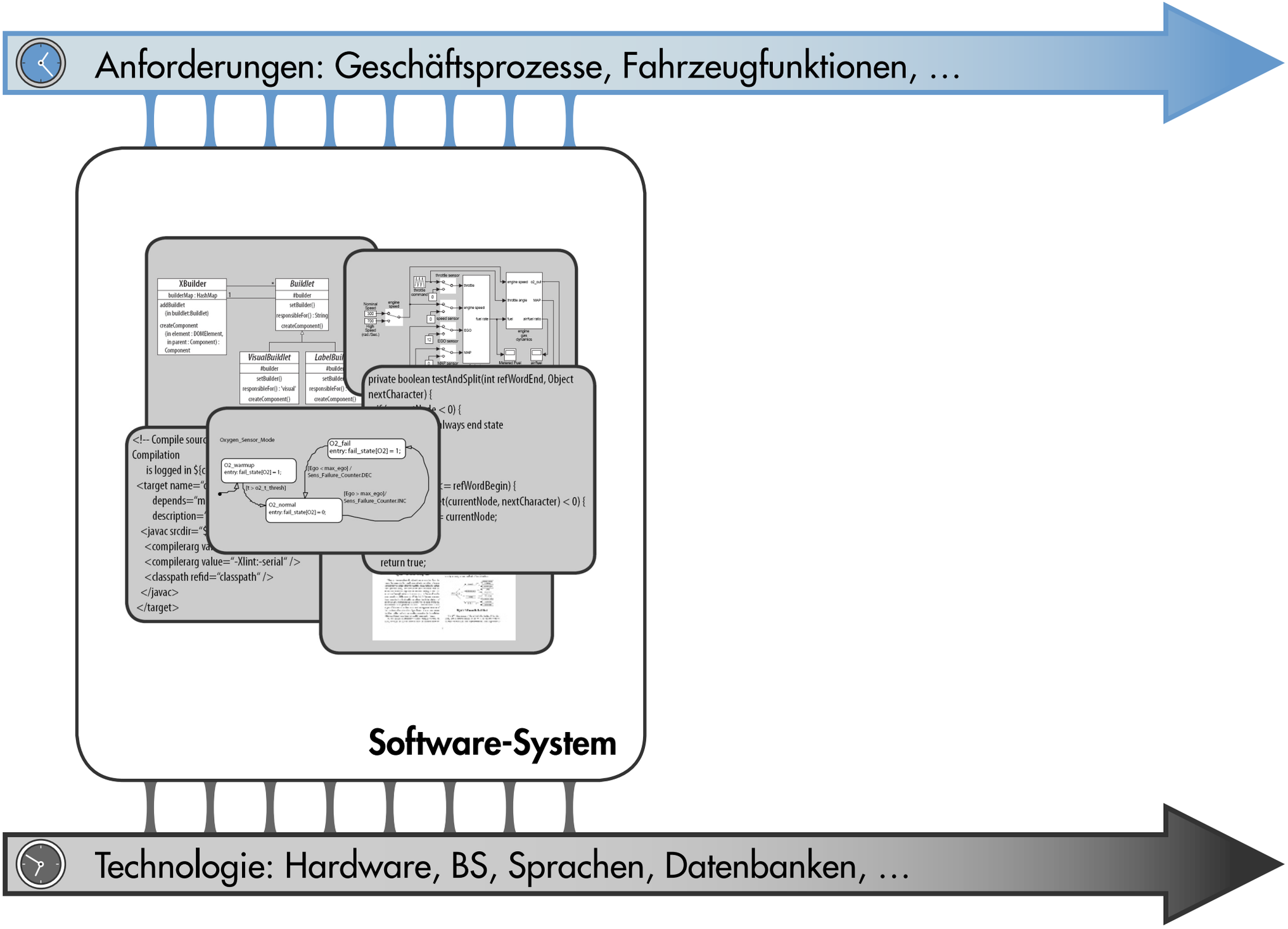

Maintainability
Analysability
Changeability
Stability
Testability
Maintainability compliance
Metrics
Factor
Criteria
„is influenced by“

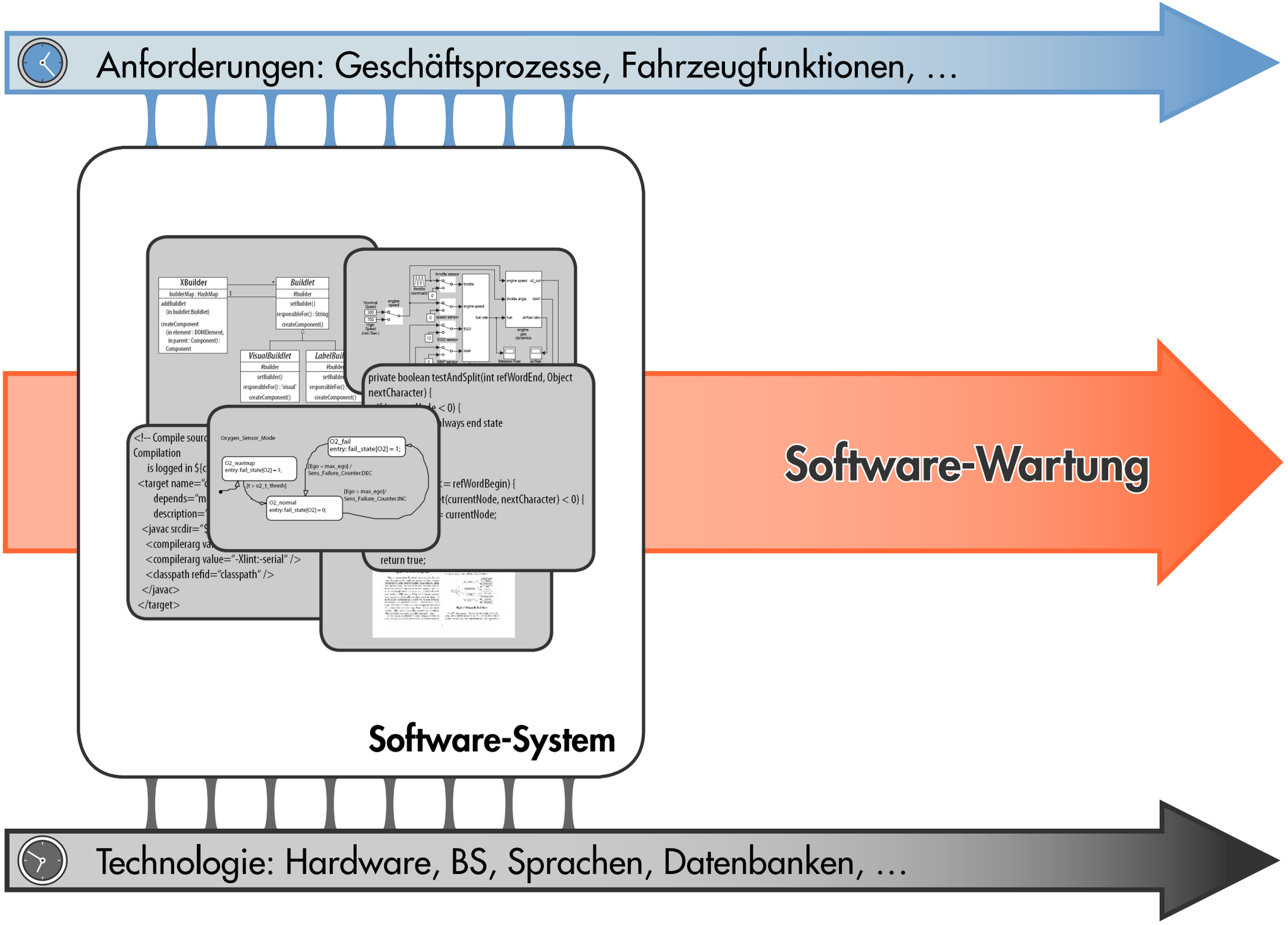
Anforderungen: Geschäftsprozesse, Fahrzeugfunktionen, …

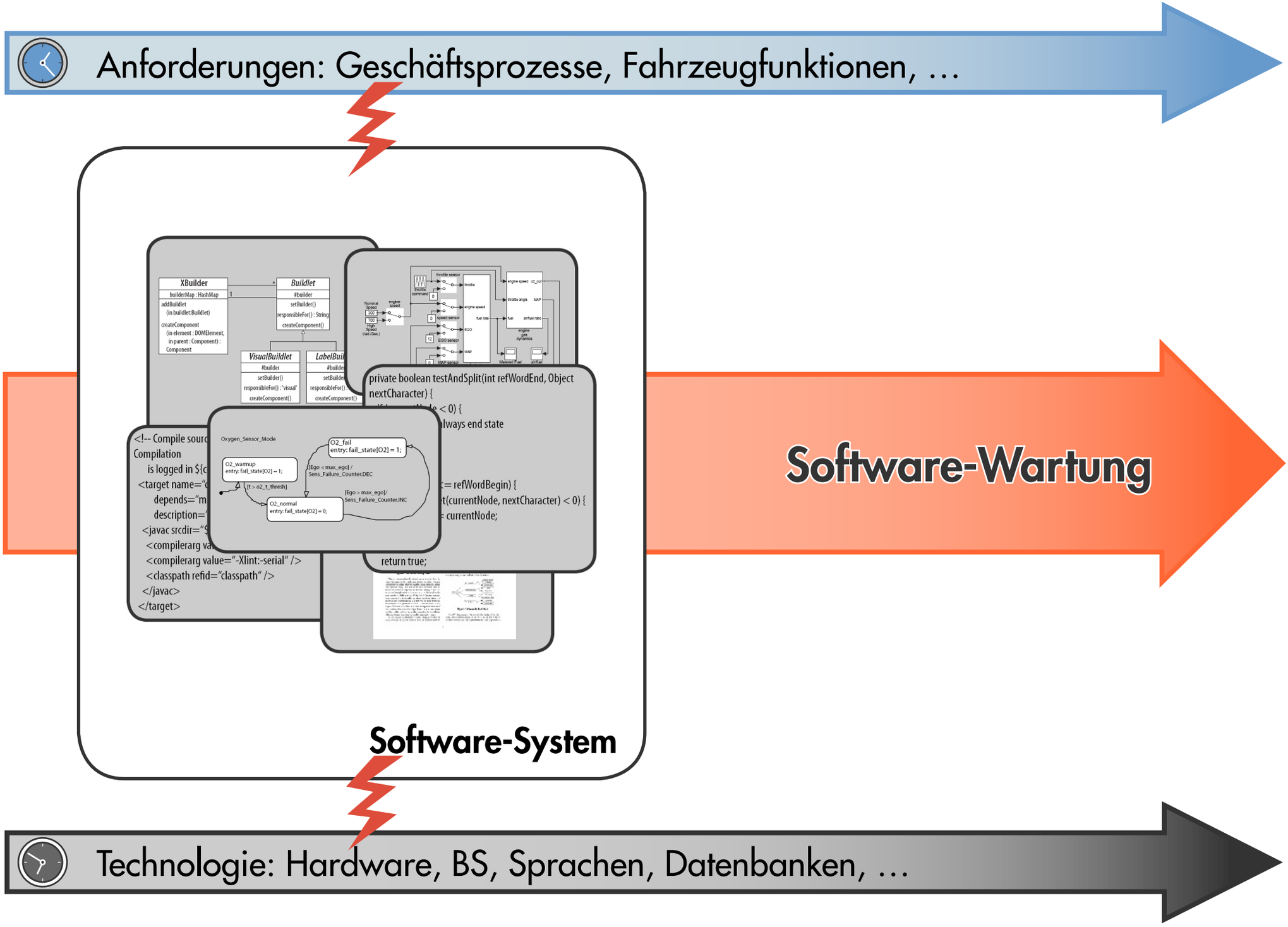


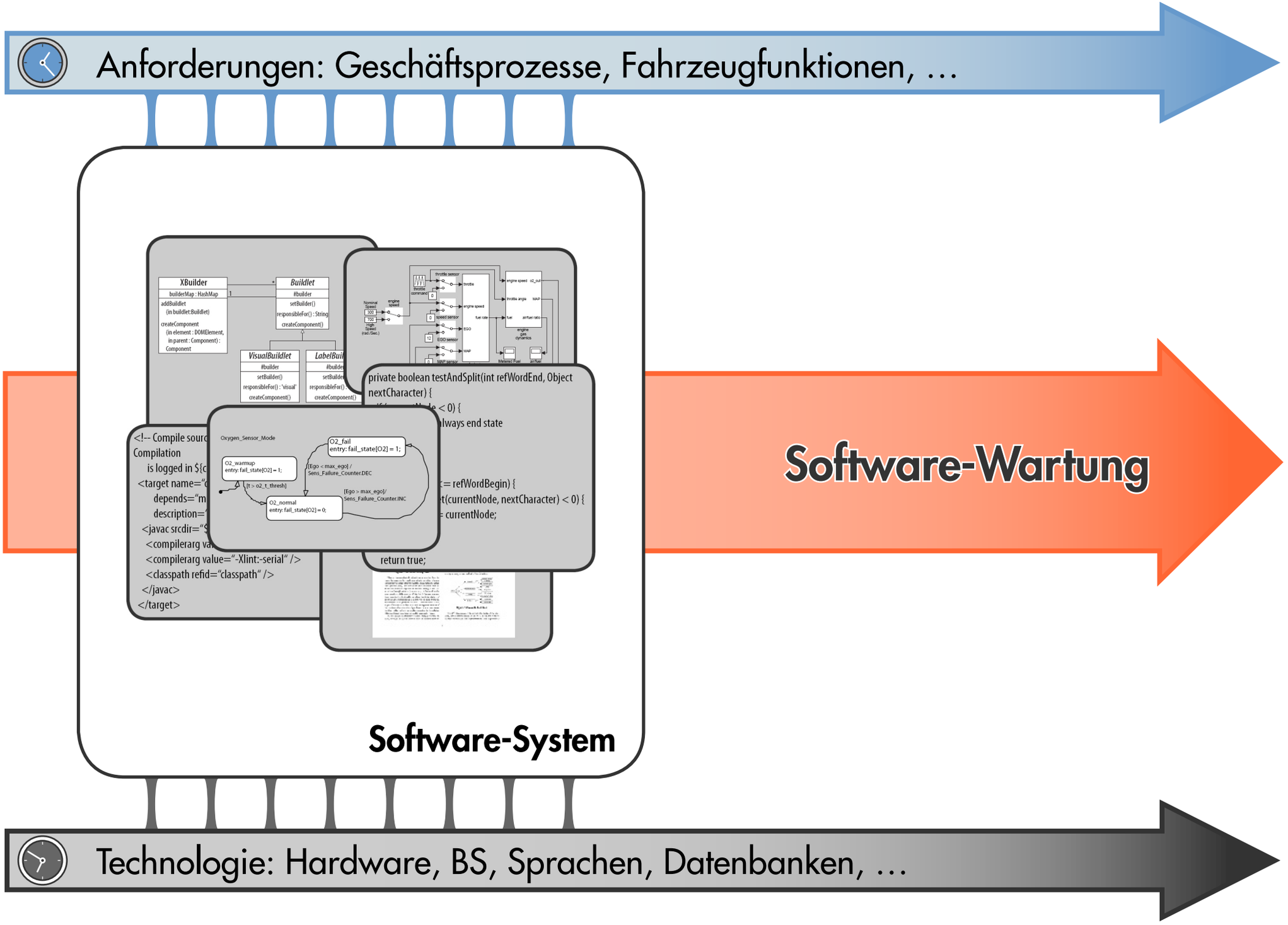
Software-System

Technologie: Hardware, BS, Sprachen, Datenbanken, …


Anforderungen: Geschäftsprozesse, Fahrzeugfunktionen, …
XBuilder
builderMap : HashMap
addBuildlet
(in buildlet:Buildlet)
createComponent
(in element : DOMElement,
in parent : Component) :
Component
Buildlet
#builder
setBuilder()
responsibleFor() : String
createComponent()
VisualBuildlet
#builder
setBuilder()
responsibleFor() : 'visual'
createComponent()
private boolean testAndSplit(int refWordEnd, Object
nextCharacter) {
always end state
= refWordBegin) {
t(currentNode, nextCharacter) < 0) {
= currentNode;
return true;
<!-- Compile sourc
Compilation
is logged in ${c
<target name="c
depends="m
description="
<javac srcdir="S
<compilerarg va
<compilerarg value="-Xlint:-serial" />
<classpath refid="classpath" />
</javac>
</target>
Oxygen_Sensor_Mode
O2_fail
entry: fail_state[O2] = 1;
O2_warmup
entry: fail_state[O2] = 1;
[Ego < max_ego] /
Sens_Failure_Counter.DEC
[t > o2_t_thresh]
[Ego > max_ego]/
Sens_Failure_Counter.INC
O2_normal
entry: fail_state[O2] = 0;
Software-System
Technologie: Hardware, BS, Sprachen, Datenbanken, …

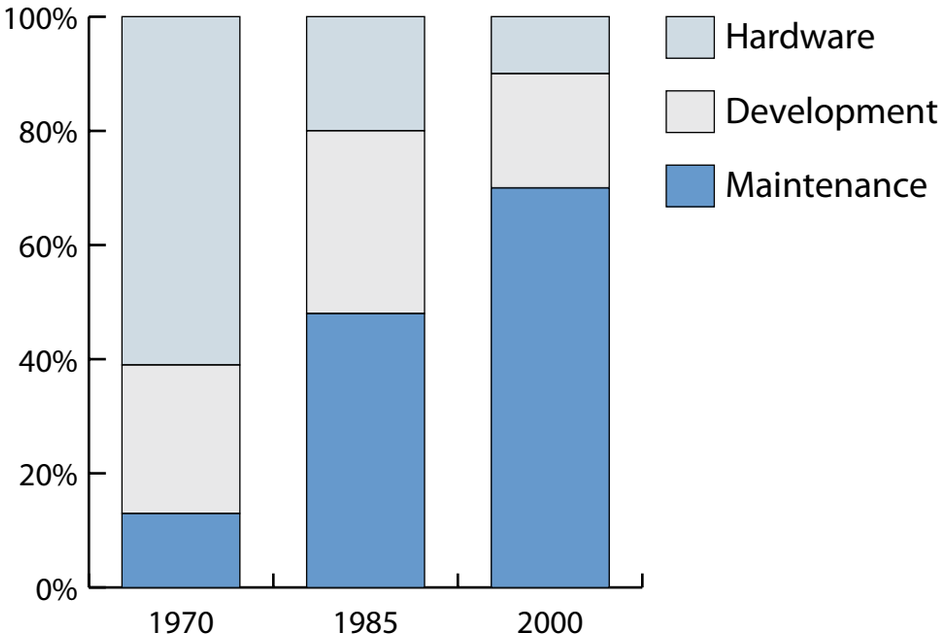

Anforderungen: Geschäftsprozesse, Fahrzeugfunktionen, …
XBuilder
builderMap : HashMap
addBuildlet
(in buildlet:Buildlet)
createComponent
(in element : DOMElement,
in parent : Component) :
Component
Buildlet
#builder
setBuilder()
responsibleFor() : String
createComponent()
VisualBuildlet
#builder
setBuilder()
responsibleFor() : 'visual'
createComponent()
private boolean testAndSplit(int refWordEnd, Object
nextCharacter) {
always end state
return true;
Oxygen_Sensor_Mode
O2_fail
entry: fail_state[O2] = 1;
O2_warmup
entry: fail_state[O2] = 1;
[Ego < max_ego] /
Sens_Failure_Counter.DEC
[t > o2_t_thresh]
[Ego > max_ego]/
Sens_Failure_Counter.INC
O2_normal
entry: fail_state[O2] = 0;
Software-System
Software-Wartung
Technologie: Hardware, BS, Sprachen, Datenbanken, …


Anforderungen: Geschäftsprozesse, Fahrzeugfunktionen, …
XBuilder
builderMap : HashMap
addBuildlet
(in buildlet:Buildlet)
createComponent
(in element : DOMElement,
in parent : Component) :
Component
Buildlet
#builder
setBuilder()
responsibleFor() : String
createComponent()
VisualBuildlet
#builder
setBuilder()
responsibleFor() : 'visual'
createComponent()
private boolean testAndSplit(int refWordEnd, Object
nextCharacter) {
always end state
return true;
Oxygen_Sensor_Mode
O2_fail
entry: fail_state[O2] = 1;
O2_warmup
entry: fail_state[O2] = 1;
[Ego < max_ego] /
Sens_Failure_Counter.DEC
[t > o2_t_thresh]
[Ego > max_ego]/
Sens_Failure_Counter.INC
O2_normal
entry: fail_state[O2] = 0;
<target name="
depends="m
description="
<javac srcdir="
<compilerarg va
<compilerarg value="-Xlint:-serial" />
<classpath refid="classpath" />
</javac>
</target>
Software-System
Software-Wartung
Technologie: Hardware, BS, Sprachen, Datenbanken, …

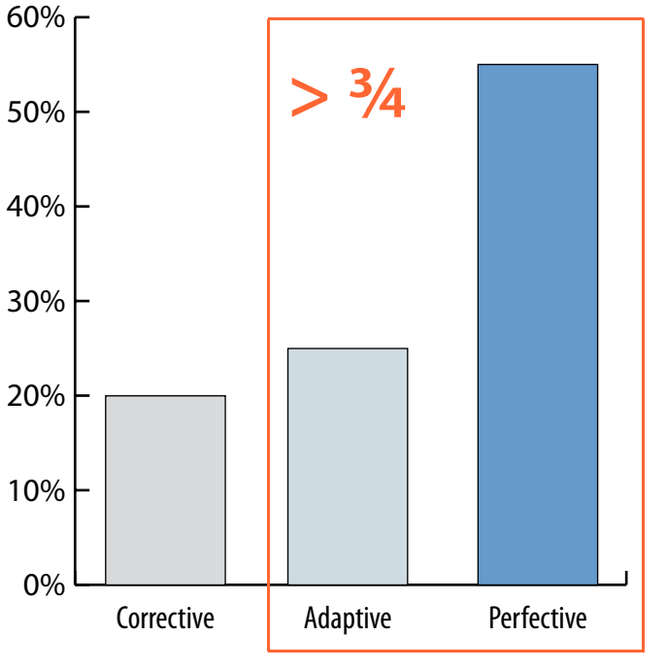

Anforderungen: Geschäftsprozesse, Fahrzeugfunktionen, …
XBuilder
builderMap : HashMap
addBuildlet
(in buildlet:Buildlet)
createComponent
(in element : DOMElement,
in parent : Component) :
Component
Buildlet
#builder
setBuilder()
responsibleFor() : String
createComponent()
VisualBuildlet
#builder
setBuilder()
responsibleFor() : 'visual'
createComponent()
private boolean testAndSplit(int refWordEnd, Object
nextCharacter) {
always end state
return true;
Oxygen_Sensor_Mode
O2_fail
entry: fail_state[O2] = 1;
O2_warmup
entry: fail_state[O2] = 1;
[Ego < max_ego] /
Sens_Failure_Counter.DEC
[t > o2_t_thresh]
[Ego > max_ego]/
Sens_Failure_Counter.INC
O2_normal
entry: fail_state[O2] = 0;
Software-System
Software-Wartung
Technologie: Hardware, BS, Sprachen, Datenbanken, …

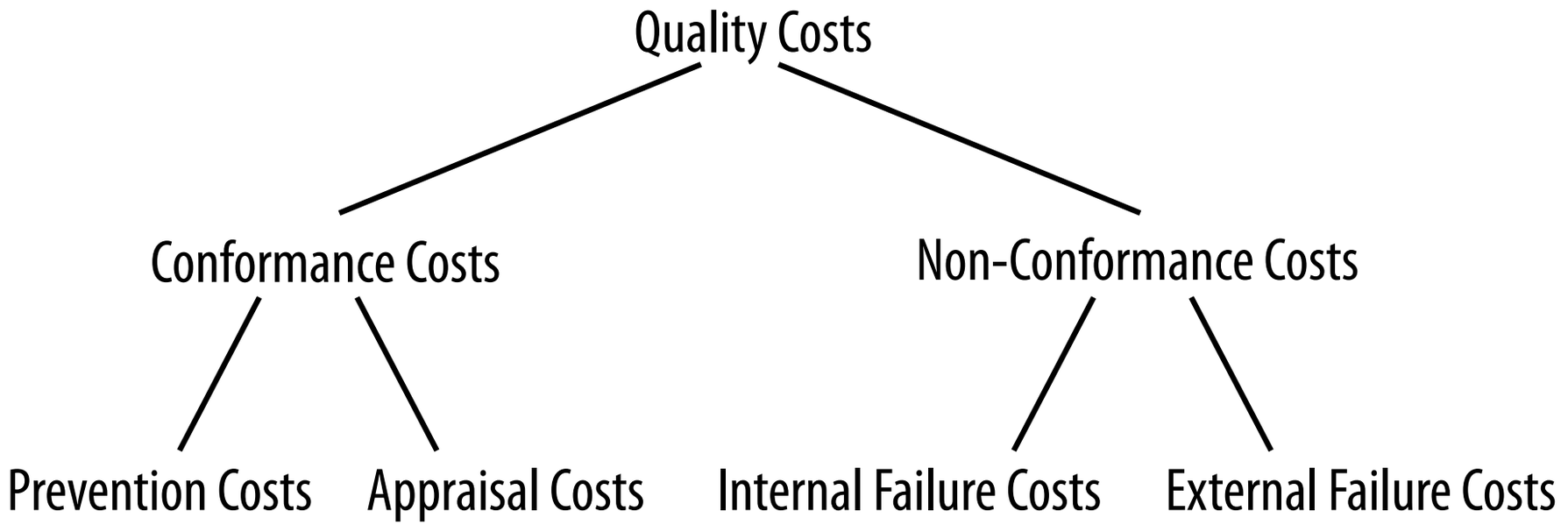
Relevanz

Lebenszyklus-Kosten



Wartungsfälle




Quality Costs
Conformance Costs
Non-Conformance Costs
Prevention Costs
Appraisal Costs
Internal Failure Costs
External Failure Costs

```
#include                                        <math.h>
#include                                       <sys/time.h>
#include                                       <X11/Xlib.h>
#include                                      <X11/keysym.h>
                                              double L ,o ,P
                                             ,_=dt,T,Z,D=1,d,
                                             s[999],E,h= 8,I,
                                             J,K,w[999],M,m,O
                                            ,n[999],j=33e-3,i=
                                            1E3,r,t, u,v ,W,S=
                                            74.5,l=221,X=7.26,
                                            a,B,A=32.2,c, F,H;
                                            int N,q, C, y,p,U;
                                           Window z; char f[52]
                                         ; GC k; main(){ Display*e=
 XOpenDisplay( 0); z=RootWindow(e,0); for (XSetForeground(e,k=XCreateGC (e,z,0,0),BlackPixel(e,0))
; scanf("%lf%lf%lf",y +n,w+y, y+s)+1; y ++); XSelectInput(e,z= XCreateSimpleWindow(e,z,0,0,400,400,
0,0,WhitePixel(e,0) ),KeyPressMask); for(XMapWindow(e,z); ; T=sin(O)){ struct timeval G={ 0,dt*1e6}
; K= cos(j); N=1e4; M+= H*_; Z=D*K; F+=_*P; r=E*K; W=cos( O); m=K*W; H=K*T; O+=D*_*F/ K+d/K*E*_; B=
sin(j); a=B*T*D-E*W; XClearWindow(e,z); t=T*E+ D*B*W; j+=d*_*D-_*F*E; P=W*E*B-T*D; for (o+=(I=D*W+E
*T*B,E*d/K *B+v+B/K*F*D)*_; p<y; ){ T=p[s]+i; E=c-p[w]; D=n[p]-L; K=D*m-B*T-H*E; if(p [n]+w[ p]+p[s
]== 0|K <fabs(W=T*r-I*E +D*P) |fabs(D=t *D+Z *T-a *E)> K)N=1e4; else{ q=W/K *4E2+2e2; C= 2E2+4e2/ K
 *D; N-1E4&& XDrawLine(e ,z,k,N ,U,q,C); N=q; U=C; } ++p; } L+=_* (X*t +P*M+m*l); T=X*X+ l*l+M *M;
  XDrawString(e,z,k ,20,380,f,17); D=v/l*15; i+=(B *l-M*r -X*Z)*_; for(; XPending(e); u *=CS!=N){
                                       XEvent z; XNextEvent(e ,&z);
                                           ++*((N=XLookupKeysym
                                             (&z.xkey,0))-IT?
                                             N-LT? UP-N?& E:&
                                             J:& u: &h); --*(
                                             DN -N? N-DT ?N==
                                             RT?&u: & W:&h:&J
                                              ); } m=15*F/l;
                                              c+=(I=M/ l,l*H
                                              +I*M+a*X)*_; H
                                              =A*r+v*X-F*l+(
                                              E=.1+X*4.9/l,t
                                              =T*m/32-I*T/24
                                               )/S; K=F*M+(
                                               h* 1e4/l-(T+
                                               E*5*T*E)/3e2
                                               )/S-X*d-B*A;
                                               a=2.63 /l*d;
                                               X+=( d*l-T/S
                                                *(.19*E +a
                                                *.64+J/1e3
                                                )-M* v +A*
                                                Z)*_; l +=
                                                K *_; W=d;
                                                sprintf(f,
                                                "%5d  %3d"
                                                "%7d",p =l
                                               /1.7,(C=9E3+
                                    O*57.3)%0550,(int)i); d+=T*(.45-14/l*
                                   X-a*130-J* .14)*_/125e2+F*_*v; P=(T*(47
                                   *I-m* 52+E*94 *D-t*.38+u*.21*E) /1e2+W*
                                   179*v)/2312; select(p=0,0,0,0,&G); v-=(
                                    W*F-T*(.63*m-I*.086+m*E*19-D*25-.11*u
                                     )/107e2)*_; D=cos(o); E=sin(o); } }
```

```
                for (const_iterator<Permission> permission =
                     permissions.begin();
                     permission != permissions.end(); ++permission) {
                    if (!permission->isInternal()) {
                        if (user->hasPermission (permission)) {
                            cout << "   " << permission << endl;
                        }
                    }
                }
            }
        }
    } /* bezieht sich auf if in Zeile 172 */
}
```

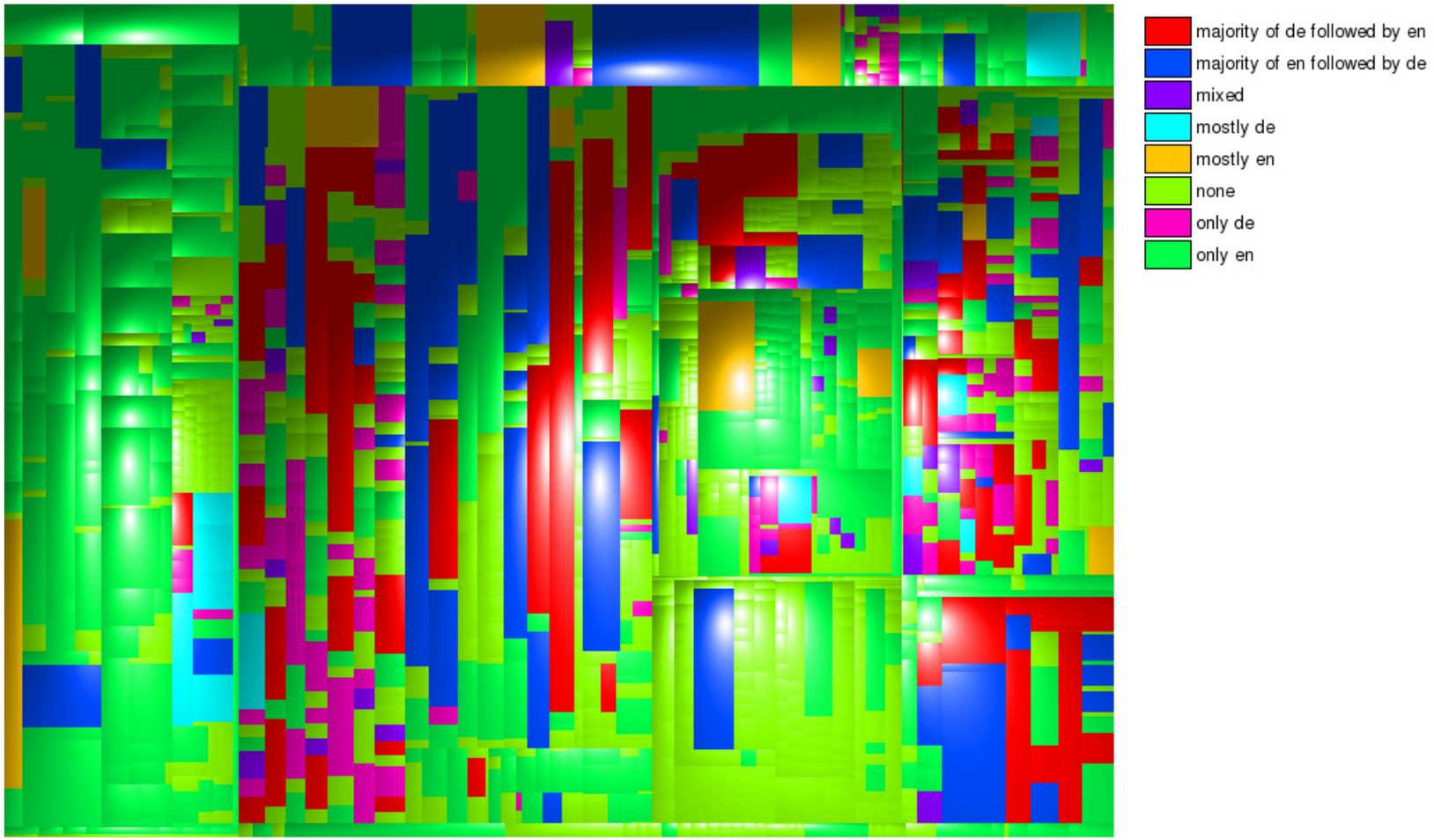
- 30.444 Zeilen C# in einer Klasse
- 3.526 Zeilen C# in einer Methode
- Schachtelungstiefe 24
- 14 Parameter

Bezeichner	Anzahl
S0	810
S1	612
r1	490
I1	423
S2	362
i1	248
S3	211
SQL	199
CM	193
I2	155

Type	#	%	chars	%
Keywords	967.665	11%	4.650.273	13%
Delimiters	4.096.112	47%	4.096.112	11%
Operators	531.444	6%	669.932	2 %
Identifiers	2.873.232	32%	25.646.263	72%
Literals	301.081	3%	708.308	2%
Total	8.769.534	100%	35.770.888	100%

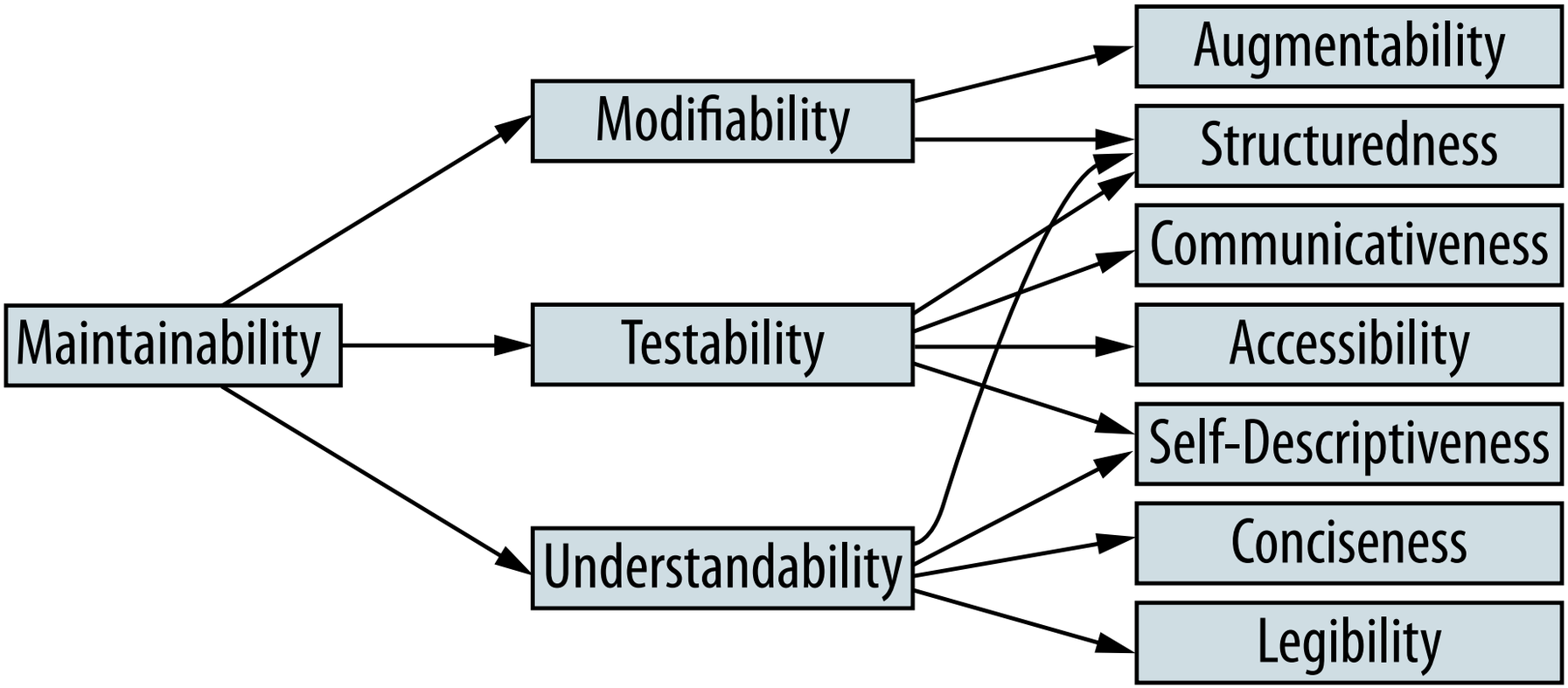
- Eclipse 3.0: 94.829 Bezeichner bestehend aus 7.233 Wörtern
- Identifikatoren sind das Vokabular eines Programms (ca. 70%)
- Synonyme & Homonyme
- Limitierung durch Programmiersprachen

F. Deißenböck, M. Pizka, *Concise and Consistent Naming*, IWPC, 2005



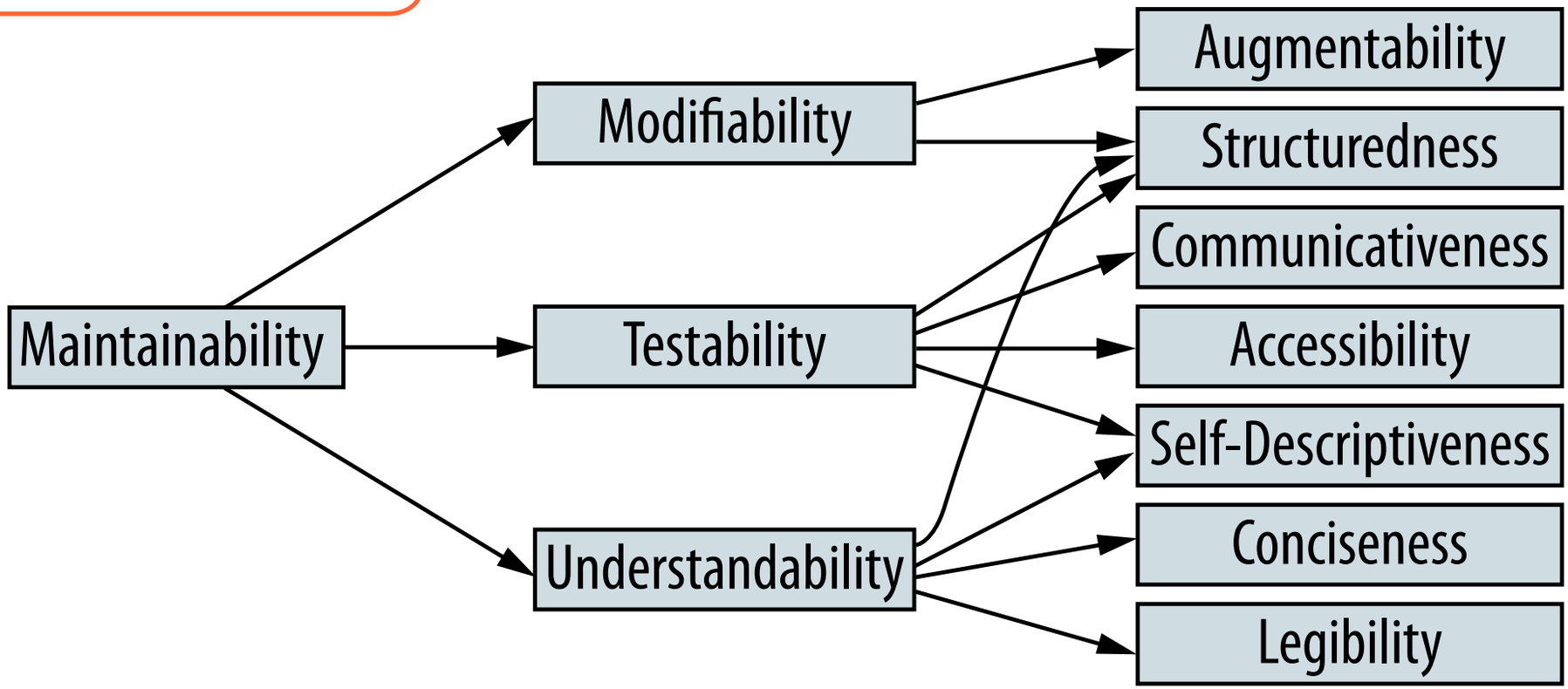

```
/* if-else-Kaskade */
if (GlobalSettings::mayPrintToConsole ()) {
      if (!user->isAdmin()) {
            if (!permission->isInternal()) {
                  if (user->hasPermission ...
                        ...
```

```
int search (String s, String[] strings) {
    //strings = toLowerCase(strings);
    strings = sort(strings);
    List<String> l = asList(strings);
    return search(s, strings);
}
```



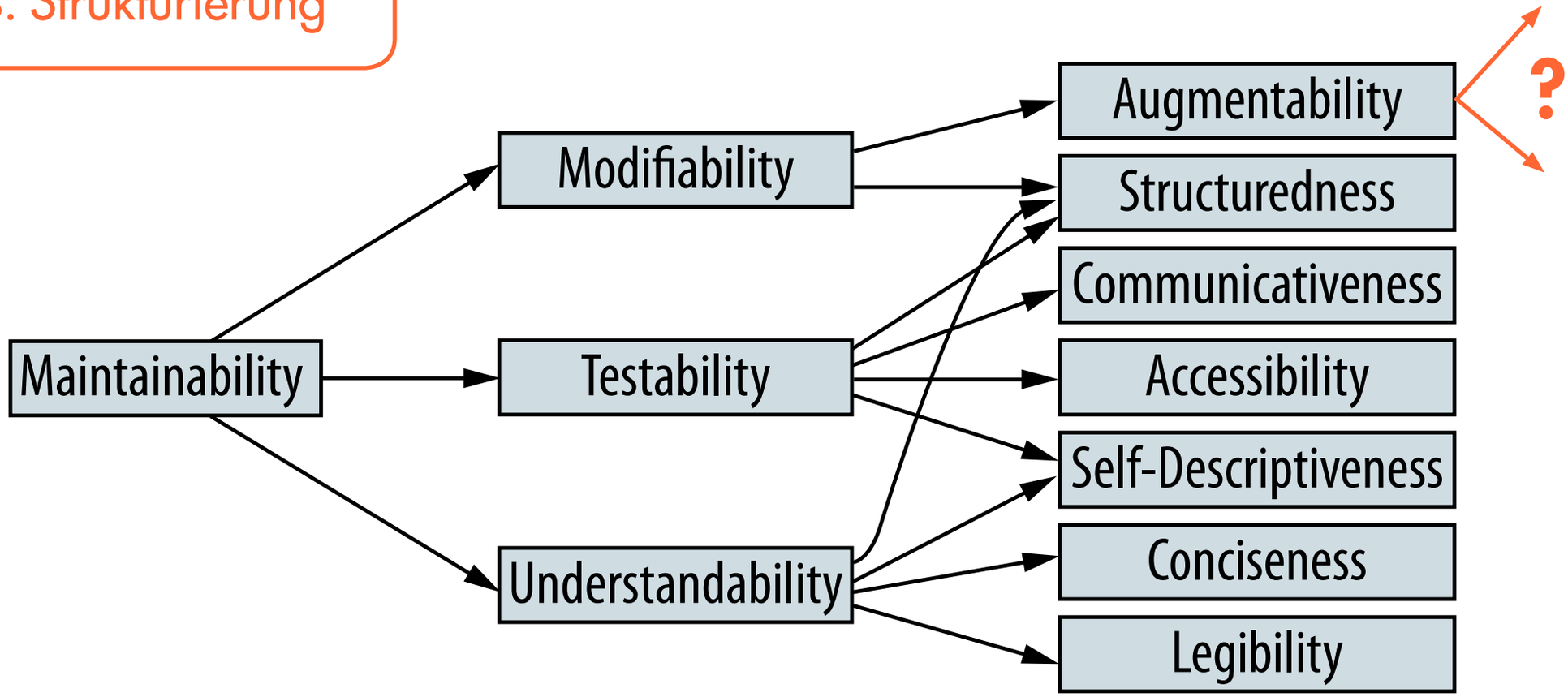
B. W. Boehm et al., Characteristics of Software Quality, 1978

1. Begründung
2. Überprüfung
3. Strukturierung



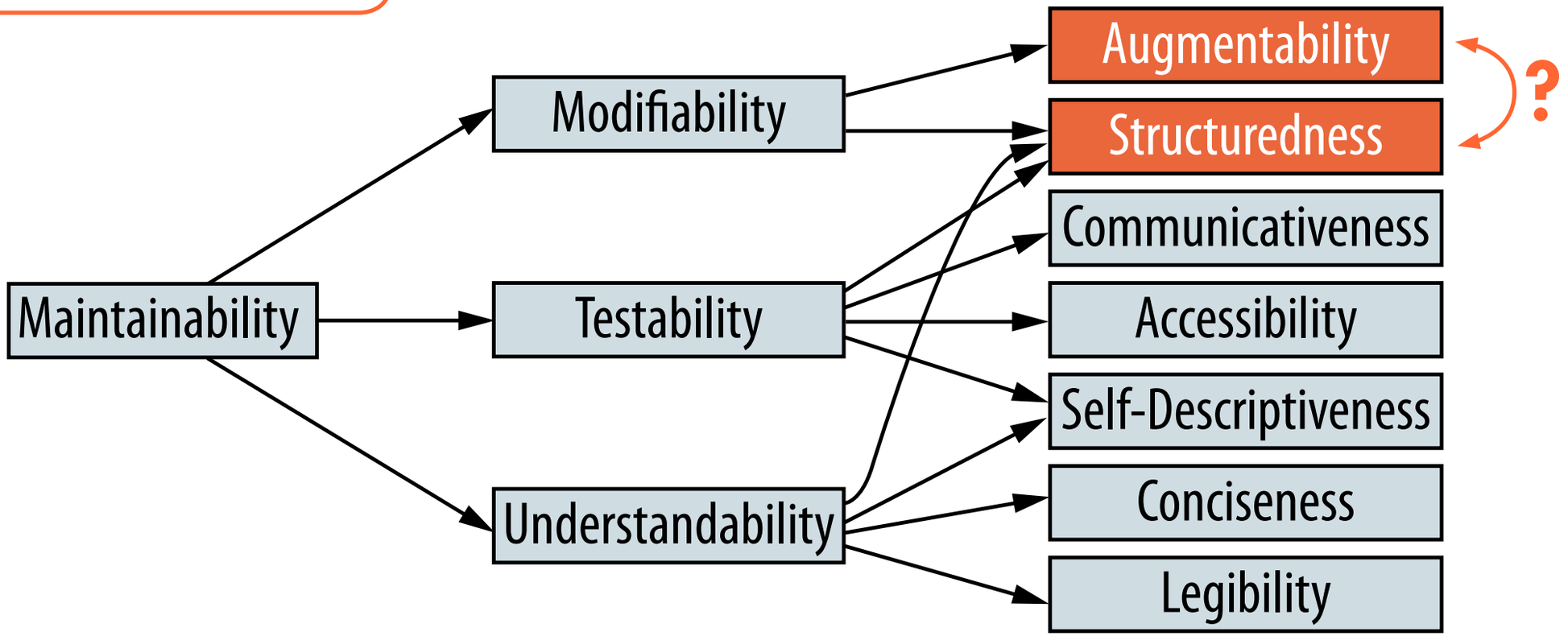
B. W. Boehm et al., Characteristics of Software Quality, 1978

1. Begründung
2. Überprüfung
3. Strukturierung



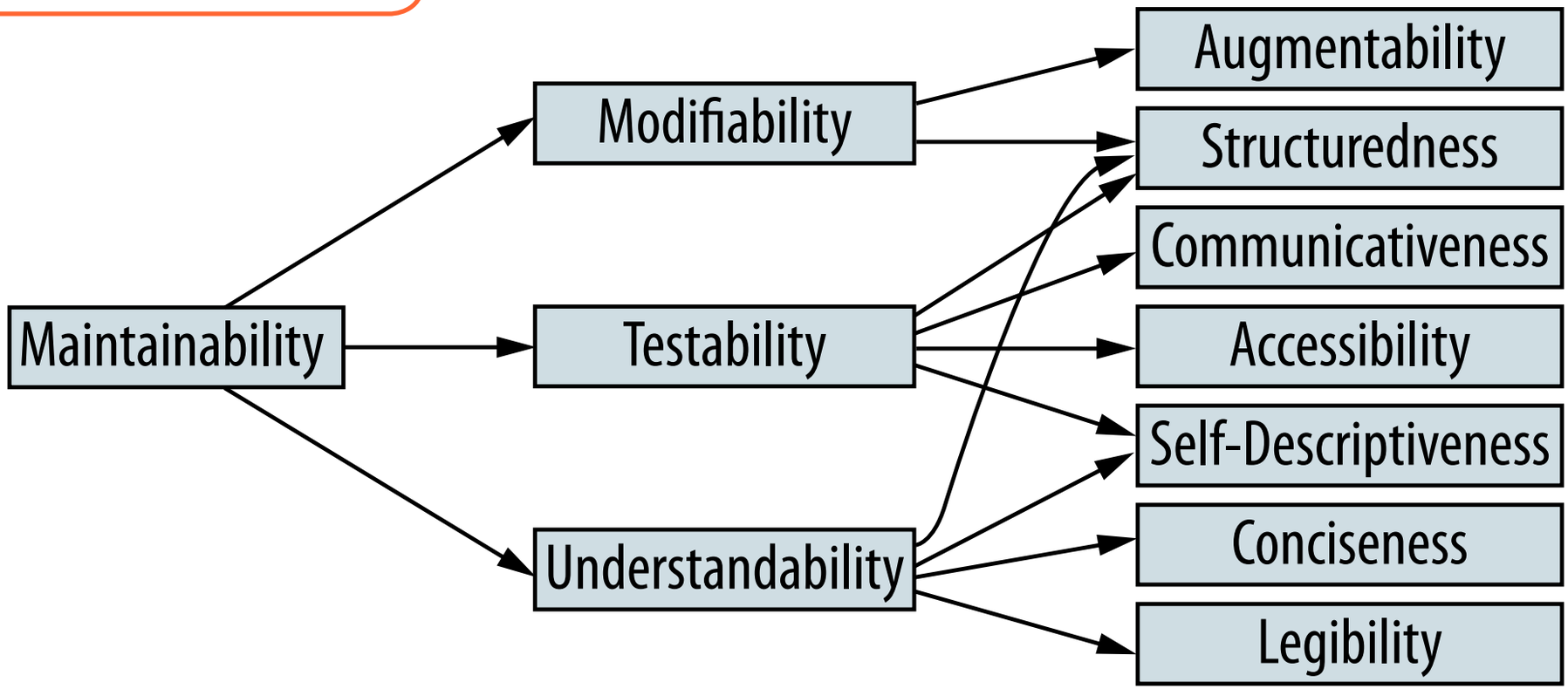
B. W. Boehm et al., Characteristics of Software Quality, 1978

1. Begründung
2. Überprüfung
3. Strukturierung

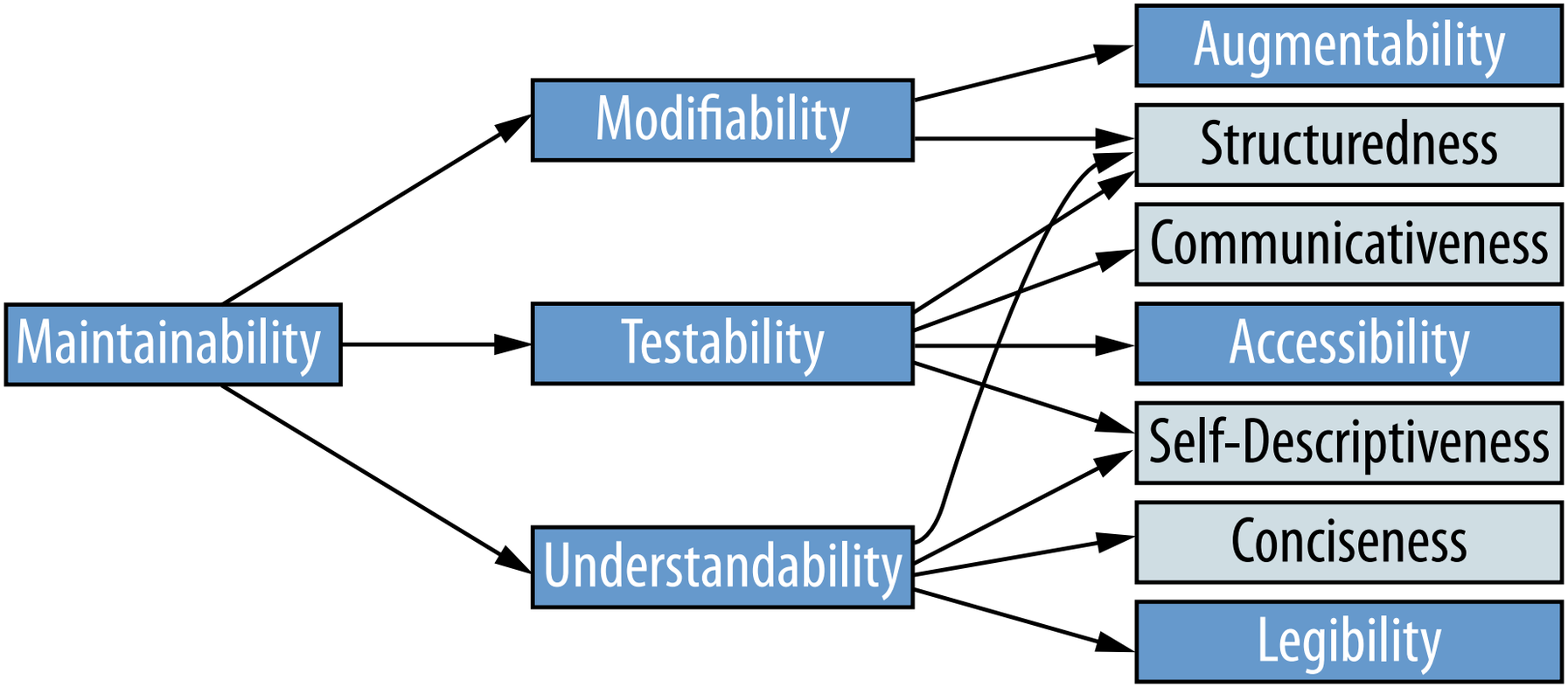


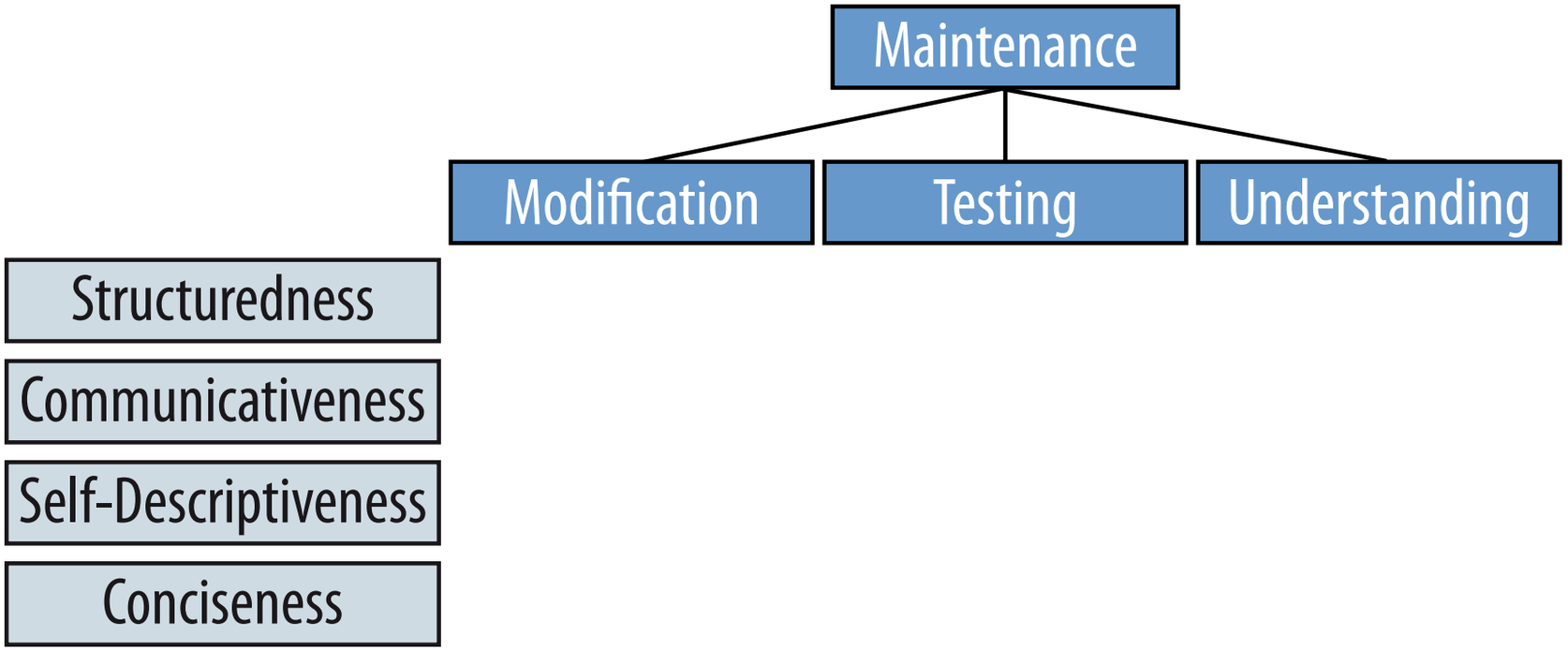
B. W. Boehm et al., Characteristics of Software Quality, 1978

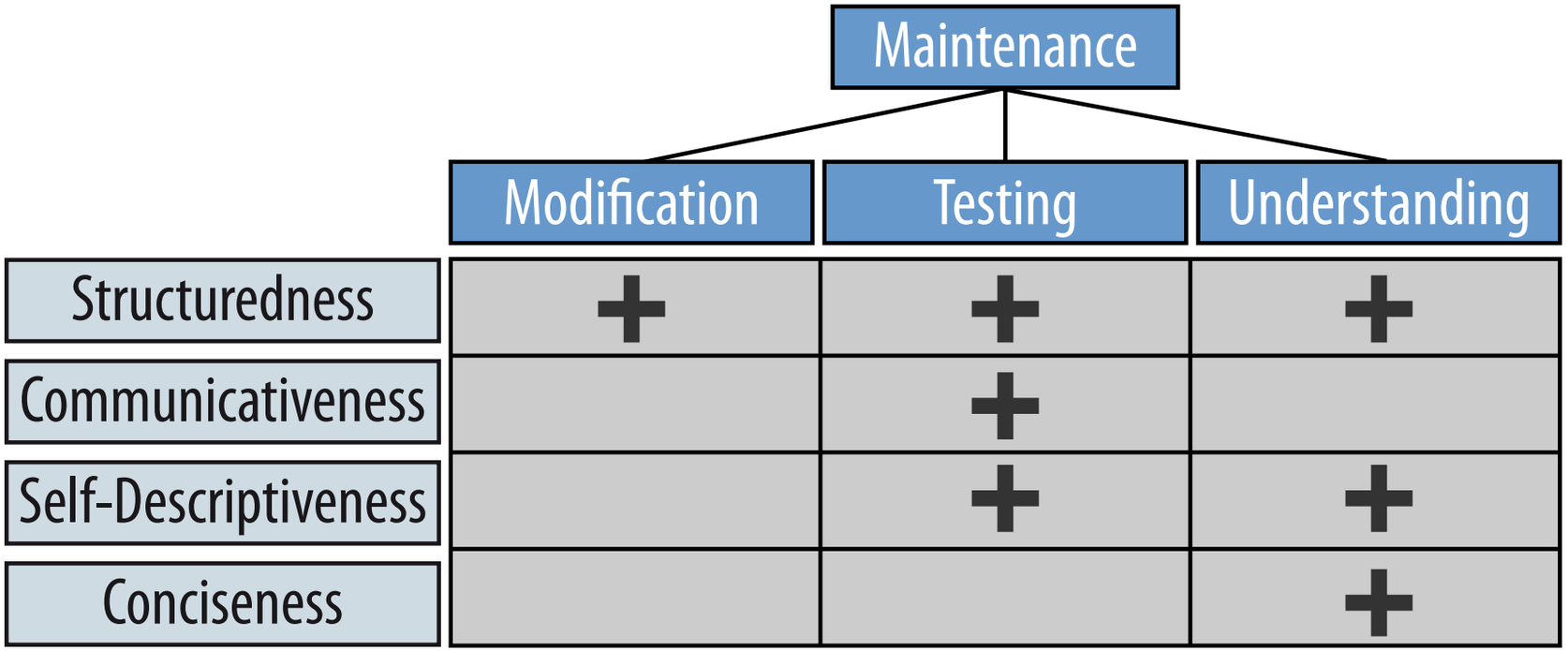
1. Begründung
2. Überprüfung
3. Strukturierung



B. W. Boehm et al., Characteristics of Software Quality, 1978

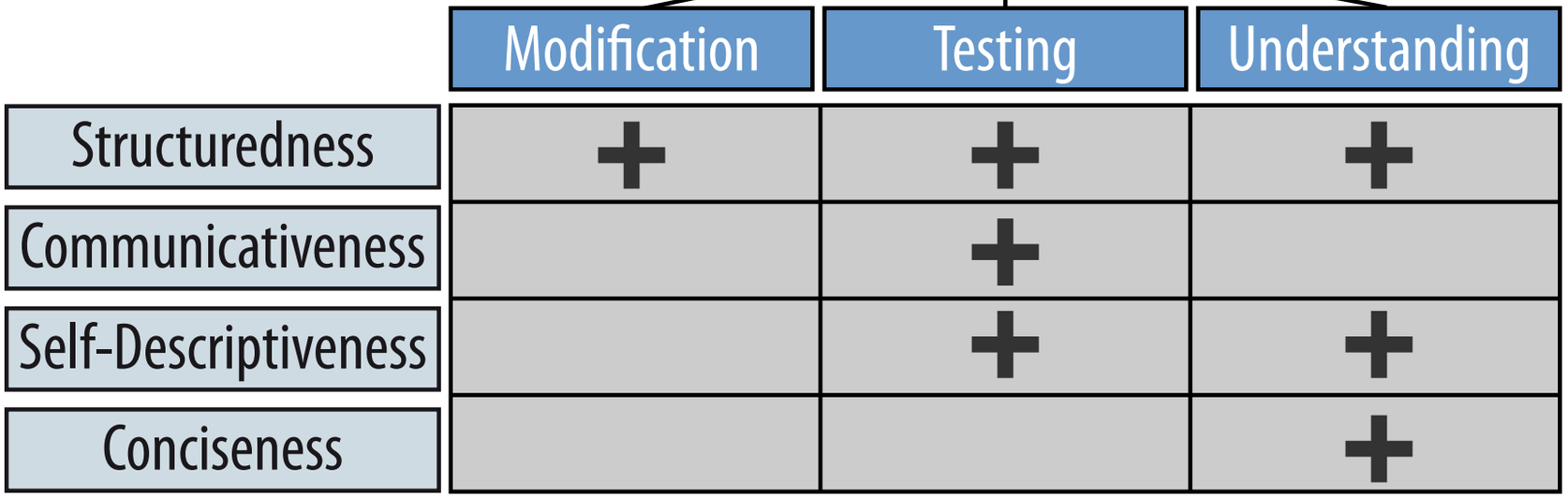

Maintainability
Modifiability
Testability
Understandability
Augmentability
Structuredness
Communicativeness
Accessibility
Self-Descriptiveness
Conciseness
Legibility

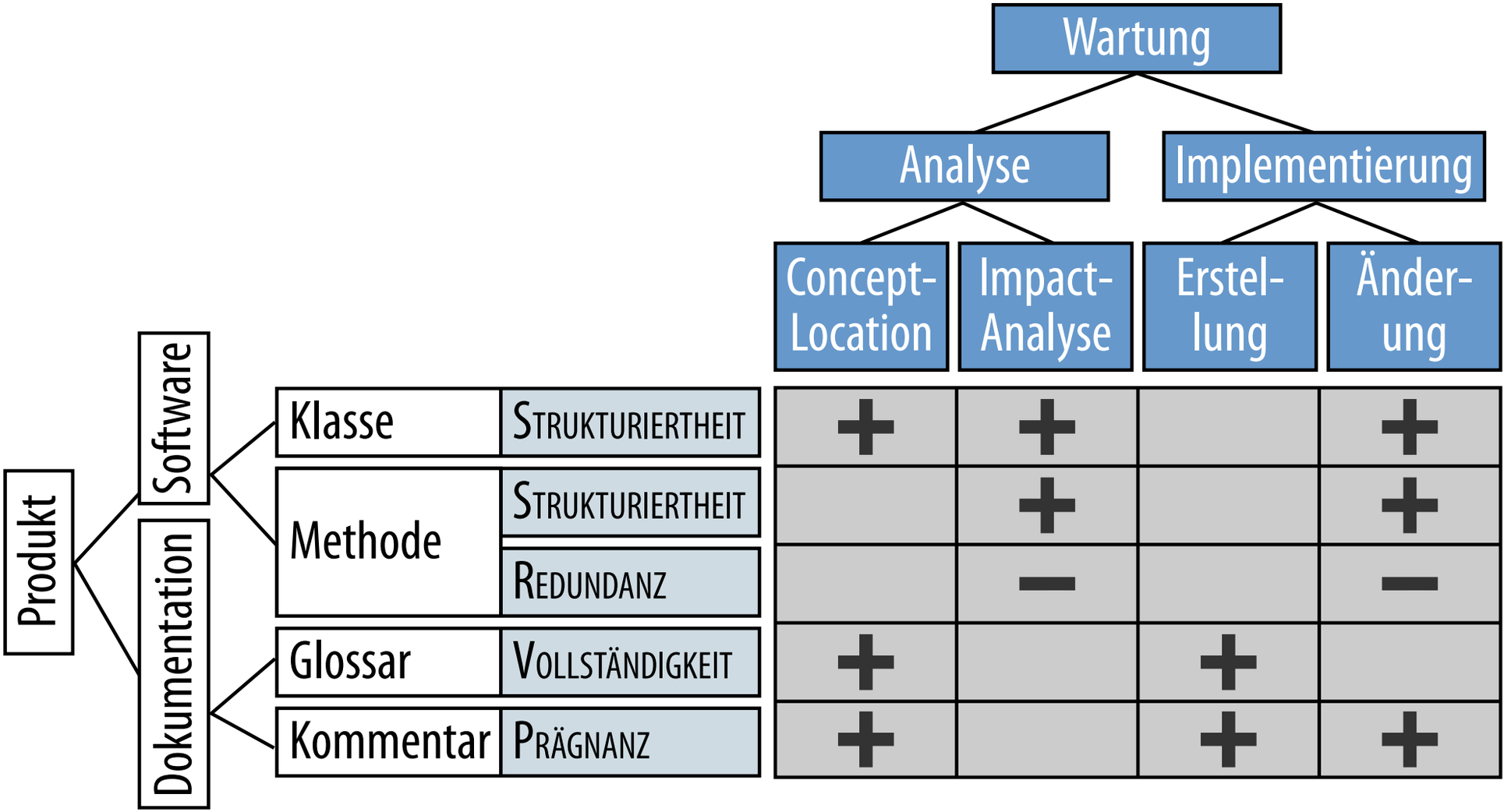

Maintenance
Modification
Testing
Understanding
Structuredness
Communicativeness
Self-Descriptiveness
Conciseness


Maintenance
Modification
Testing
Understanding
Structuredness
Communicativeness
Self-Descriptiveness
Conciseness

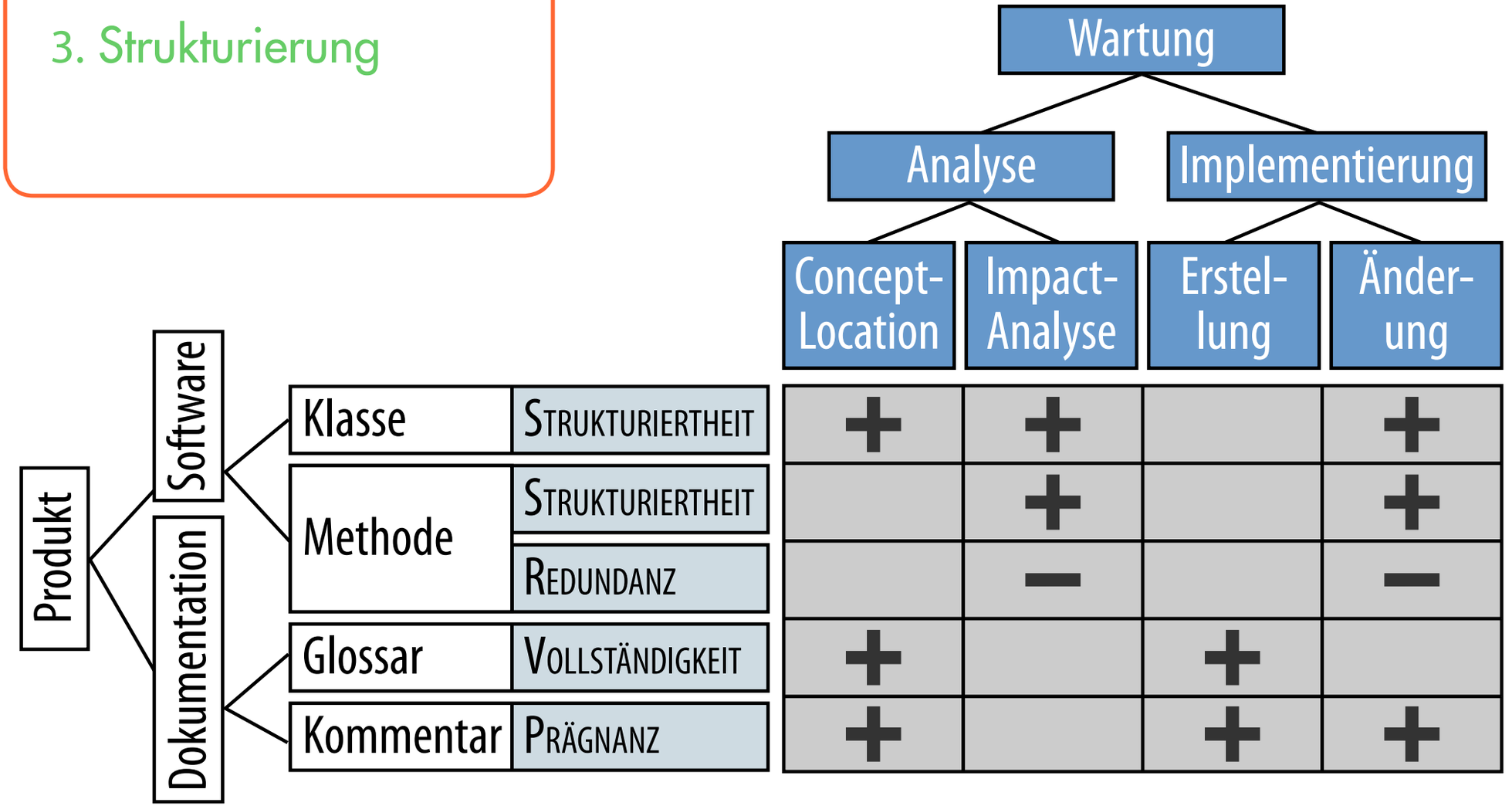
»The effort needed to make specified modifications to a component implementation.«

SEI Open Systems Glossary




Wartung
Analyse
Implementierung
Concept-Location
Impact-Analyse
Erstel-lung
Änder-ung
Produkt
Software
Dokumentation
Klasse
Strukturiertheit
Methode
Strukturiertheit
Redundanz
Glossar
Vollständigkeit
Kommentar
Prägnanz
+
+
+
+
+
+
−
−
+
+
+
+

1. Begründung
2. Überprüfung
3. Strukturierung



TODO: ist das wirklich Okay, einem QWidget einen QObject Parent zu geben?

TODO: Soll das so sein? Jetzt echt? Alle alten Sachen in die Tonne?

TODO: WEG MIT DIESEM DRECK - RK SOLLTE FÜR JEDEN INDEX IMMER…

TODO: Nicht benutzen, hat hier überhaupt nix verloren

FIXME: solve timeout problem

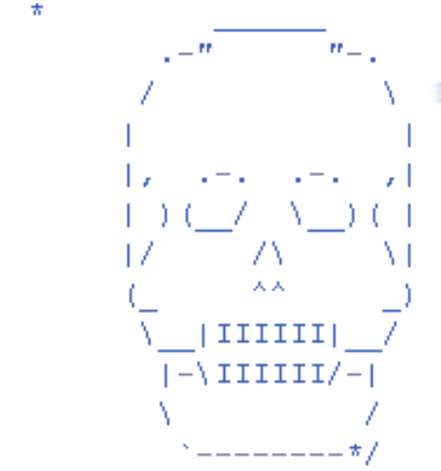
FIXME: REFACTOREN ! ich habe das jetzt einfache rüberkopiert, ist Müll so

TODO: remove this hack

TODO: Worin unterscheidet sich dieser Test vom vorherigen??

```
29.10.2009 (26
           27 /// @todo [redacted] Dieser Code hier ist FALSCH. Wir sind jedoch kurz vor RC2 für R2009.
           28 /// Es wäre wesentlich gefährlicher dies hier jetzt zu korrigieren (Als sich weiterhin darauf
           29 /// zu verlassen, dass das [redacted] diese Aufgaben für uns erledigt!)
```

```
/**
 *
 *
```



[redacted]: Nur mit äußerster Vorsicht benutzen, da dieses Attribut nicht selten via JavaScript gesetzt wird und dies nicht immer korrekt !!!
'[redacted]' arbeitet zuverlässiger

[redacted] (06/2010): so nicht mehr richtig, weil die [redacted] inzwischen ebenfalls eine [redacted] hat.

```
try {
    ...
} catch (Throwable t) {}
```

```
try {
    ...
} catch (Exception ex) {
    ex.printStrackTrace();
}
```

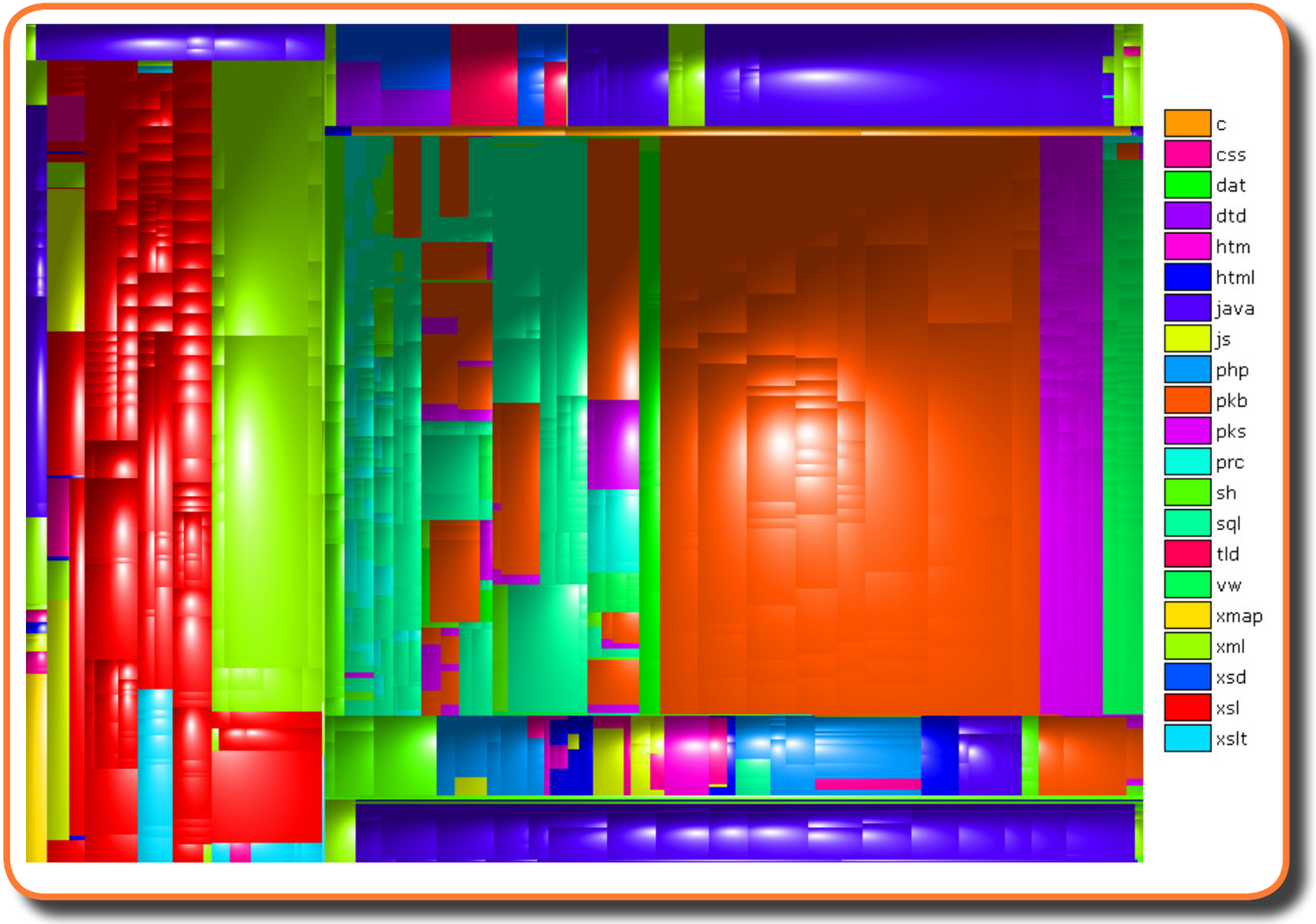
```
try {
    ...
} catch (Exception ex) {
    throw new
    RuntimeException(ex);
}
```

- ungenügende/keine Behandlung von Ausnahmen
- inkonsistente Verwendung von Ausnahmen
- Ausnahmen vs Rückgabewerte
- Java: Checked vs Unchecked Exceptions

```
public String format(String field) {
    int i;
    field = field.replaceAll("&|\\\\&", "&amp;").replaceAll("[\\n]{1,}", "<p>");

    StringBuffer sb = new StringBuffer();
    StringBuffer currentCommand = null;

    char c;
    boolean escaped = false, incommand = false;

    for (i = 0; i < field.length(); i++) {
        c = field.charAt(i);
        if (escaped && (c == '\\')) {
            sb.append('\\');
            escaped = false;
        } else if (c == '\\') {
            if (incommand){
                /* Close Command */
                String command = currentCommand.toString();
                Object result = CHARS.get(command);
                if (result != null) {
                    sb.append((String) result);
                } else {
                    sb.append(command);
                }
            }
            escaped = true;
            incommand = true;
            currentCommand = new StringBuffer();
        } else if (!incommand && (c == '{' || c == '}')) {
            // Swallow the brace.
        } else if (Character.isLetter(c) || (c == '%')
            || (Globals.SPECIAL_COMMAND_CHARS.indexOf(String.valueOf(c)) >
            escaped = false;

            if (!incommand)
                sb.append(c);
                // Else we are in a command, and should not keep the letter.
            else {
                currentCommand.append(c);
                testCharCom: if ((currentCommand.length() == 1)
                    && (Globals.SPECIAL_COMMAND_CHARS.indexOf(currentComm
```

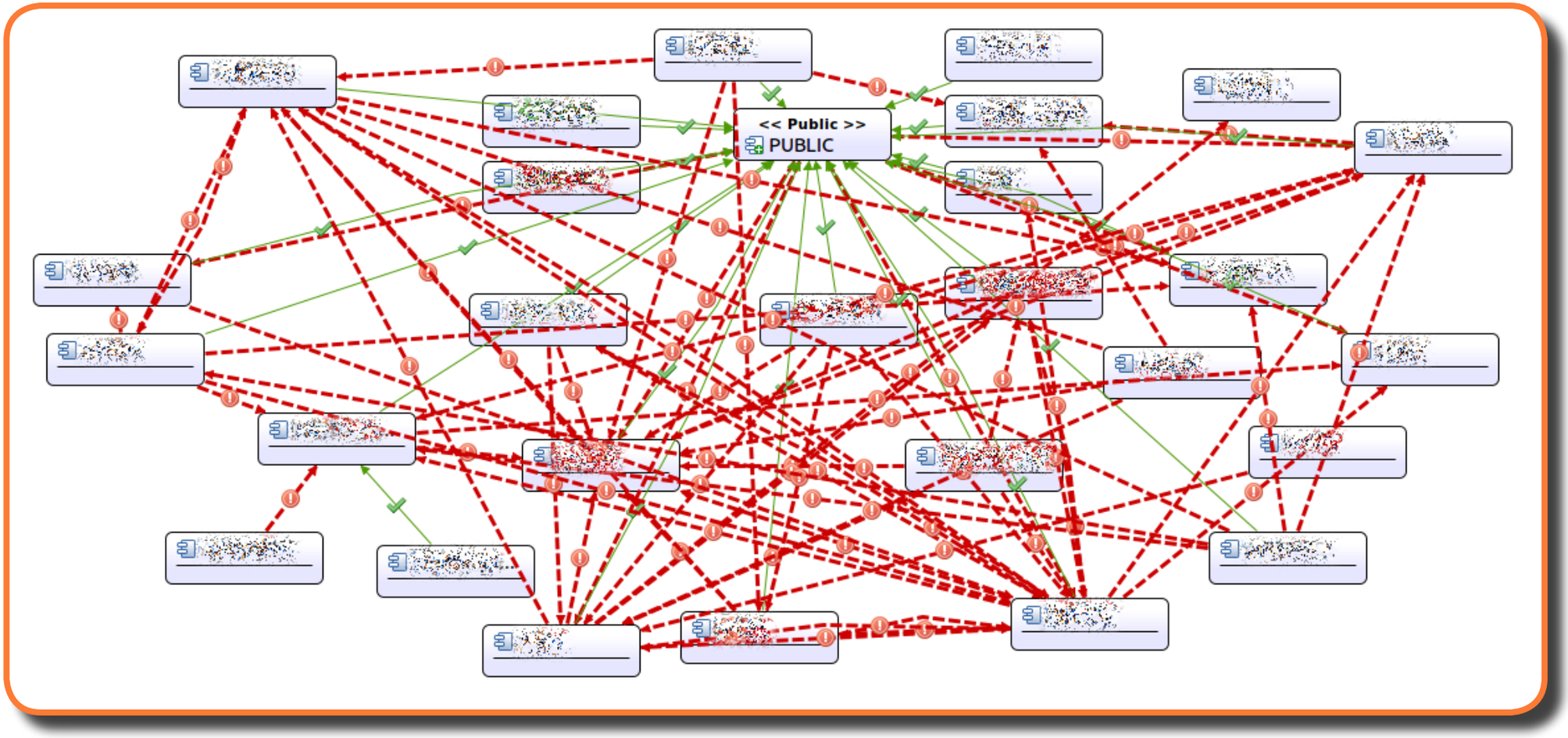
```
public String format(String field) {
    int i;
    field = field.replaceAll("&|\\\\&", "&amp;").replaceAll("[\\n]{1,}", "<p

    StringBuffer sb = new StringBuffer();
    StringBuffer currentCommand = null;

    char c;
    boolean escaped = false, incommand = false;

    for (i = 0; i < field.length(); i++) {
        c = field.charAt(i);
        if (escaped && (c == '\\')) {
            sb.append('\\');
            escaped = false;
        } else if (c == '\\') {
            if (incommand){
                /* Close Command */
                String command = currentCommand.toString();
                Object result = Globals.HTMLCHARS.get(command);
                if (result != null) {
                    sb.append((String) result);
                } else {
                    sb.append(command);
                }
            }
            escaped = true;
            incommand = true;
            currentCommand = new StringBuffer();
        } else if (!incommand && (c == '{' || c == '}')) {
            // Swallow the brace.
        } else if (Character.isLetter(c) || (c == '%')
            || (Globals.SPECIAL_COMMAND_CHARS.indexOf(String.valueO
            escaped = false;

            if (!incommand)
                sb.append(c);
                // Else we are in a command, and should not keep the letter.
            else {
                currentCommand.append(c);
                testCharCom: if ((currentCommand.length() == 1)
                    && (Globals.SPECIAL_COMMAND_CHARS.indexOf(currentC
```

Juergens et al., Do Code Clones Matter?, ICSE 2009

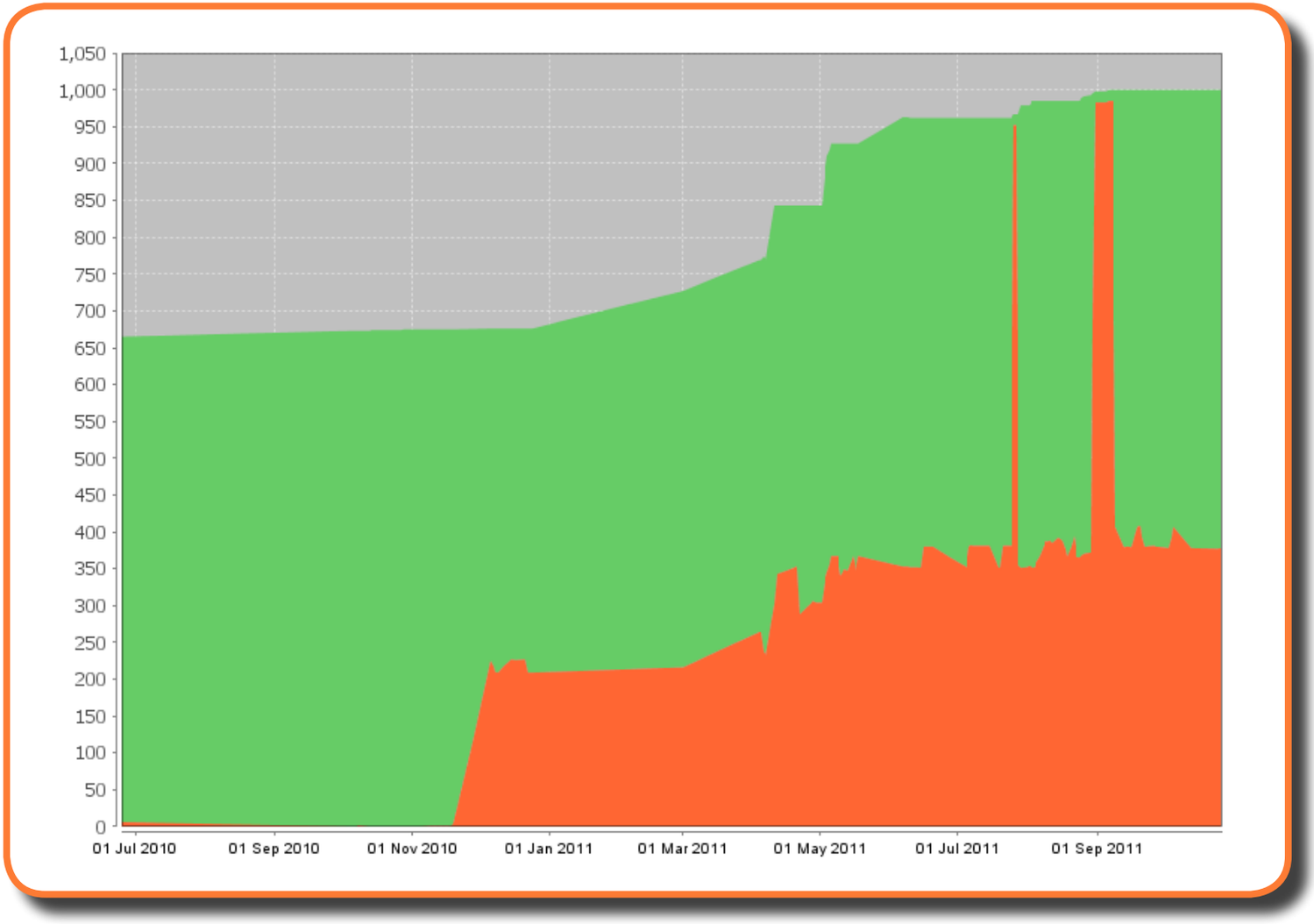


- >10 Programmiersprachen in einem System
- Wahl der Sprache auf Basis der Entwicklerverfügbarkeit
- veraltete Technologien
- Sprachvermischung

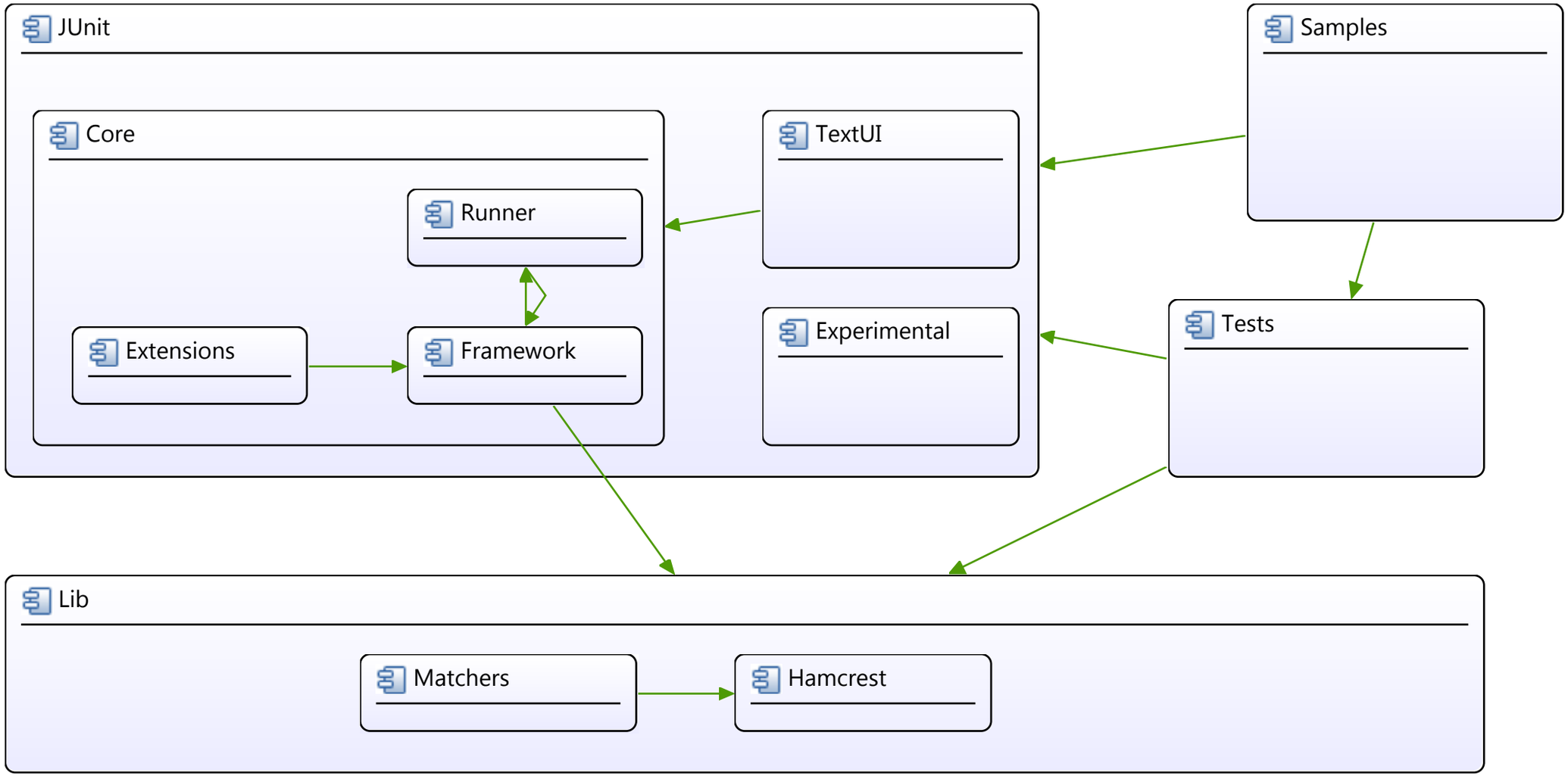
```
<body>
    <%    If Request.QueryString("ID") <> "" Then    %>
        <SCRIPT language="JavaScript">
            function doLogin() { X }
        </script>
    <% Else %>
        <SCRIPT language="JavaScript">
            function doLogin() { Y }
        </script>
    <% End If %>
    ...
</body>
```

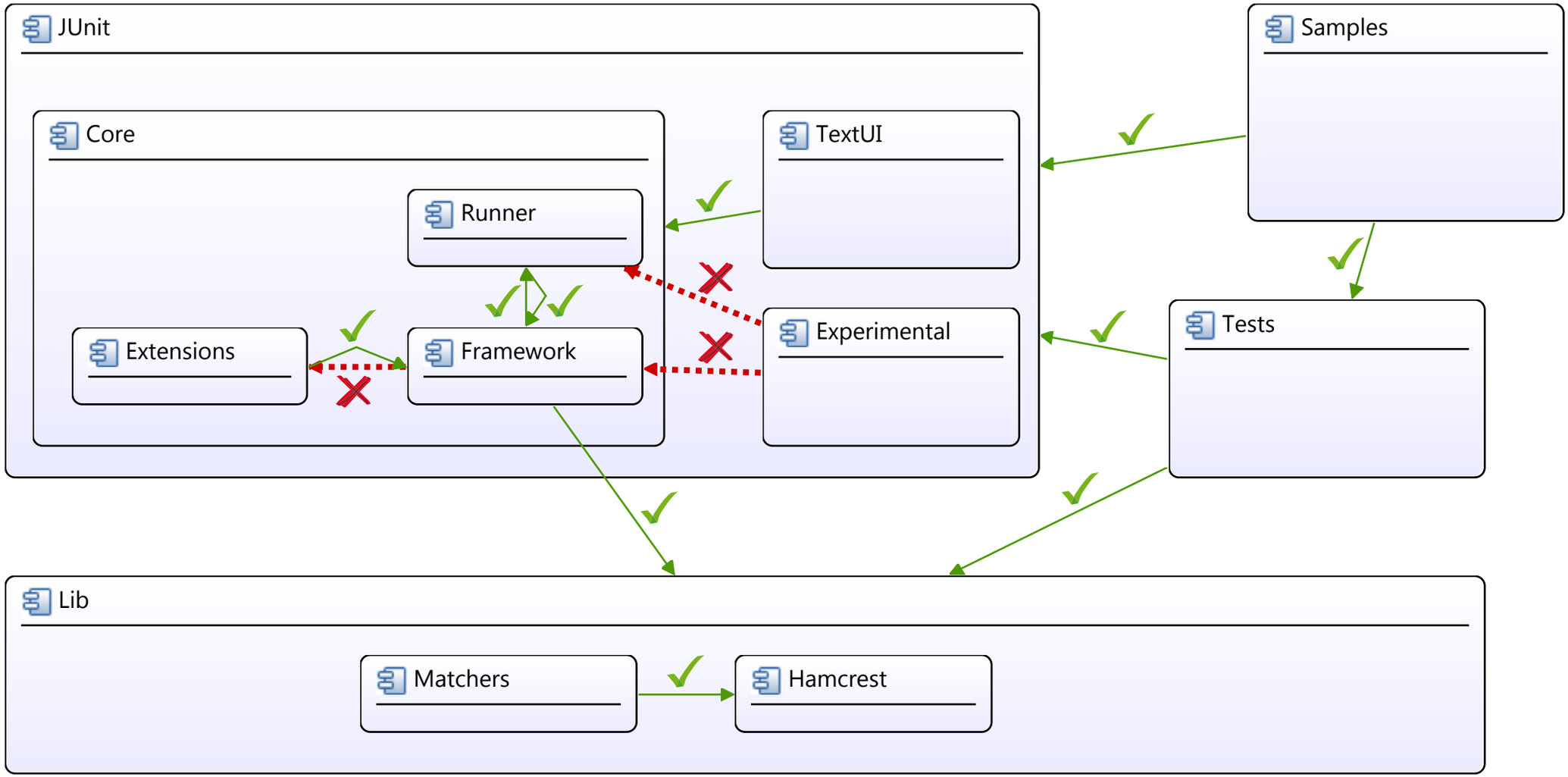


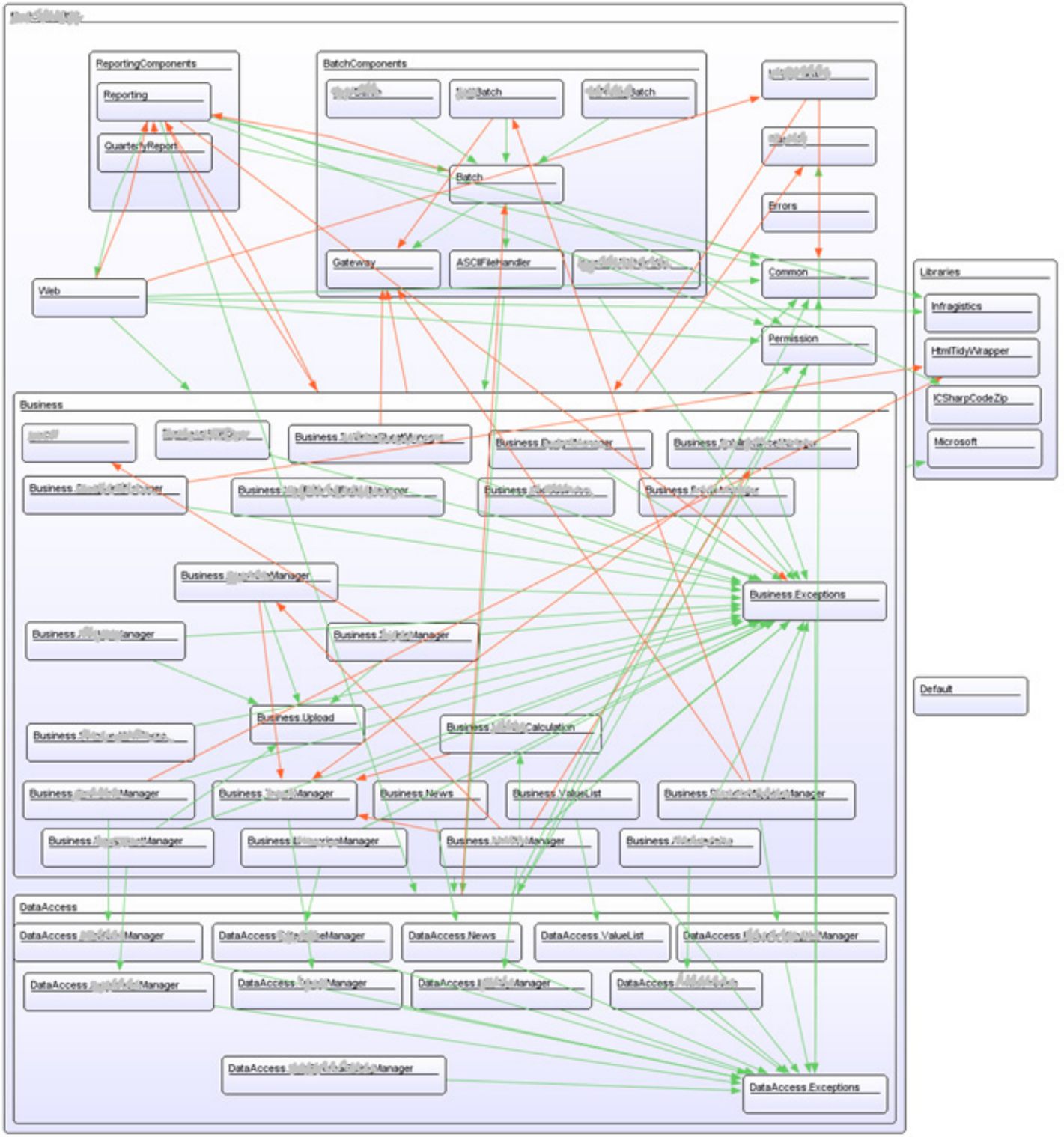
- 35 Trigger
- entführte Tabellen/verlorene User
- SQL-Injection
- Impact-Analyse auf Zuruf

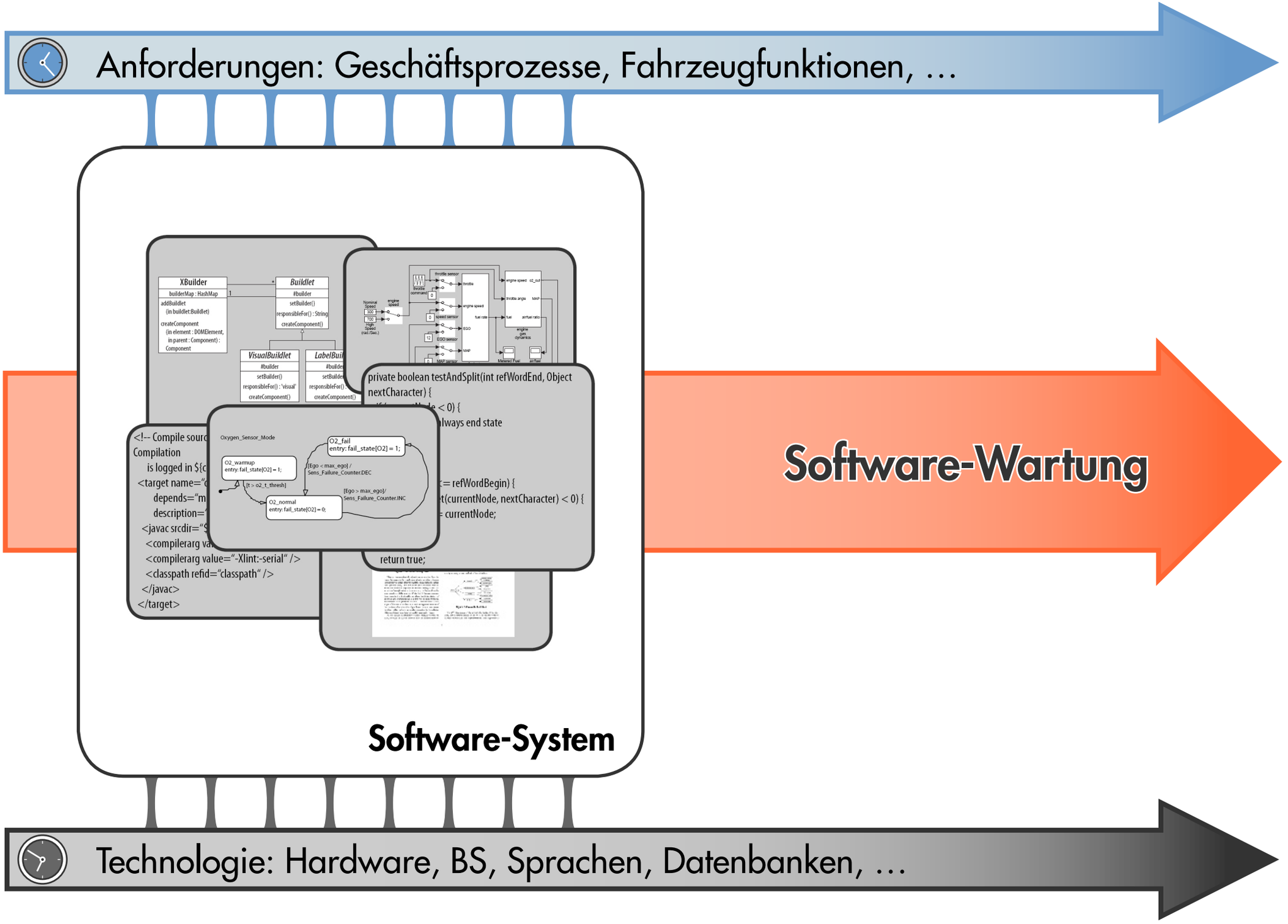


- Tests, die kontinuierlich fehlschlagen
- Unit-Tests, die nichts testen
- Mangelnde Testbarkeit
- Capture&Replay-Werkzeuge

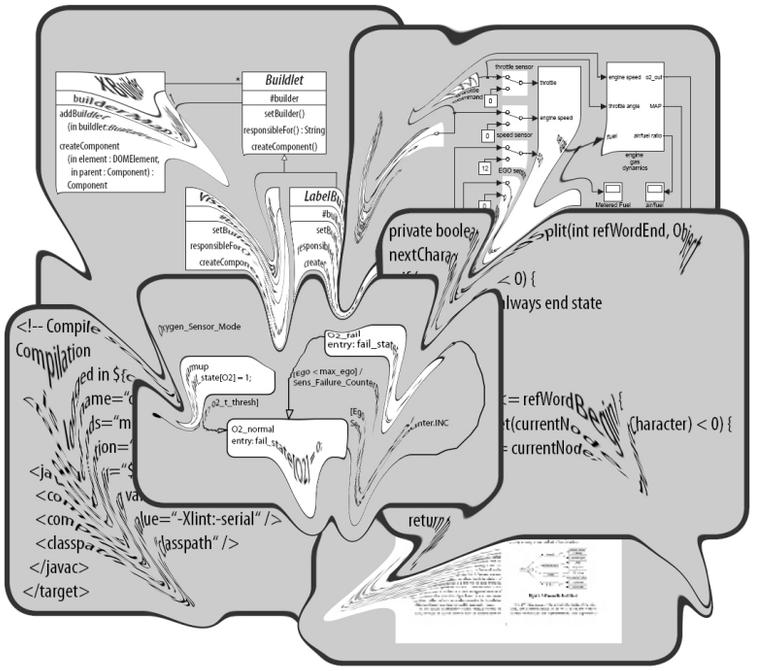

JUnit
Core
Runner
Extensions
Framework
TextUI
Experimental
Samples
Tests
Lib
Matchers
Hamcrest


JUnit
Core
Runner
Extensions
Framework
TextUI
Experimental
Samples
Tests
Lib
Matchers
Hamcrest


ReportingComponents
Reporting
QuarterlyReport
BatchComponents
Batch
Gateway
ASCIIFileHandler
Errors
Common
Permission
Web
Libraries
Infragistics
HtmlTidyWrapper
ICSharpCodeZip
Microsoft
Business
Business Exceptions
Business Upload
Default
DataAccess
DataAccess News
DataAccess ValueList
DataAccess Exceptions


Anforderungen: Geschäftsprozesse, Fahrzeugfunktionen, …
XBuilder
builderMap : HashMap
addBuildlet
(in buildlet:Buildlet)
createComponent
(in element : DOMElement,
in parent : Component) :
Component
Buildlet
#builder
setBuilder()
responsibleFor() : String
createComponent()
VisualBuildlet
#builder
setBuilder()
responsibleFor() : 'visual'
createComponent()
private boolean testAndSplit(int refWordEnd, Object
nextCharacter) {
always end state
= refWordBegin) {
t(currentNode, nextCharacter) < 0) {
= currentNode;
return true;
<!-- Compile sourc
Compilation
is logged in ${c
<target name="c
depends="m
description="
<javac srcdir="S
<compilerarg va
<compilerarg value="-Xlint:-serial" />
<classpath refid="classpath" />
</javac>
</target>
Oxygen_Sensor_Mode
O2_warmup
entry: fail_state[O2] = 1;
O2_fail
entry: fail_state[O2] = 1;
O2_normal
entry: fail_state[O2] = 0;
[t > o2_t_thresh]
[Ego < max_ego] /
Sens_Failure_Counter.DEC
[Ego > max_ego]/
Sens_Failure_Counter.INC
Software-System
Software-Wartung
Technologie: Hardware, BS, Sprachen, Datenbanken, …

Anforderungen: Geschäftsprozesse, Fahrzeugfunktionen, …



Software-System

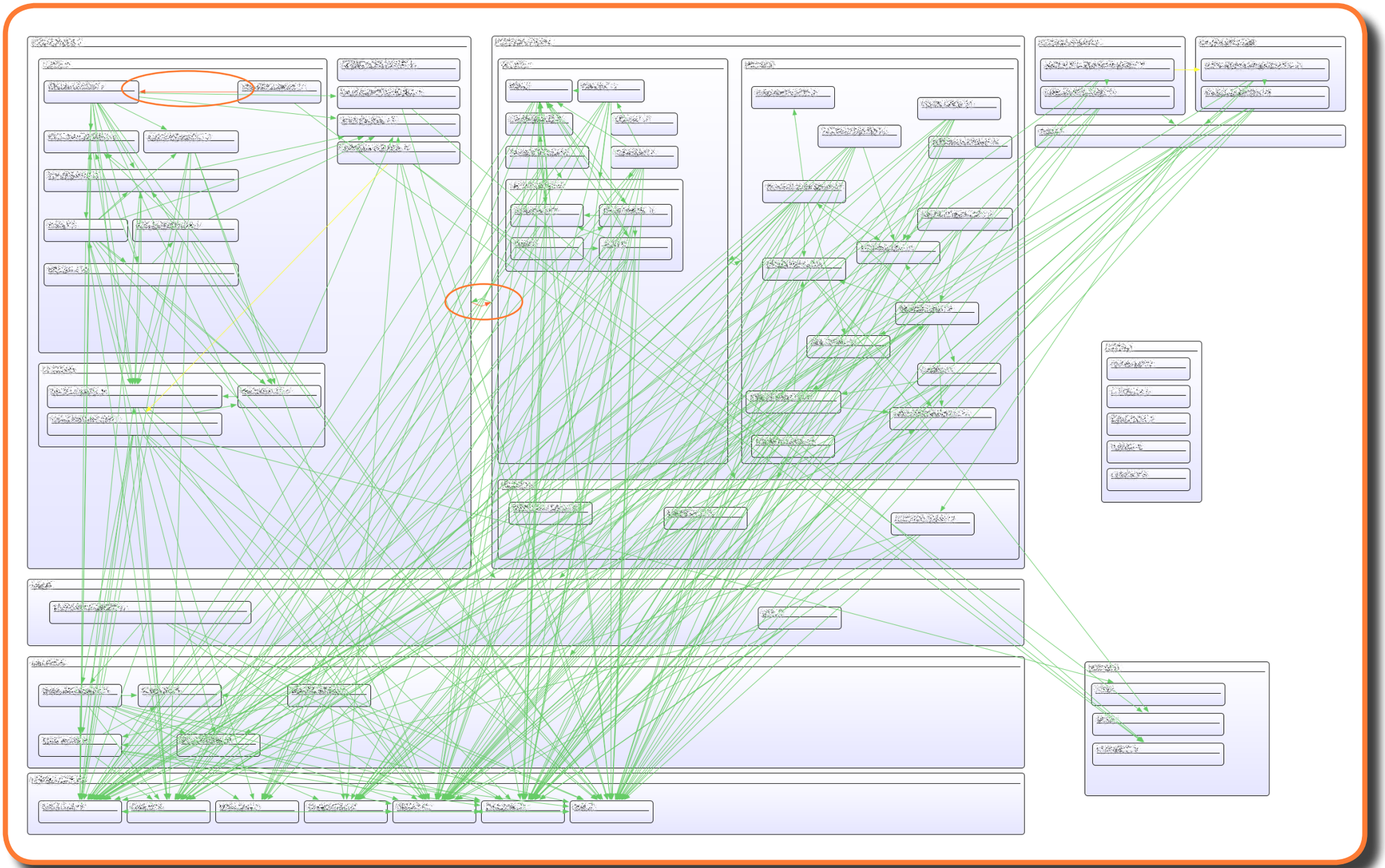
Software-Wartung

Technologie: Hardware, BS, Sprachen, Datenbanken, …

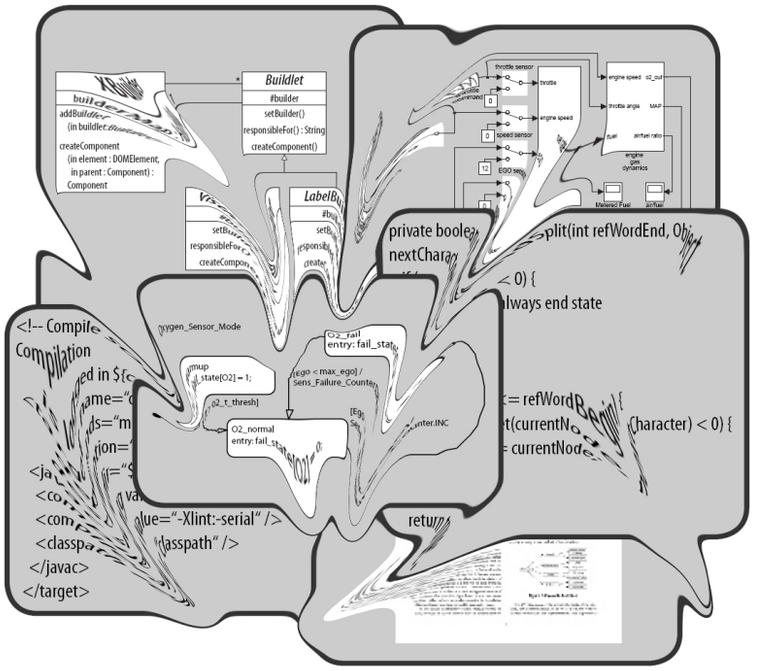
»Programs, like people, get old«*

- »Software aging … in all *successful* products«
- Ursachen:
  - mangelnde Bewegung
  - unkundige Behandlung
- Symptome:
  - Gewichtszunahme
  - sinkende Performance und Zuverlässigkeit
- Prävention:
  - Vorausplanen (Qualität, Flexibles Design)
  - Aufzeichnung aufbewahren (Dokumentation)
  - zweite Meinung einholen (Review)
- Heilkunde:
  - Alterungsprozess aufhalten, Nachdokumentation
  - Restrukturieren, Amputation

* D. L. Parnas, Software Aging, ICSE, 1994



Anforderungen: Geschäftsprozesse, Fahrzeugfunktionen, …

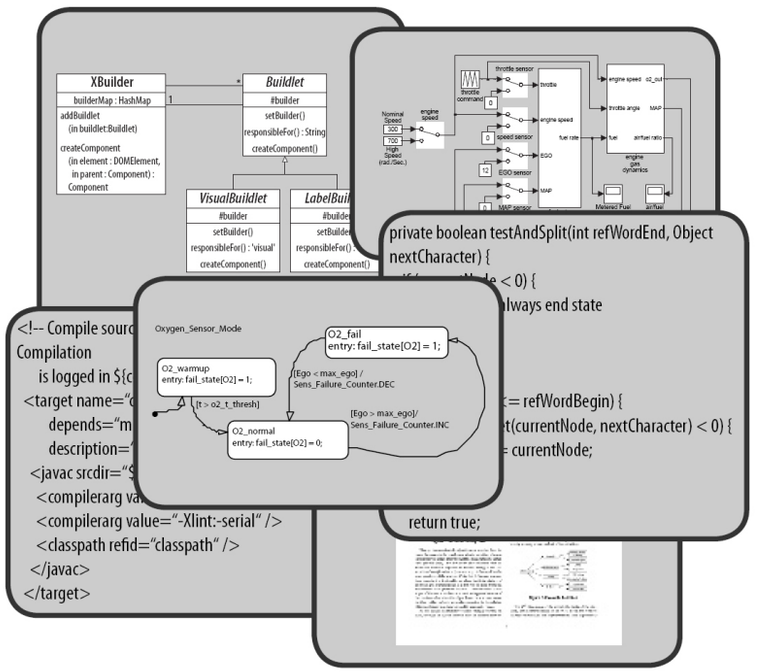


Software-System

Software-Wartung

Technologie: Hardware, BS, Sprachen, Datenbanken, …

Anforderungen: Geschäftsprozesse, Fahrzeugfunktionen, …

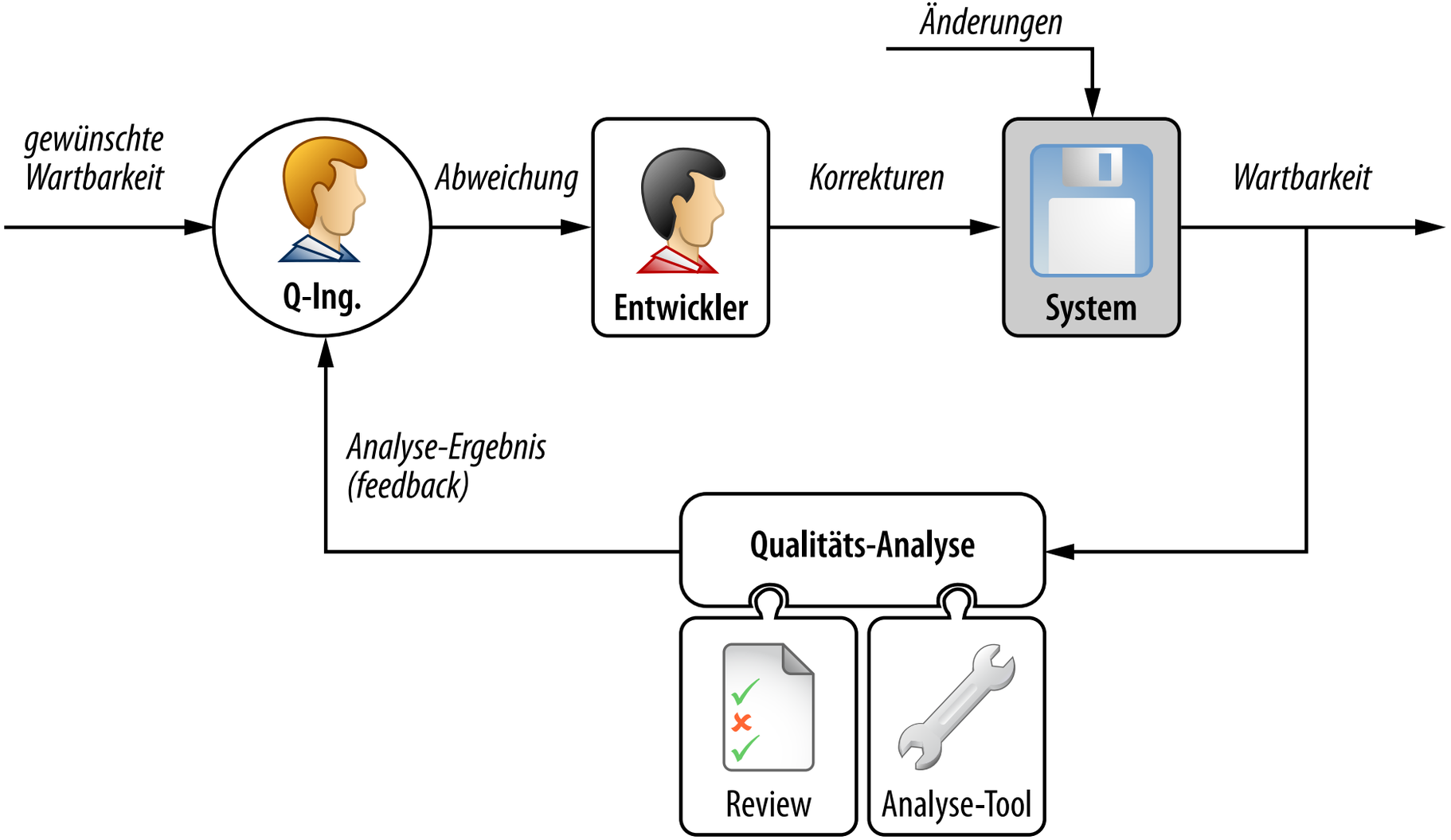


Software-System

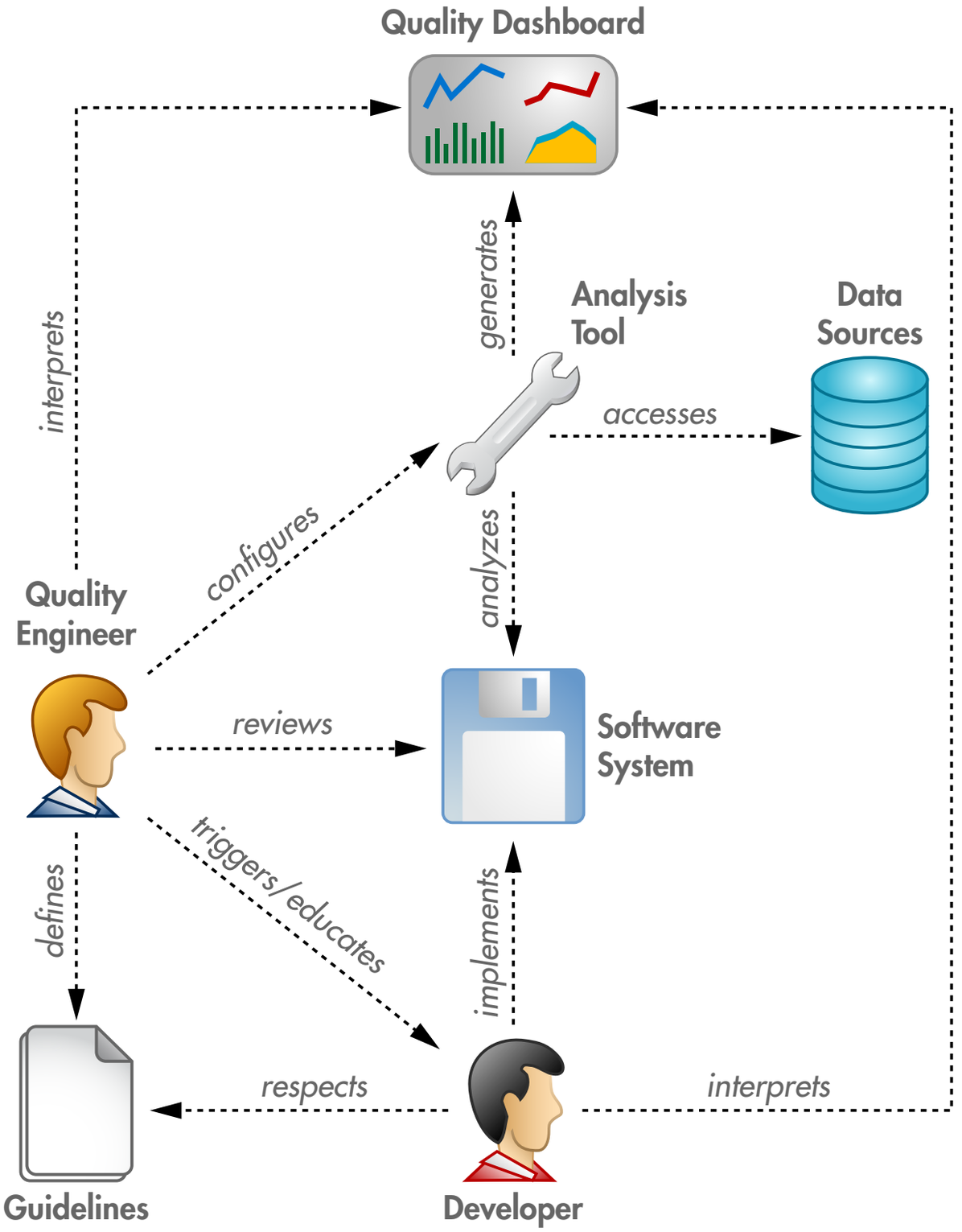
Software-Wartung

Kontinuierliches Qualitäts-Controlling

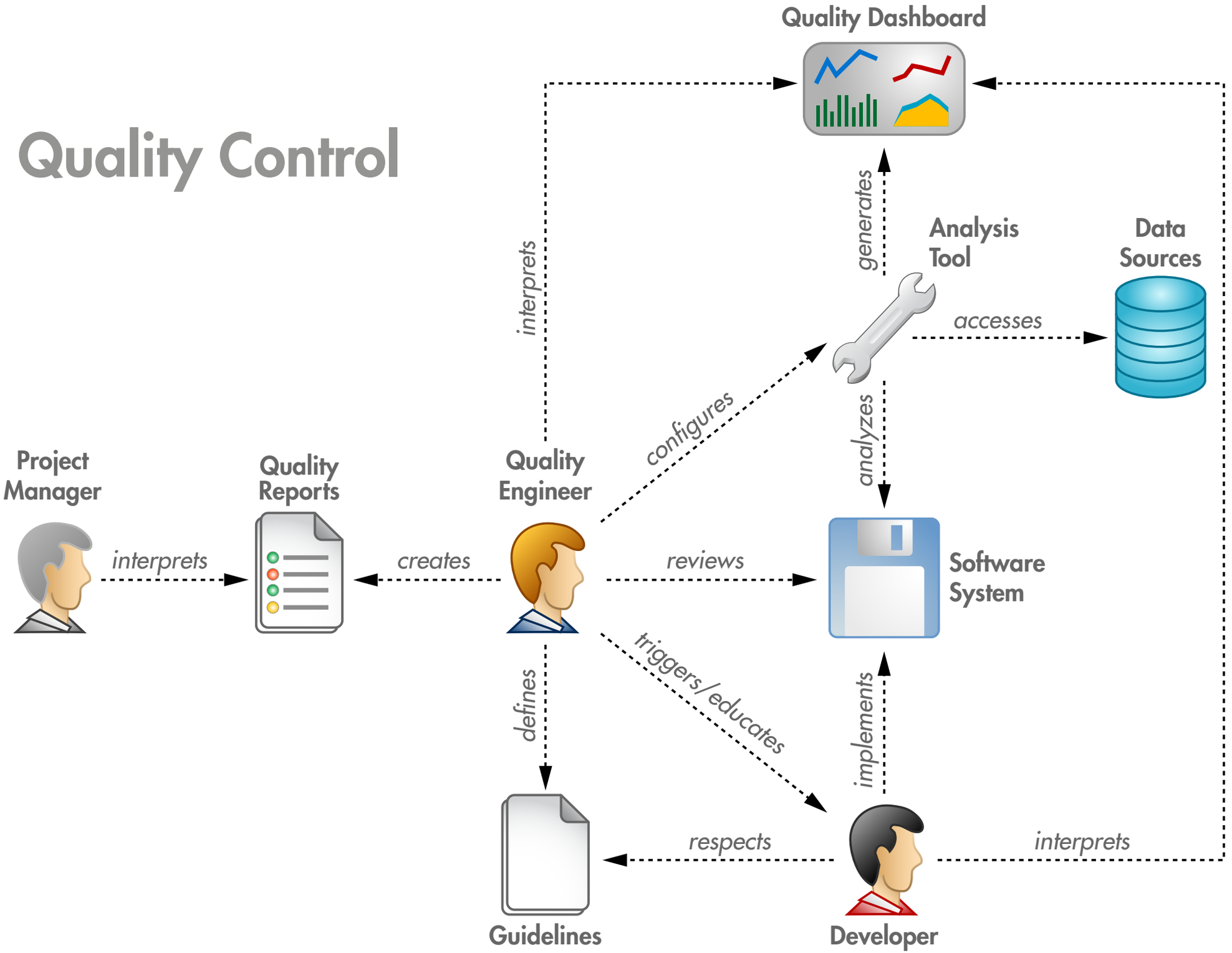
Technologie: Hardware, BS, Sprachen, Datenbanken, …


Änderungen
gewünschte Wartbarkeit
Q-Ing.
Abweichung
Entwickler
Korrekturen
System
Wartbarkeit
Analyse-Ergebnis (feedback)
Qualitäts-Analyse
Review
Analyse-Tool

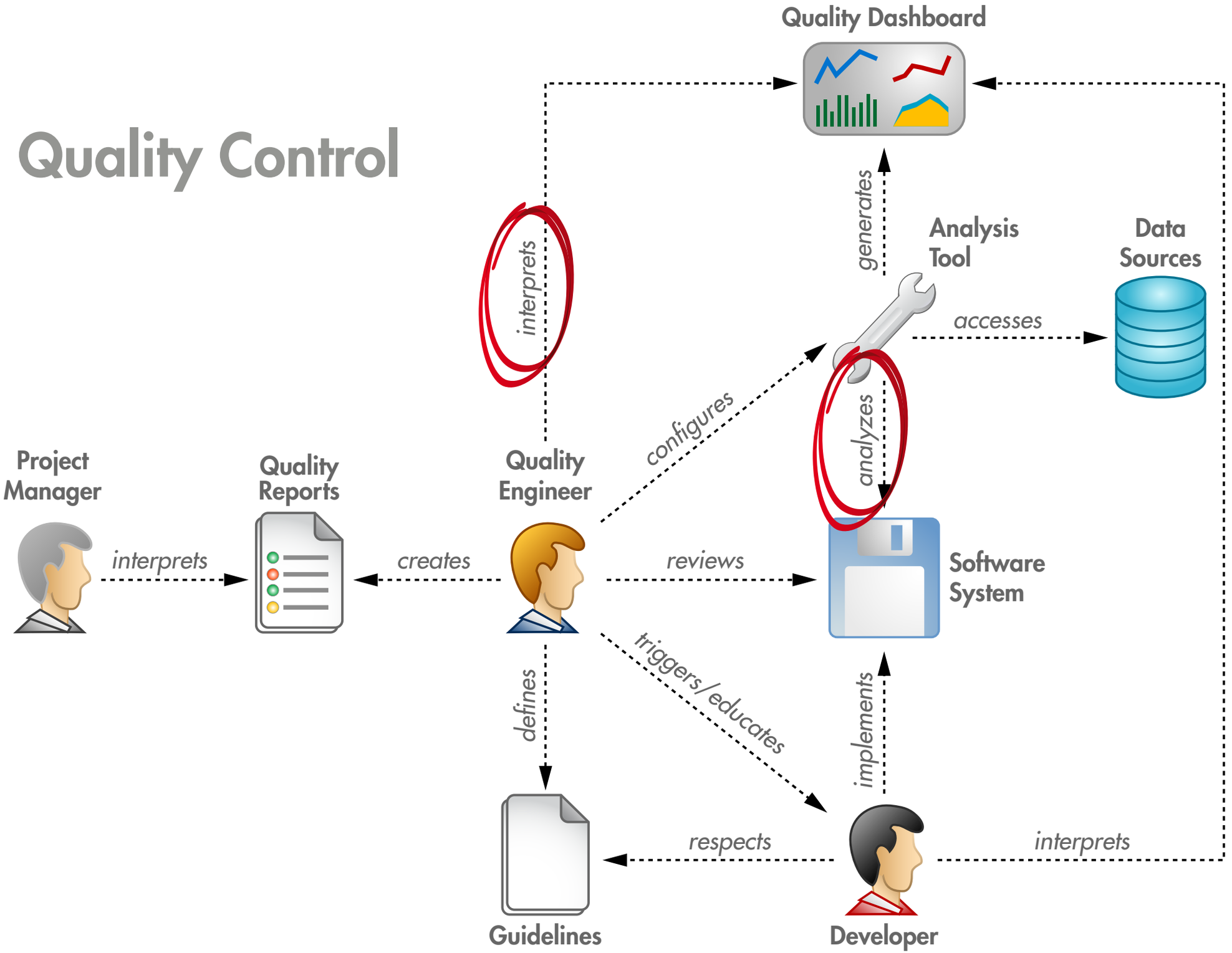
Quality Control



Quality Control



Quality Control



Trugschluss: Messbar == Relevant

CycC
CR
LCOM
No. Identifiers
CBO
No. Classes
Cohesion
Clone Coverage
NOPM
Assertion Density
Line Coverage
NOPA
Defect Density
Statements
Statement Coverage
Flow
DIT
Abstractness
Branch Coverage
NOC
Halstead
Coupling
Stability
No. Methods
Blow Up
Fan Out
LOC
NOAM
RFC
WMC
Fan In
MCDC
RFSS

OO Design Metrics

Fan-In

Fan-Out

Coupling

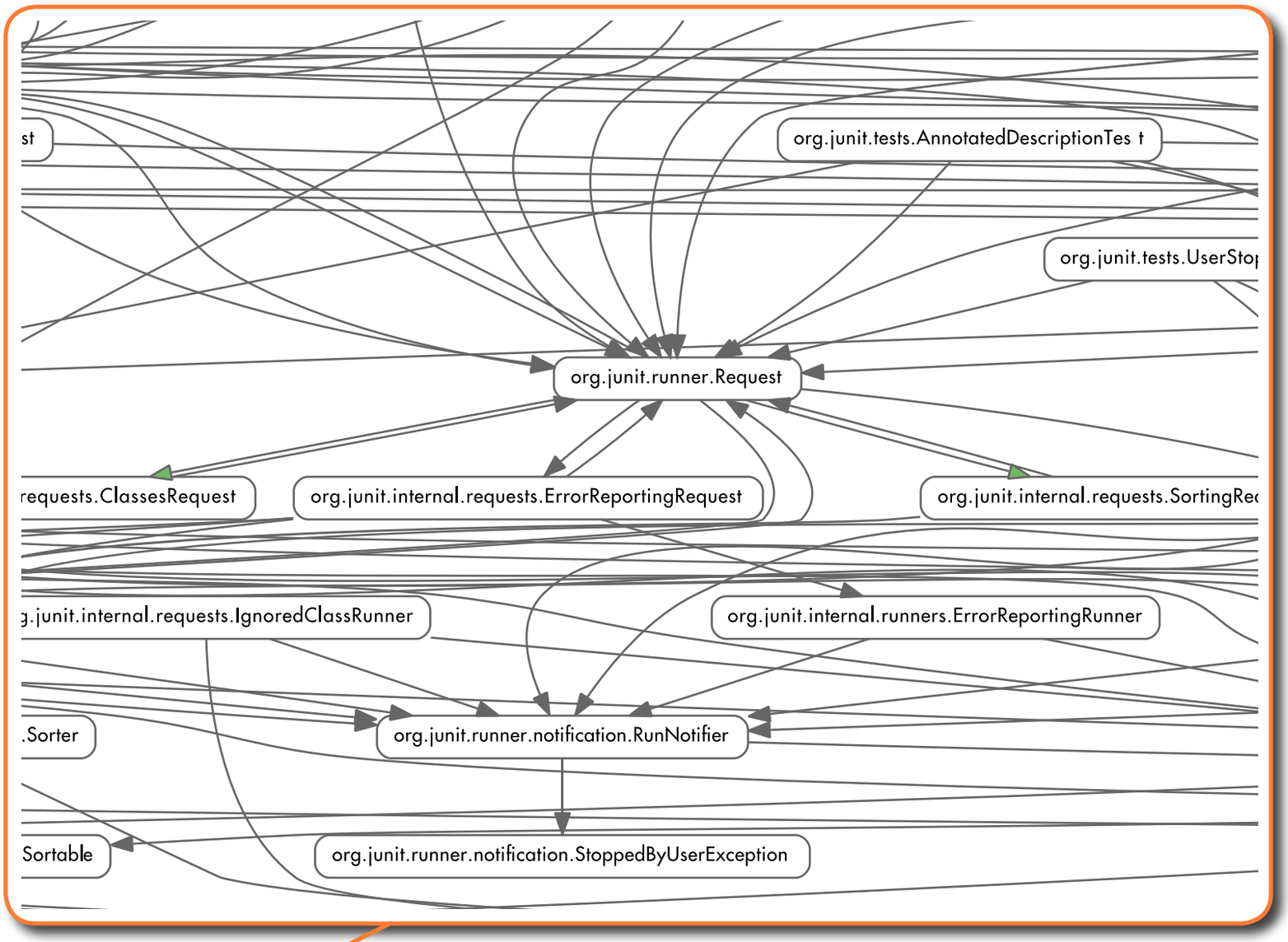
Cohesion

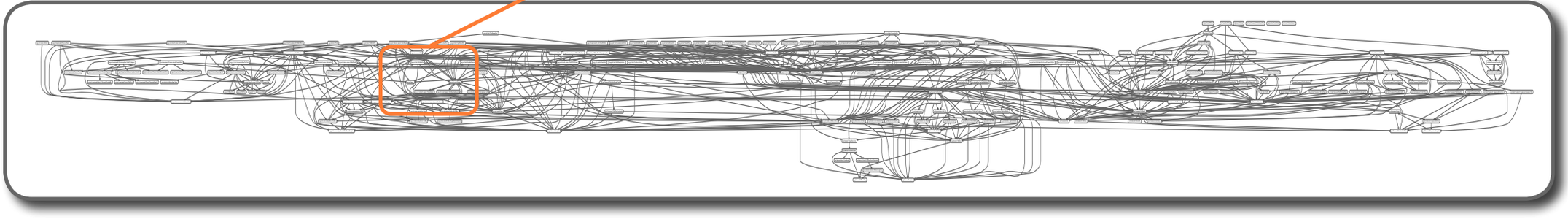
Stability

Depth of Inheritance

Children

…





OO Design Metrics

Fan-In 16

Fan-Out 4

Coupling

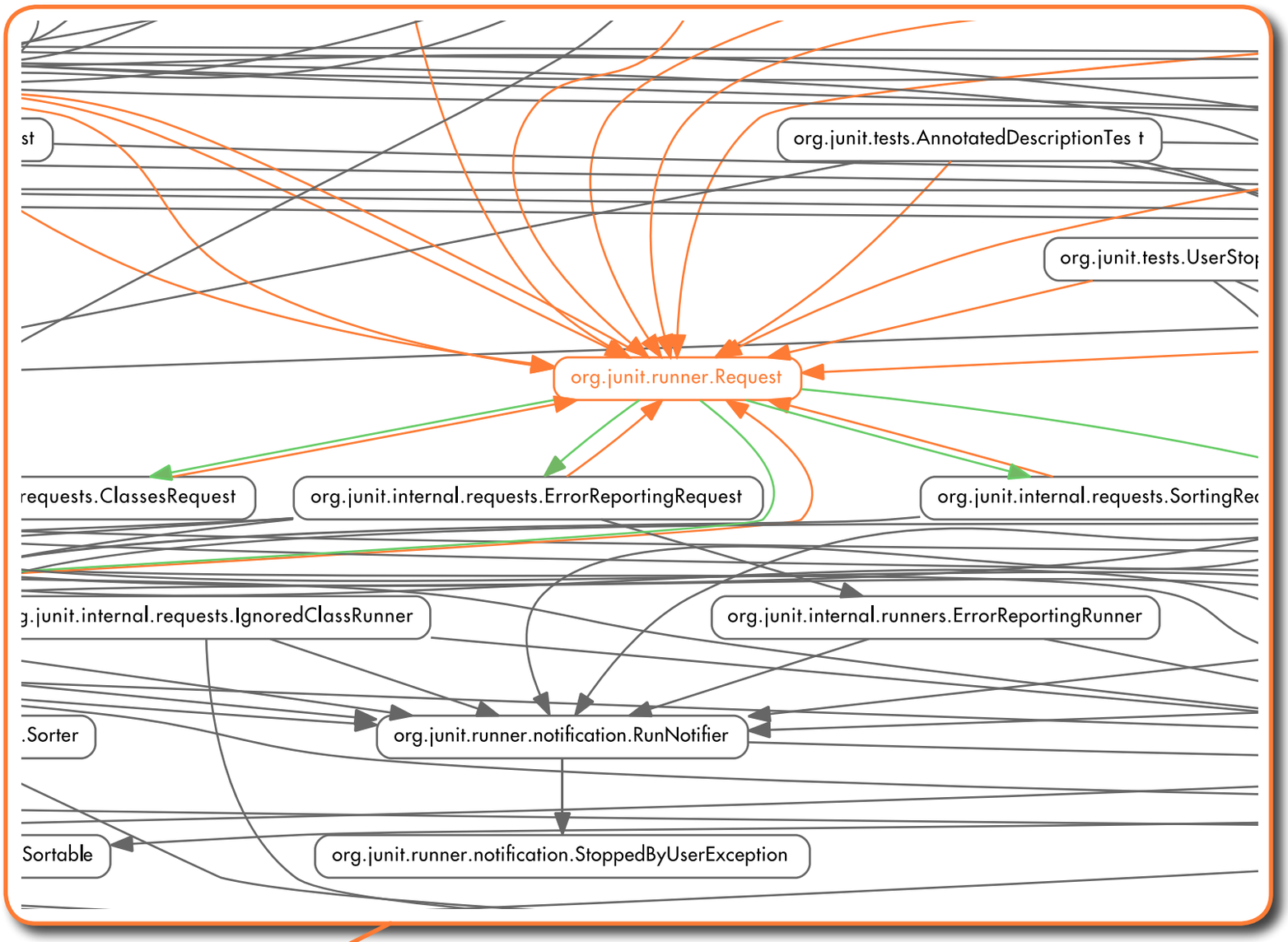
Cohesion

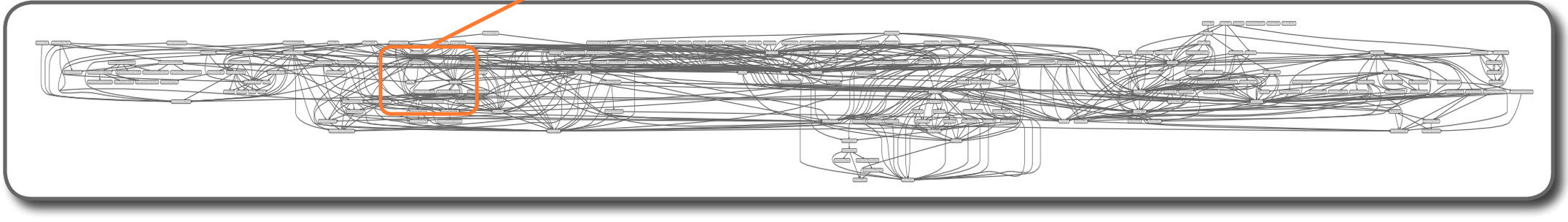
Stability

Depth of Inheritance

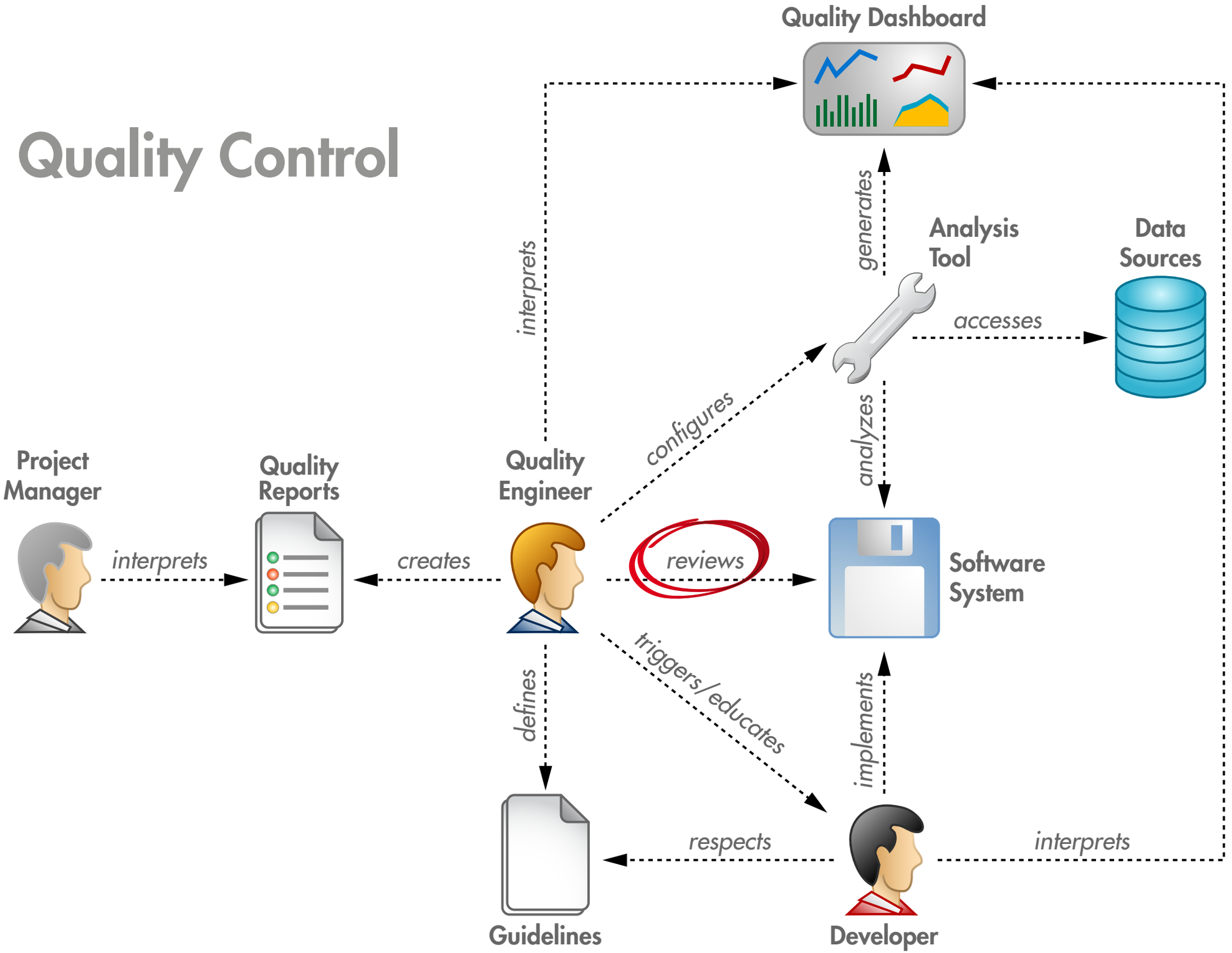
Children

…





Quality Control

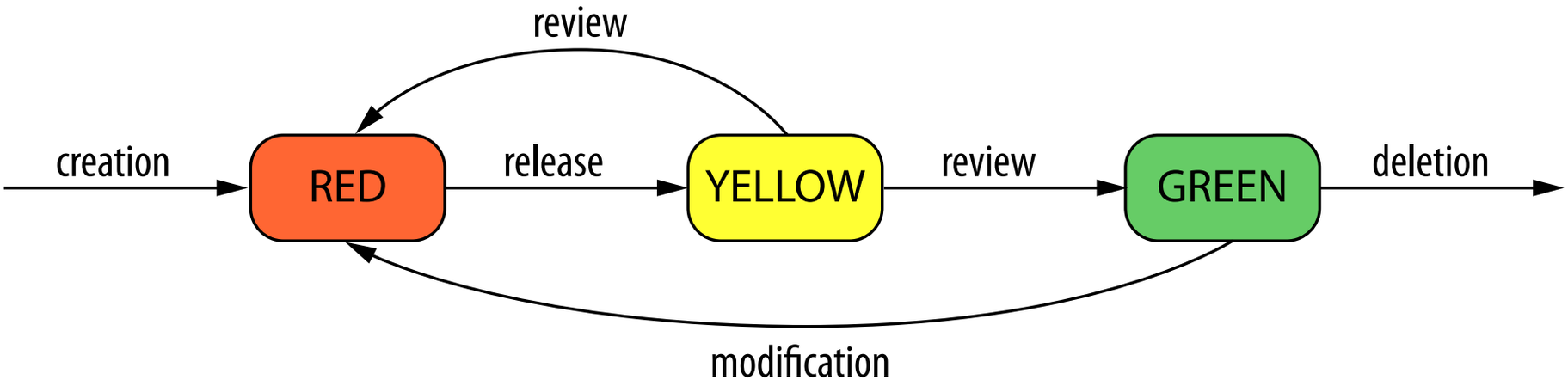


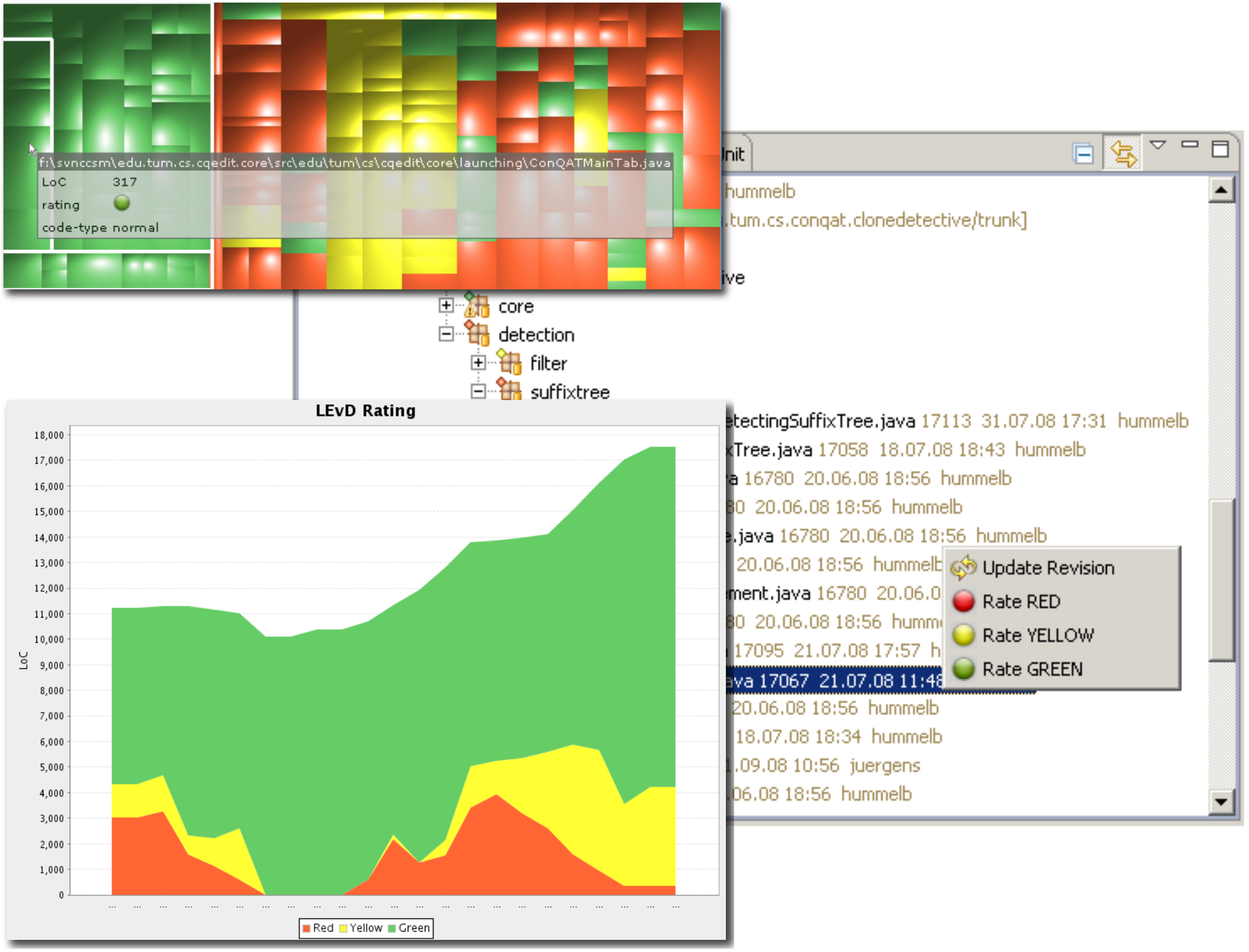
Manuelle Reviews

- viele relevante Qualitätseigenschaften sind nicht automatisch analysierbar
  - Bezeichner
  - Kommentare
  - sinnvolle Verwendung von Datenstrukturen und Algorithmen
  - Konsistenz der Fehlerbehandlung
  - logische Redundanz
  - …
- diese Eigenschaften bedürfen der manuelle Analyse
- manuelle Analysen werden in der Praxis nur spärlich eingesetzt
- manuelle Analysen sollten soweit möglich durch automatische Analysen unterstüzt werden

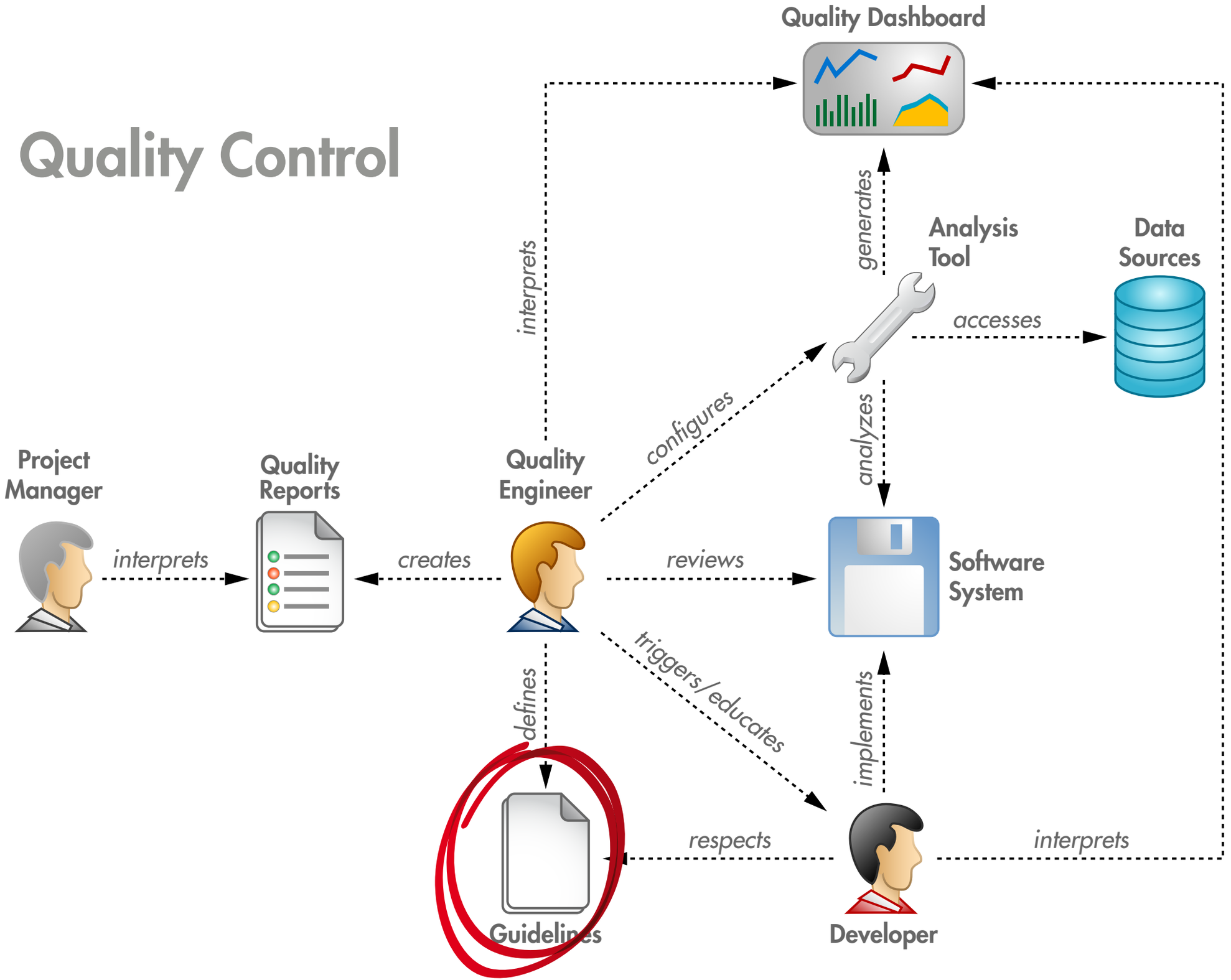
Reviews bei der ConQAT-Entwicklung

- flächendeckendes Review aller Änderungen auf Basis des Ampelzustands der angepassten Dateien
- Sammlung der Issues als TODO-Tags im Code




f:\svnccsm\edu.tum.cs.cqedit.core\src\edu\tum\cs\cqedit\core\launching\ConQATMainTab.java
LoC 317
rating
code-type normal
core
detection
filter
suffixtree
hummelb
.tum.cs.conqat.clonedetective/trunk]
LEvD Rating
LoC
18,000
17,000
16,000
15,000
14,000
13,000
12,000
11,000
10,000
9,000
8,000
7,000
6,000
5,000
4,000
3,000
2,000
1,000
0
Red Yellow Green
tectingSuffixTree.java 17113 31.07.08 17:31 hummelb
Tree.java 17058 18.07.08 18:43 hummelb
16780 20.06.08 18:56 hummelb
20.06.08 18:56 hummelb
.java 16780 20.06.08 18:56 hummelb
20.06.08 18:56 hummelb
ment.java 16780 20.06.0
20.06.08 18:56 hummelb
17095 21.07.08 17:57 h
va 17067 21.07.08 11:48
20.06.08 18:56 hummelb
18.07.08 18:34 hummelb
1.09.08 10:56 juergens
06.08 18:56 hummelb
Update Revision
Rate RED
Rate YELLOW
Rate GREEN

Quality Control

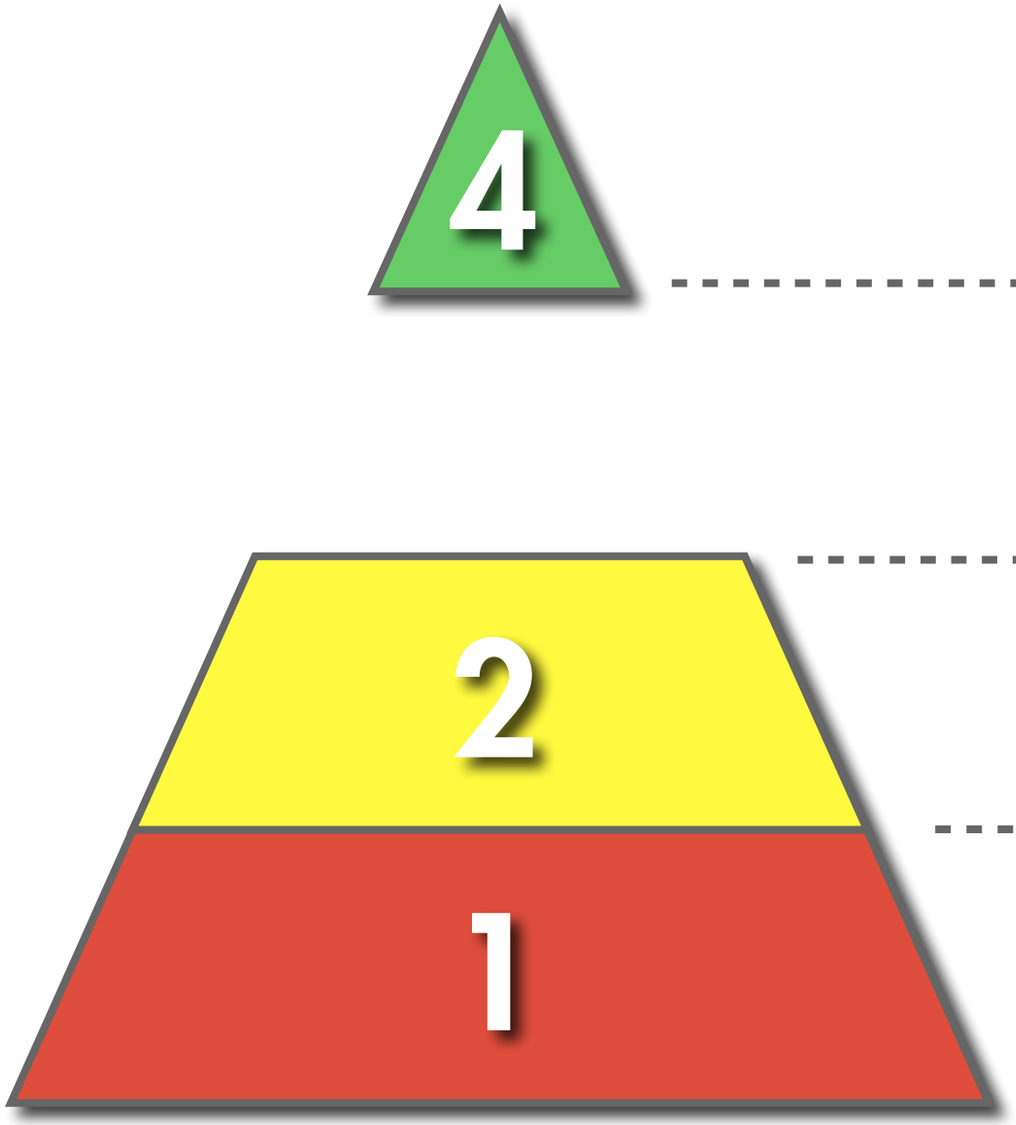


Messung erfordert Ziele

Keine Defizite

Egal

Messung erfordert Ziele

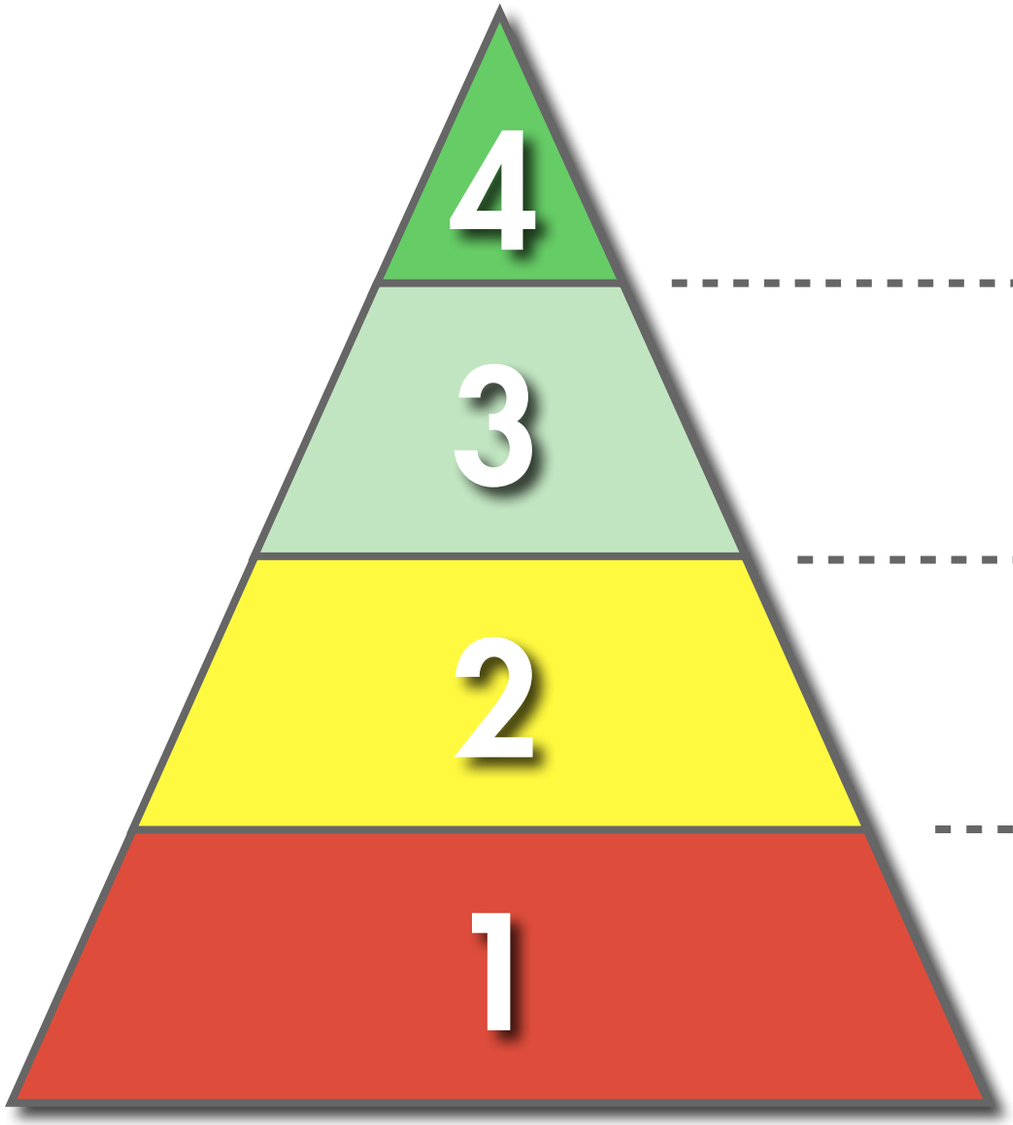


Keine Defizite

Keine neue Defizite

Egal

Messung erfordert Ziele



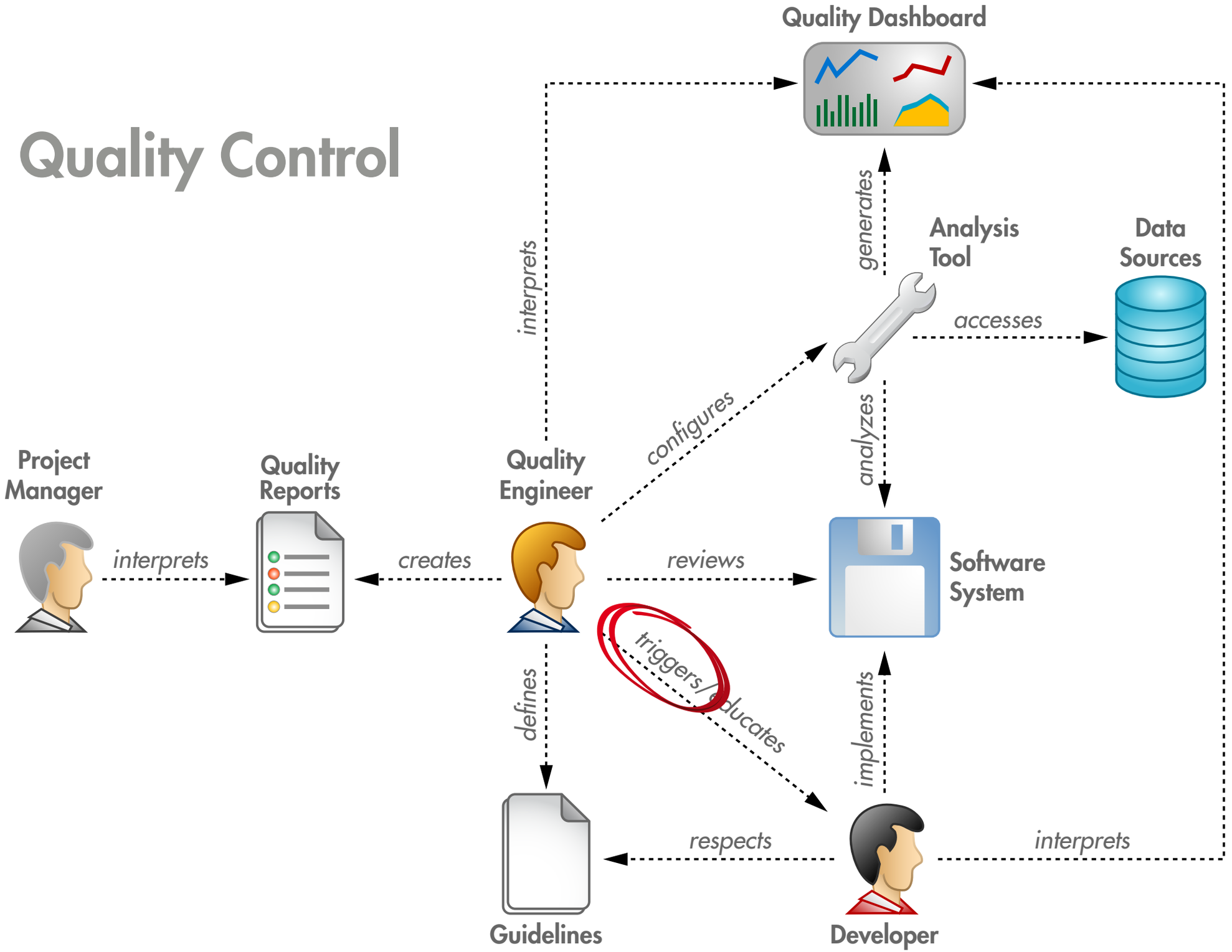
Keine Defizite

Keine Defizite in geändertem Code

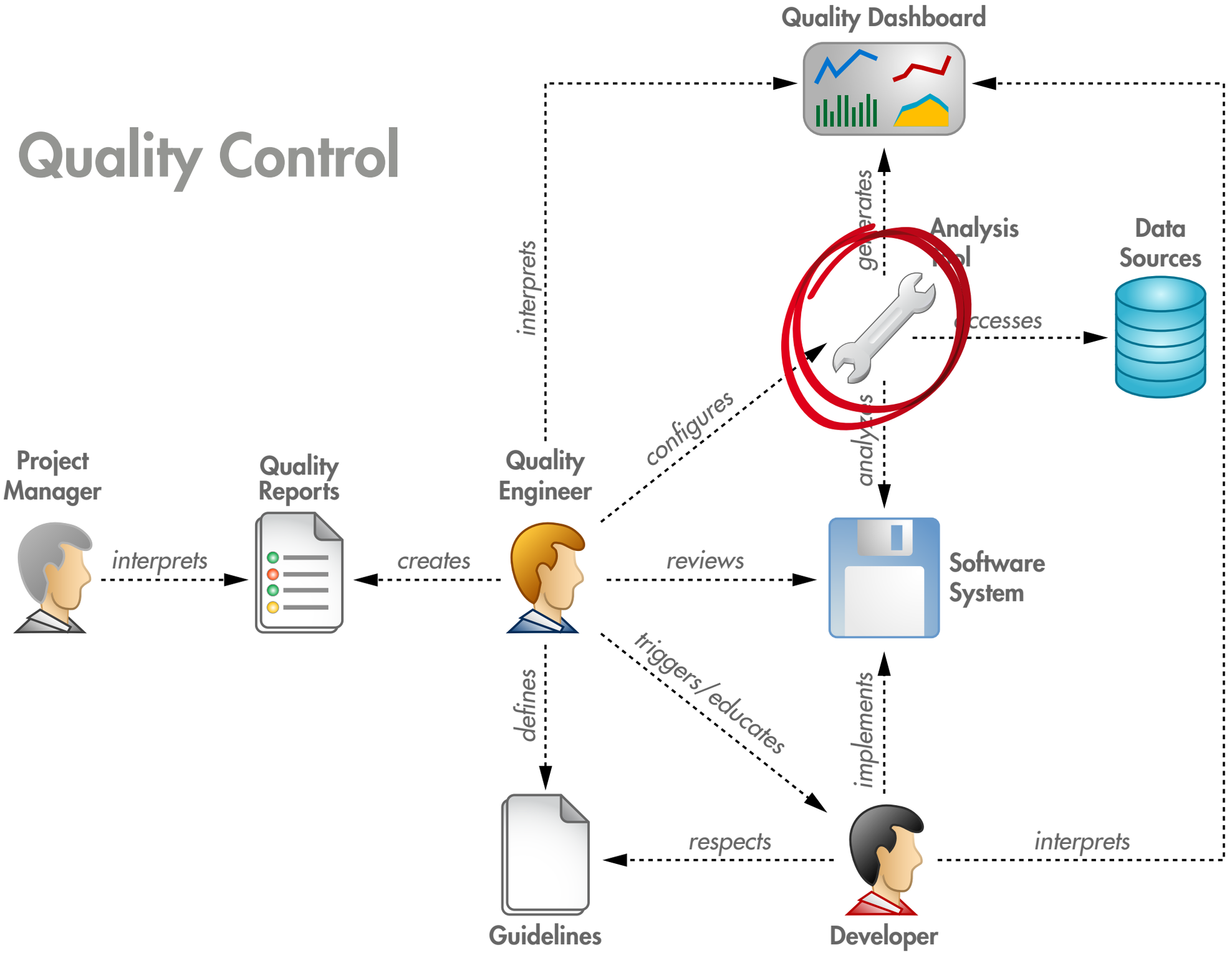
Keine neue Defizite

Egal

Quality Control

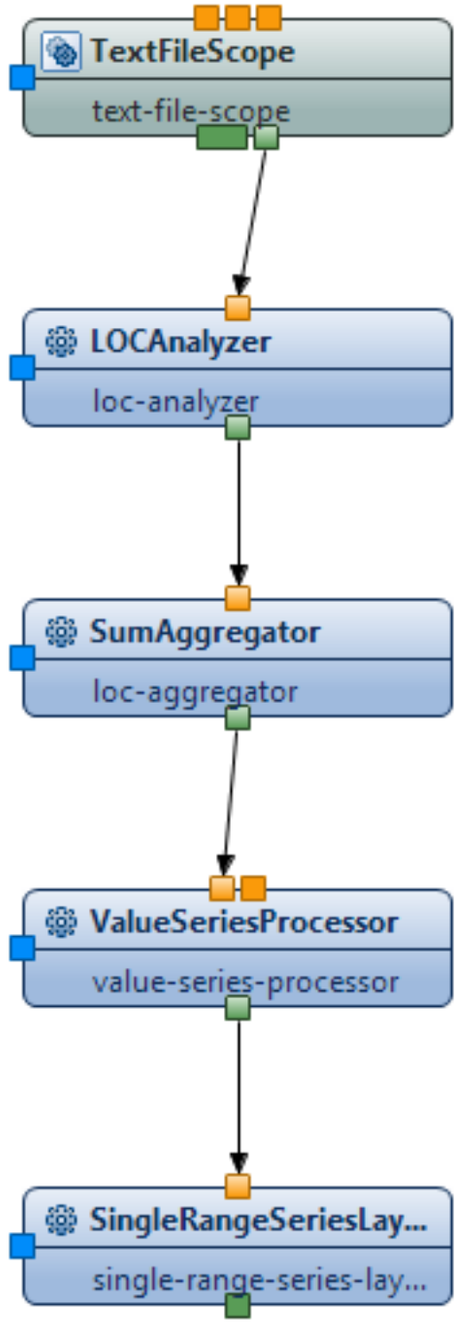


Quality Control



ConQAT

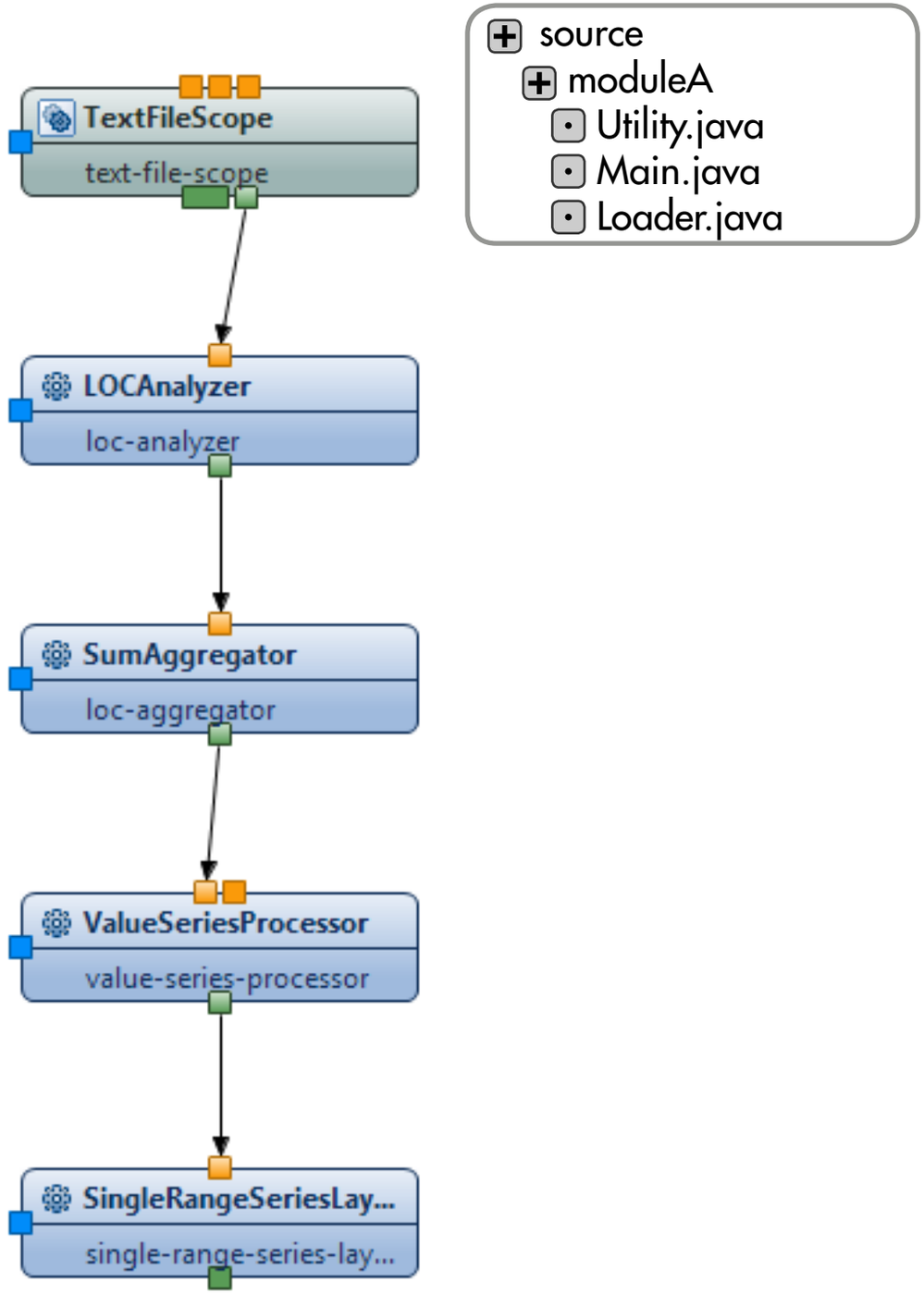
- graphische DSL zur Spezifikation der Analysekonfiguration
- Prozessoren beschreiben Analyseoperationen
- Kanten beschreiben Datenfluss
- Blöcke ermöglichen die Wiederverwendung von Analysefunktionalität



http://www.conqat.org

ConQAT

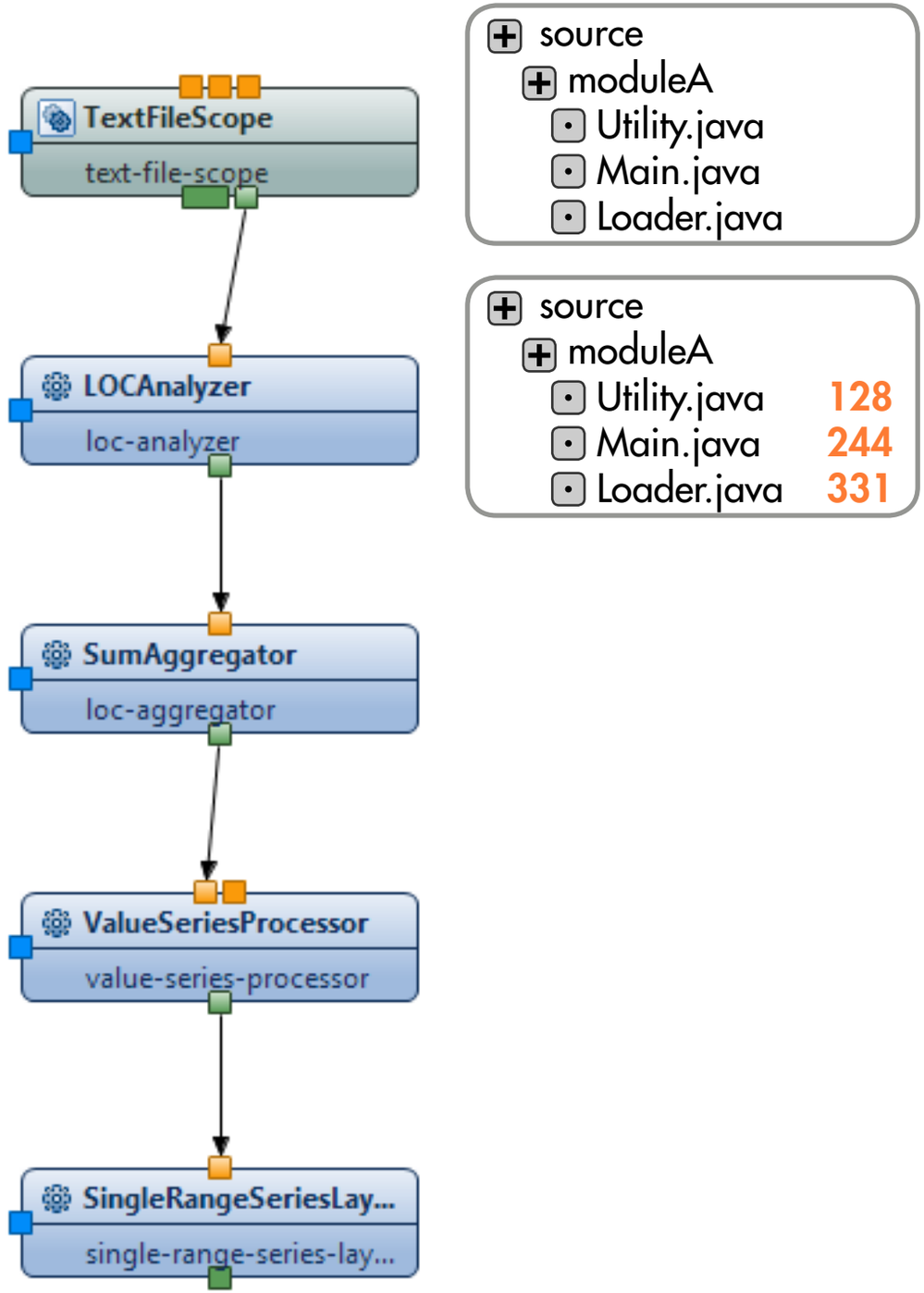
- graphische DSL zur Spezifikation der Analysekonfiguration
- Prozessoren beschreiben Analyseoperationen
- Kanten beschreiben Datenfluss
- Blöcke ermöglichen die Wiederverwendung von Analysefunktionalität



http://www.conqat.org

ConQAT

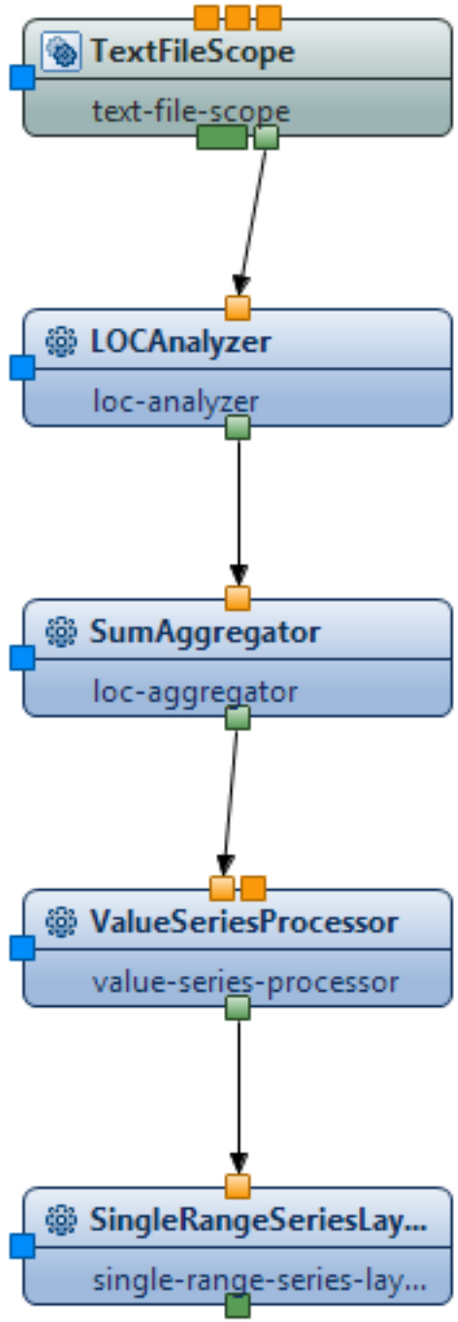
- graphische DSL zur Spezifikation der Analysekonfiguration
- Prozessoren beschreiben Analyseoperationen
- Kanten beschreiben Datenfluss
- Blöcke ermöglichen die Wiederverwendung von Analysefunktionalität

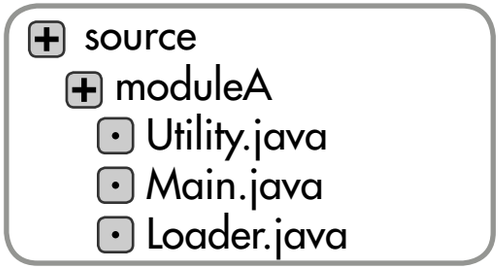


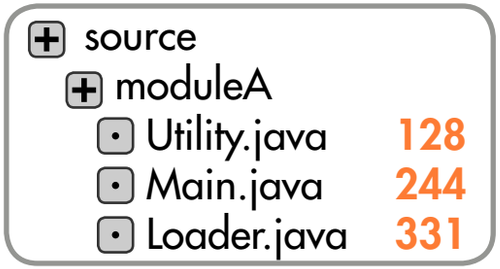
http://www.conqat.org

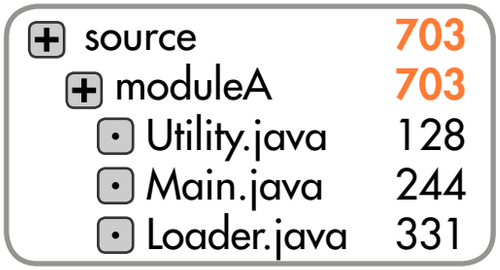
ConQAT

- graphische DSL zur Spezifikation der Analysekonfiguration
- Prozessoren beschreiben Analyseoperationen
- Kanten beschreiben Datenfluss
- Blöcke ermöglichen die Wiederverwendung von Analysefunktionalität





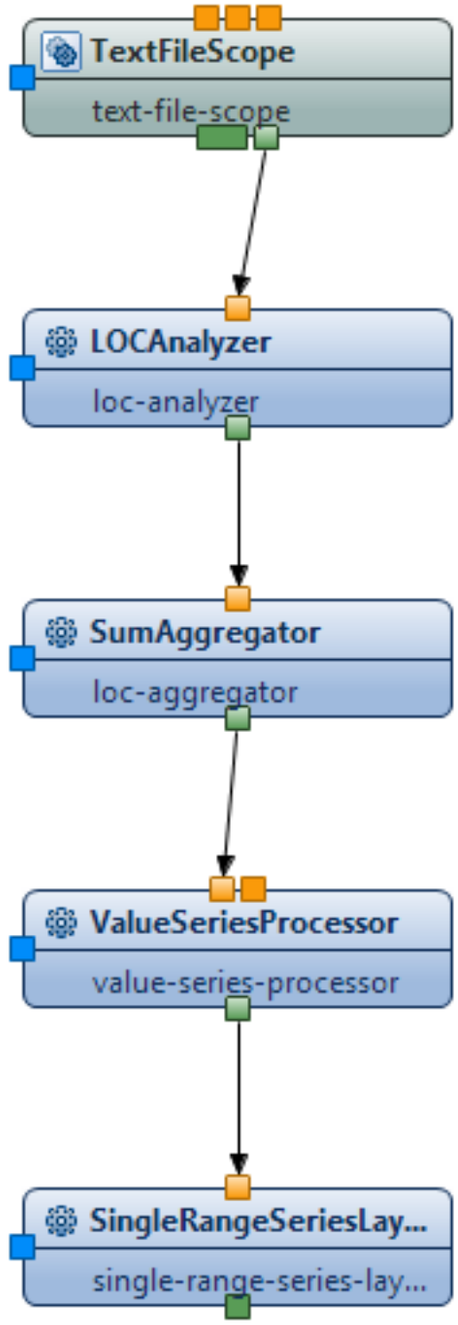


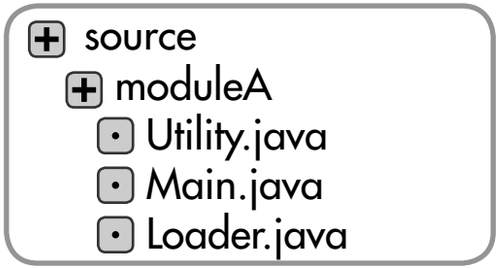


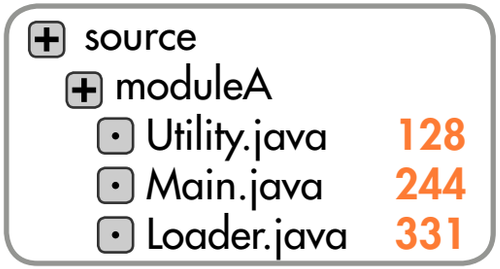
http://www.conqat.org

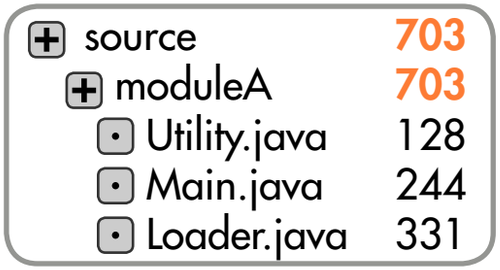
ConQAT

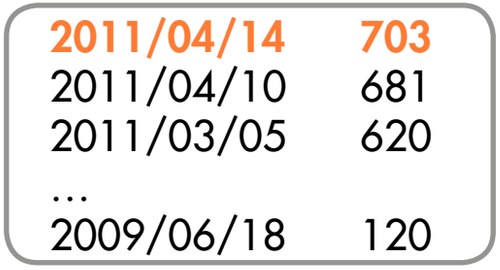
- graphische DSL zur Spezifikation der Analysekonfiguration
- Prozessoren beschreiben Analyseoperationen
- Kanten beschreiben Datenfluss
- Blöcke ermöglichen die Wiederverwendung von Analysefunktionalität







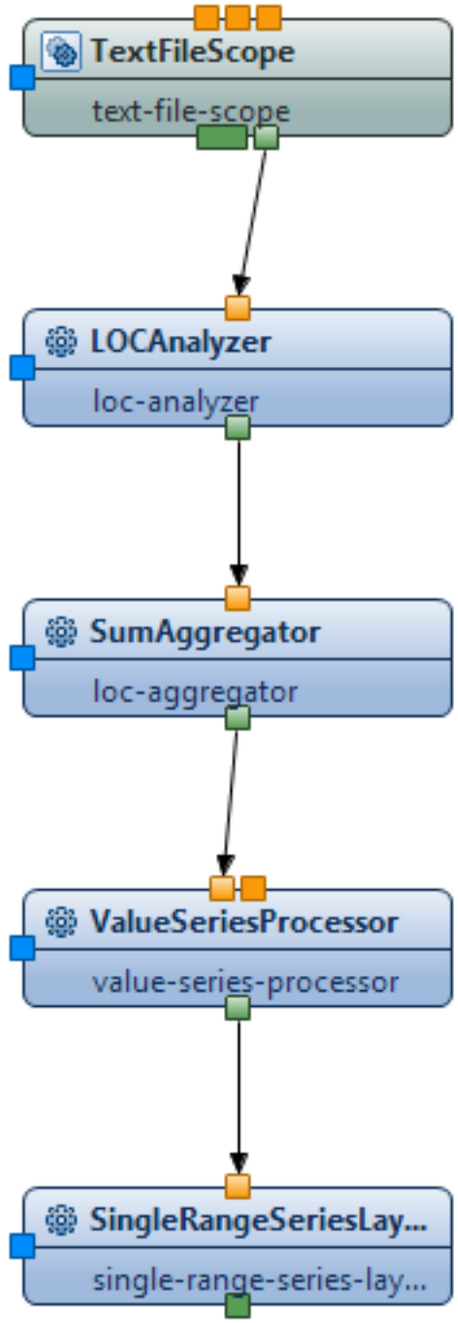


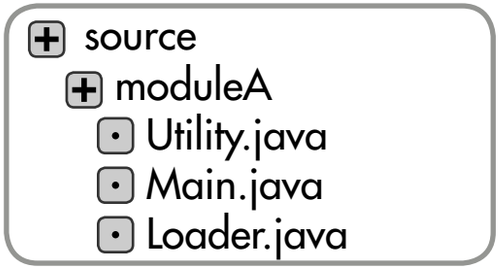


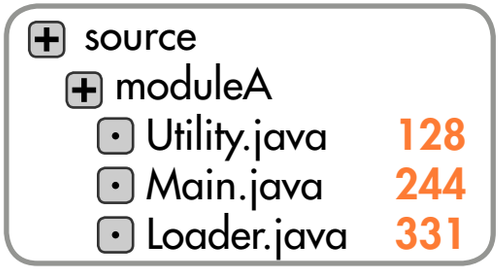
http://www.conqat.org

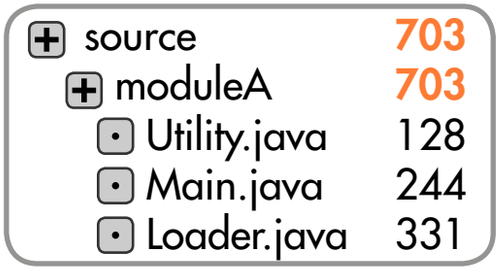
ConQAT

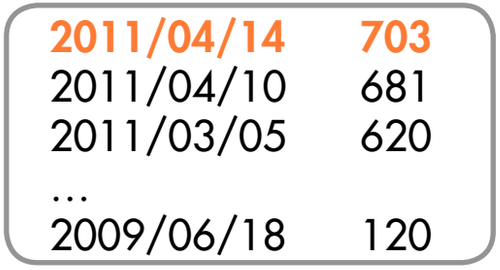
- graphische DSL zur Spezifikation der Analysekonfiguration
- Prozessoren beschreiben Analyseoperationen
- Kanten beschreiben Datenfluss
- Blöcke ermöglichen die Wiederverwendung von Analysefunktionalität

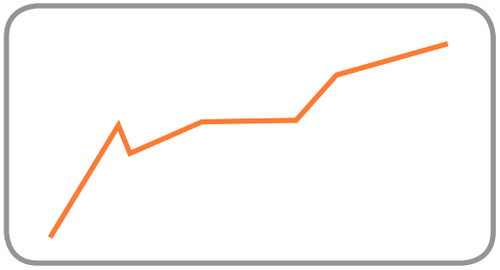






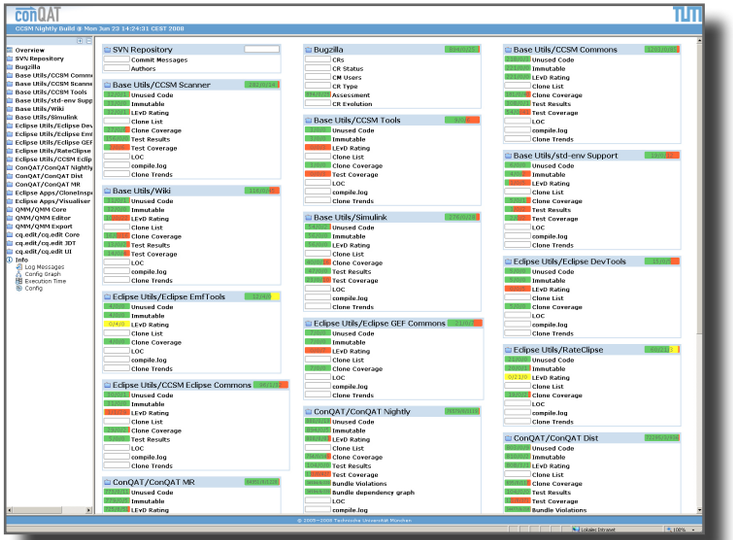


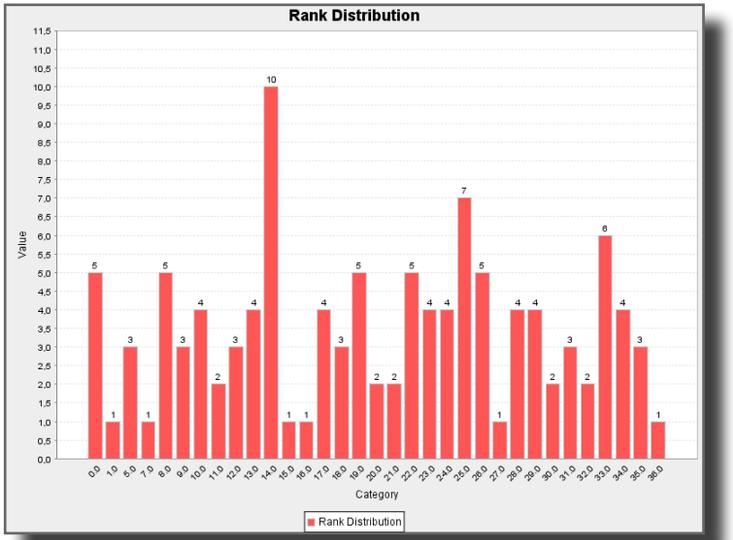


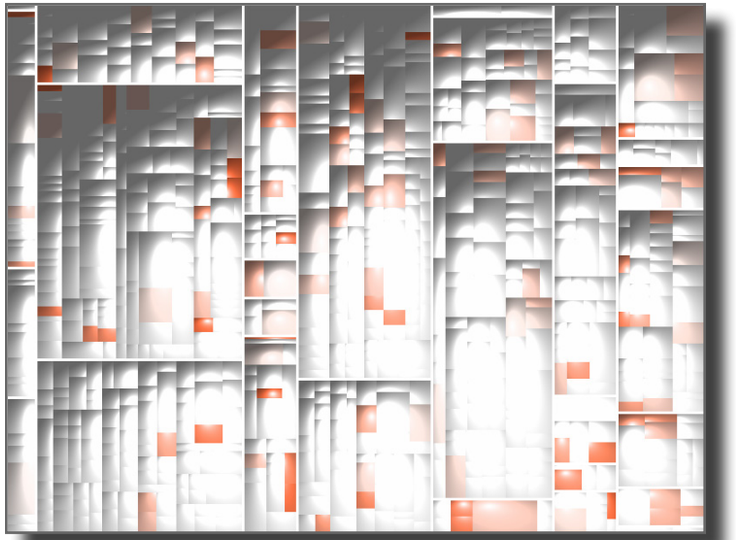


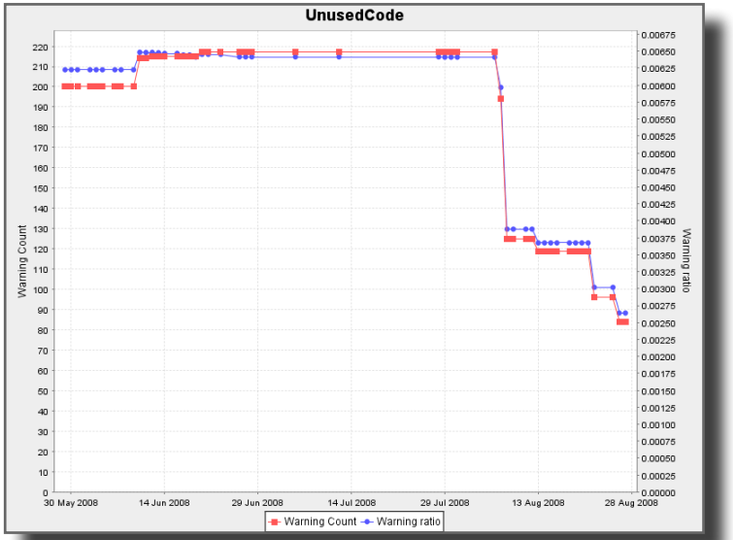
http://www.conqat.org

Qualitäts-Dashboards

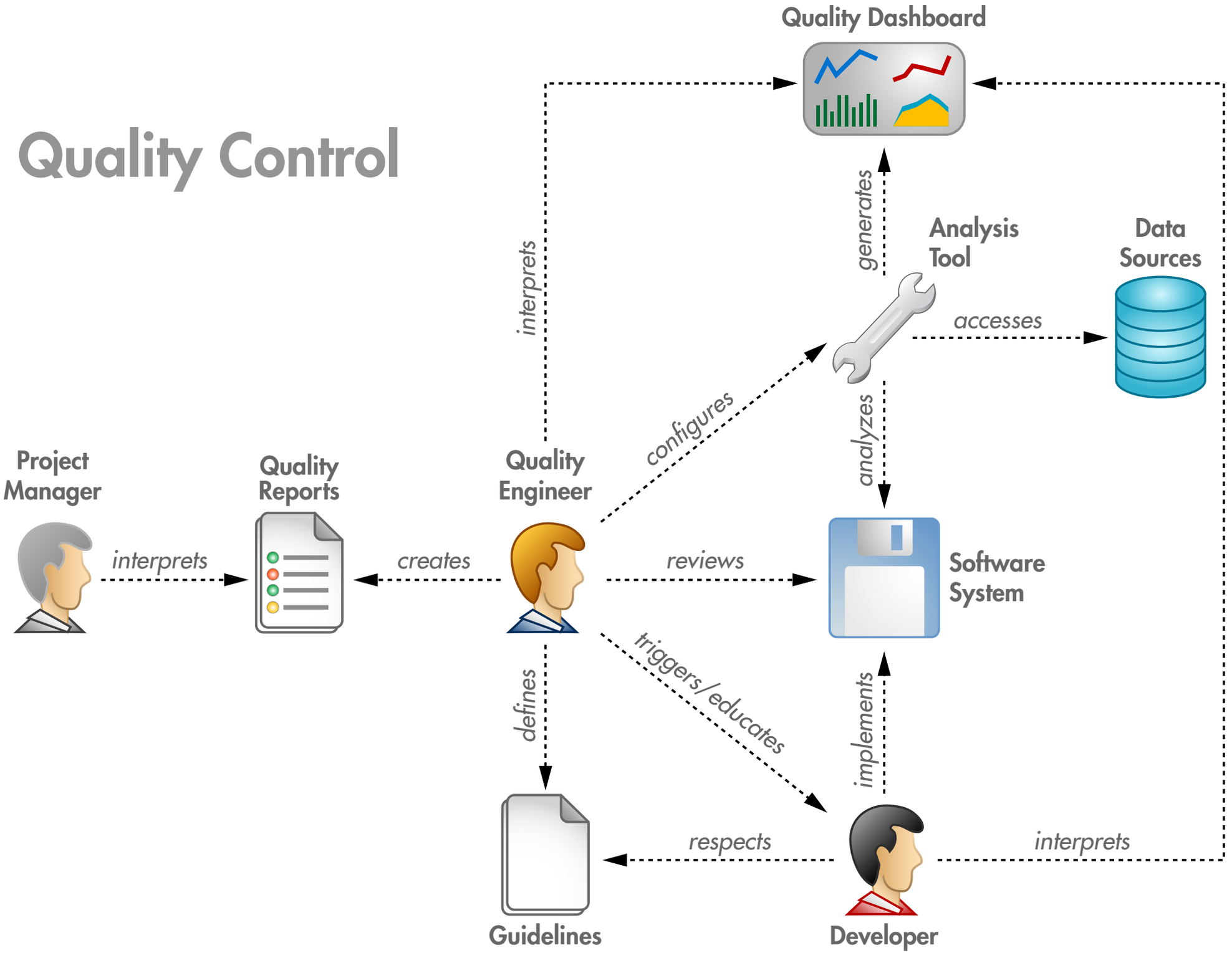








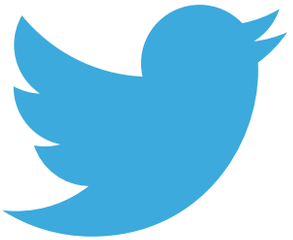
Quality Control



Kontakt

Dr. Florian Deißenböck · deissenboeck@cqse.eu · +49 179 7857188

CQSE GmbH
Lichtenbergstraße 8
85748 Garching bei München



@deissenboeck