

# MATERIALSAMMLUNG - SOFTWAREQUALITÄT

Prof. Dr. Hans-Jürgen Buhl



Wintersemester 2017/2018

Bergische Universität Wuppertal  
Fakultät 4 — Mathematik und Naturwissenschaften  
Fachgruppe Mathematik und Informatik

Praktische Informatik  
PIBUW - WS2017/18  
Oktober 2017  
12. Auflage

Version: 31. Januar 2018

# Inhaltsverzeichnis

Chaos an Hannovers Geldautomaten . . . . .	18
Vorbemerkungen – Softwarequalität heute . . . . .	25
Haftungsausschluß . . . . .	25
IDE Eclipse . . . . .	29
Beispiele für Softwaredisfunktionalitäten . . . . .	39
Deep Impact . . . . .	39
USV-Software legt Server lahm . . . . .	39
Chaos an Hannovers Geldautomaten . . . . .	40
Therac 25 . . . . .	40
Berliner Magnetbahn . . . . .	41
Elektronik-Fehler führt zu Überhitzung bei Volvo-PKW . . . . .	41
The Patriot Missile . . . . .	42
Kontenabrufverfahren startet wegen Softwareproblemen als Provisorium . . . . .	43
Buffer Overflow im Linux-Kernel . . . . .	43
Auch Superhirne können irren - das Risiko Computer . . . . .	44
Explosion der Ariane 5 . . . . .	45
Neueste Risikoinformationen/Softwareprobleme . . . . .	45
<b>1. Softwarequalität</b> . . . . .	<b>47</b>
1.1. Komponententests/Unit-Tests . . . . .	47
1.2. Refactoring . . . . .	48
1.3. Ariane 5 Failure - Full Report . . . . .	48
1.4. Bug plötzlicher unbeabsichtigter Automobilbeschleunigung, ... . . . .	48
1.5. cppcheck CodAn-Tool . . . . .	49
1.6. Exceptions/Trap/Ausnahmebehandlung . . . . .	51
1.7. Spezifikation einer abstrakten Datenkapsel . . . . .	52
1.7.1. Axiomatische Spezifikation . . . . .	52
1.7.2. Beschreibende (denotationale) Spezifikation . . . . .	52
1.7.3. Spezifikation durch Codeverträge . . . . .	55
1.8. iContract (Java) . . . . .	55
1.9. D contracts . . . . .	55
1.10. Prinzipien der ordnungsgemäßen Programmerstellung . . . . .	61
1.11. sanitize-Optionen in gcc 5 . . . . .	61
1.12. Debug-Hilfspakete in OpenSuse . . . . .	62
1.12.1. debuginfo, debugsource in OpenSUSE . . . . .	62
1.12.2. Umstellung auf projektspezifische Codechecks . . . . .	64

1.13. Modularisierung . . . . .	67
1.13.1. Prinzipien der Modularisierung . . . . .	67
1.13.2. Typen der Modularisierung . . . . .	67
1.14. Natürlichsprachige Codeverträge (Kommentare und Verbalisierung) . . . . .	69
1.14.1. Fallstricke umgangssprachlicher Spezifikation . . . . .	69
1.15. Codeverträge . . . . .	71
1.15.1. REQUIRE(), ENSURE(), ID() und invariant() . . . . .	71
1.15.2. Klassifikation der Klassenmethoden gemäß SdV . . . . .	74
1.16. Der Sicherheits-Supergau Spectre/Meltdown . . . . .	76
1.17. Wiederverwendbarkeit in höheren Programmiersprachen . . . . .	79
1.18. Erste einfache Code-Contracts in Eiffel: Vorbedingungen und Klassen-Invarianten . . . . .	85
1.19. Vererbung und Codeverträge (Fortsetzung von 1.15) . . . . .	86
1.20. Integrierte Entwicklungsumgebungen zur Codequalitätssteigerung (am Beispiel: PyDev) . . . . .	88
1.21. Softwarequalitäts-Mängel . . . . .	88
1.22. Ein erstes C++-Projekt mit Umbrello und Doxygen . . . . .	88
1.23. Code-integrierte Dokumentation (mit autom. Dokumentationserzeugung) . . . . .	99
1.24. Ungarische Notation und deren nicht unbedingt codequalitätssteigernde Auswirkungen . . . . .	103
1.25. Eclipse CDT in Zusammenarbeit mit Umbrello/Papyrus, Doxygen, ... . . . .	103
1.26. nana-Installation . . . . .	117
1.27. Vertragsverletzungen zur Laufzeit . . . . .	119
1.28. C++-nana-Contracts in Eclipse . . . . .	122
1.29. Nachbedingungen mit Gleitkommawerten: absolute oder relative Abweichung . . . . .	131
1.29.1. Klasse Wuerfel . . . . .	131
1.29.2. Contract der Klasse Wuerfel . . . . .	132
1.29.3. double_adds.h . . . . .	133
1.29.4. double_math.h . . . . .	134
1.30. Vererbung und Codeverträge . . . . .	135
1.31. beA-Sicherheitsmängel . . . . .	139
1.32. Biometrische Merkmale zum Authentifizieren . . . . .	139
1.33. Vertragsverletzungen zur Laufzeit (Fortsetzung) . . . . .	140
1.33.1. Vertragsverletzungen eines Eiffel-Programms in EiffelStudio: . . . . .	140
1.33.2. Vertragsverletzung bei CLI-Debuglauf: backtrace . . . . .	141
1.33.3. Vertragsverletzungen bei C++-Debug-Lauf in ddd . . . . .	144
1.33.4. Automatischer Start von ddd beim CLI-Debug-Lauf und Vertragsverletzung . . . . .	145
1.34. Nichtänderungs-Verträge für Attribute: Framebedingungen . . . . .	151
1.35. Ultimative Nichtänderungsverträge: const-Methoden . . . . .	153
<b>2. Programming by Contract . . . . .</b>	<b>157</b>
2.1. Spezifikation durch Verträge . . . . .	157
2.2. Alle Verträge der Klasse Klasse vektor . . . . .	161
2.2.1. Klassendeklaration/ Interface . . . . .	161
2.2.1.1. Grundlegende Abfragen . . . . .	164
2.2.1.2. Klassen-Invariante . . . . .	164
2.2.1.3. Konstruktoren . . . . .	164

2.2.1.4.	Destruktor	166
2.2.1.5.	abgeleitete Abfragen/Operationen auf vektor-Exemplaren	166
2.2.1.6.	Modifikatoren	168
2.2.1.7.	Operationen, die vektor-Exemplare erzeugen	168
2.2.1.8.	benutzte klassenexterne Hilfsfunktionen/-Methoden	170
2.2.2.	Quellcode	170
2.3.	Ein Vertrag zur Klasse rationalNumber	171
2.4.	Ein Vertrag mit Queries, Invariants und Actions	177
<b>A.</b>	<b>Design by Contract, by Example, in C++ with nana</b>	<b>179</b>
A.1.	A first Taste of Design by Contract	180
A.2.	Elementary Principles of Design by Contract	185
A.2.1.	First Trial	185
A.2.2.	Redesign	189
A.2.3.	Destruktor, Kopierkonstruktor und Wertzuweisung	194
A.3.	Applying the Six Principles	199
A.3.1.	Design und Contracts	199
A.3.2.	Implementierung und Tests	203
A.3.3.	old-Wert durch Kopie in Form eines geeigneten STL-Container-Exemplars	210
A.3.4.	old-Wert durch den Kopierkonstruktor	214
A.3.5.	Redesign	215
A.4.	Immutable Lists	216
A.5.	Using Immutable Lists	216
A.6.	Subcontracting in Design by Contract in Nana	218
A.6.1.	name.list-Design (Subcontracting)	218
A.6.2.	Implementierung und Tests	222
A.6.3.	Mit Frameregel	225
A.6.4.	Mit Iterator-Methode (Design)	227
A.6.5.	Implementierung der Iterator-Methode	229
A.6.6.	Test des Iterators in display_contents() und main()	230
A.6.7.	Qstl.h: Framebedingungen mit Hilfe eines Iterators	231
A.6.8.	Hilfsoperatoren für die STL	233
A.7.	Neuformulierung: Regeln und Leitlinien für PbC in C++	234



# Abbildungsverzeichnis

0.1. Design by Contract, by Example von Richard Mitchell und Jim McKim .	26
0.2. Bilder von Deep Impact . . . . .	39
0.3. <a href="http://catless.ncl.ac.uk/Risks/22.92.html">http://catless.ncl.ac.uk/Risks/22.92.html</a> . . . . .	45
2.1. Kunden-Lieferanten-Modell . . . . .	157





# Tabellenverzeichnis

- 0.1. Divergence in the Range Gate of a PATRIOT MISSILE . . . . . 42
- 2.1. Pflichten - Nutzen von Kunden und Lieferanten . . . . . 158

Softwarequalität und -korrektheit

2 V Mi 10 - 12 in HS06

*Einordnung:* Bachelor Mathematik, Nebenfach Informatik; Komb. Bachelor of Arts, Informatik; Bachelor IT; Master Wirtschaftsmathematik, Informatik; Wirtschaftswissenschaften: Modul I - Software- und Programmierertechnik; Studienschwerpunkte und Nebenfächer Informatik anderer Studiengänge

*Vorkenntnisse:* Einführung in die Informatik; Programmierkenntnisse in C++; erfolgreiche Teilnahme an xxxMAT500000

*Inhalt:* Nach Übersicht in die desaströse Lage der Qualität vorhandener aktueller Softwareprodukte (Heartbleed Bug, ...) wird anhand professioneller Entwicklungswerkzeuge in qualitätssteigernde Vorgehensweisen eingeführt.

Die Programmiermethodik „Codeverträge oder Programming/Design by Contract“ klärt die Verantwortlichkeit von Diensteanbieter (function) und Dienstenehmer (Aufrufer einer Funktion) durch genaue Vereinbarungen. Mittels des Sprachmittels der Zusicherung werden Voraussetzungen, Diensteeerfüllung und Ausnahmebedingungen zur Laufzeit eines Programms (automatisch) überprüft und führen zu Code besserer Qualität. Daneben werden diverse weitere Hilfsmittel zur Software-Qualitätsverbesserung vorgestellt und diskutiert.

Literatur: wird in der Veranstaltung bekannt gegeben.

(siehe Seite 68 des [Modulhandbuchs](#))

## Aktuell fehlende Softwarequalität

:

Malware legt mehrere Krankenhäuser in Ostengland lahm

The Heartbleed Bug  
How to Prevent the next Heartbleed

Deutsche Bank: Software-Panne  
Erneute IT-Panne

Online-Bank: Schwere Datenpanne bei Comdirect

Hirnforschung: Fehlerhafte MRT-Software schürt Zweifel an Zehntausenden Studien

Traue keinem Scan, den du nicht selbst gefälscht hast

Software-Fehler: Mars-Rover Curiosity im Sicherheitsmodus  
Software-Fehler: Mars-Rover Curiosity zurück aus Sicherheitsmodus

ExoMars: Softwarefehler könnte für Schiaparelli-Absturz verantwortlich sein  
ESA findet Softwarefehler  
Untersuchungsbericht zu Schiaparelli-Absturz

iOS 11: Taschenrechner scheitert an Grundrechenarten

Softwaretesting Stuttgart

Intel schaltet TSX wegen Bug bei Haswell ab  
Intels Haswell kommt 2013 mit neuer Speicherverwaltung  
Intel-Bug: Vorerst kein Transactional Memory  
Install the latest microcode for your processor  
Broadwell reparieren

...

Liste von Programmfehlerbeispielen  
Geschichte der Softwarefehler

...

- 2.1.00: Der Y2k-Direktor der United Nations ruft am 2.1.00 einen weltweiten Alarm vor der Benutzung von Gambio Dialysegeräten aus.
- 3.1.00: Schwere Störungen im Flugverkehr in Chicago und an der gesamten US- Ostküste.
- 3.1.00: Massive Störungen im Flugverkehr von Neuseeland durch Rechnerprobleme
- 3.1.00: Kreditkartensysteme bei amerikanischen Tankstellen ausgefallen.
- 4.1.00: FBI kann die Datenbank für Waffenlizenzen wegen Y2k-Fehler nicht mehr nutzen.
- 4.1.00: Radioaktivitätsüberwachung in Atomwaffenfabrik in Tennessee ausgefallen.
- 4.1.00: Stromausfall in Los Angeles
- 4.1.00: Justizcomputer verlängert Haftstrafen in Italien um 100 Jahre
- 4.1.00: 28 Kernkraftwerke melden Y2k-Probleme
- 4.1.00: Führerscheine in Indiana und New Mexico werden falsch ausgestellt.
- 4.1.00: Kunde einer US-Videothek soll 177.000 DM Strafgebühr bezahlen.
- 4.1.00: In Schweden und Deutschland "Zahlensalat" auf den Konten von Online-Kunden
- 5.1.00: Viertgrößter Autoversicherer der USA hat Y2k-Problem in der Policenverwaltung.
- 5.1.00: Pentagon hat zwei Stunden lang keine Kontrolle über Spionagesatelliten
- 5.1.00: Verwaltungscomputer der Feuerwehr von Washington DC mit Y2k-Fehler
- 6.1.00: Die amerikanische Datenbank für Chemieunfälle ist nicht y2k-fähig.
- 6.1.00: Pentagon stellt 230 falsche Schecks aus.
- 6.1.00: Homebanking-Programm BankUp von Macintosh hat Y2k-Problem
- 6.1.00: 40.000 Händler haben Y2k-Probleme mit Kreditkarten.
- 7.1.00: In Arkansas sind die Verwaltungscomputer von 22 Landkreisen betroffen.
- 7.1.00: Unbekannte Anzahl von 112.000 Cash Cards der US-Post fehlerhaft
- 7.1.00: Chicago-Bank kann in 8 US-Staaten keine Medicare-Überweisungen tätigen.
- 7.1.00: Utah Food Bank in Salt Lake City Total-Crash für einen Tag
- 7.1.00: Die Personalverwaltung der US-Notfallmanagementbehörde ausgefallen.
- 8.1.00: Chevy Chase Bank Window-Versionen von Quicken 99 und 2000 fehlerhaft
- 8.1.00: MCS Spectrum Buchhaltungssoftware nicht y2k-fest
- 8.1.00: First Union Bank bezahlt Arbeiter doppelt: insgesamt 2,3 Mio. US-\$
- 8.1.00: Scheckverkehr in Oregon gestört
- 10.1.00: Quicken Tool der Financial Times wegen Y2k zusammengebrochen
- 10.1.00: Wasserversorgungssystem in amerikanischer Stadt wegen Y2k ausgefallen
- 11.1.00: Datenbanken in 157 staatliche Alkoholläden wegen Y2k gestört
- 11.1.00: Medizinischer Notfallservice der Feuerwehr im Staate Washington ausgefallen
- 12.1.00: Datenbank der Wahlbezirke in Maryland gestört
- 12.1.00: Satellitenausfall des Pentagon ernster als zuerst gemeldet
- 12.1.00: 50.000 Zeugenaussagen in Topeka durch Y2k-Fehler zerstört
- 12.1.00: Y2k-Fehler in Auszahlungssoftware der Deutschen Oper in Berlin
- 13.1.00: LOTUS warnt vor Anwendung der DOMINO-Software wegen Y2k

(aus: Das weltweite Y2k-Überwachungssystem meldet)

## Programmfehler:

### Bugs

## ... und konstruktive Gegenmaßnahmen (Codeverträge) in natürlichsprachiger Art

C++14-Standard, Seite 416: *17.5.1.4 Detailed specifications*, invariants/requires/postconditions/returns oder

C++17-Standard, Seite 452: *20.4.1.4 Detailed specifications*, invariants/requires/postconditions/returns:

```
20.4.1.4 Detailed specifications [structure.specifications]
1 The detailed specifications each contain the following elements:
(1.1) — name and brief description
(1.2) — synopsis (class definition or function declaration, as appropriate)
(1.3) — restrictions on template arguments, if any
(1.4) — description of class invariants
(1.5) — description of function semantics
2 Descriptions of class member functions follow the order (as appropriate):157
(2.1) — constructor(s) and destructor
(2.2) — copying, moving & assignment functions
(2.3) — comparison functions
(2.4) — modifier functions
(2.5) — observer functions
(2.6) — operators and other non-member functions
3 Descriptions of function semantics contain the following elements (as appropriate):158
(3.1) — Requires: the preconditions for calling the function
(3.2) — Effects: the actions performed by the function
(3.3) — Synchronization: the synchronization operations (4.7) applicable to the function
(3.4) — Postconditions: the observable results established by the function
(3.5) — Returns: a description of the value(s) returned by the function
(3.6) — Throws: any exceptions thrown by the function, and the conditions that would cause the exception
(3.7) — Complexity: the time and/or space complexity of the function
(3.8) — Remarks: additional semantic constraints on the function
(3.9) — Error conditions: the error conditions for error codes reported by the function
4 Whenever the Effects: element specifies that the semantics of some function F are Equivalent to some code sequence, then the various elements are interpreted as follows. If F's semantics specifies a Requires: element, then that requirement is logically imposed prior to the equivalent-to semantics. Next, the semantics of the code sequence are determined by the Requires:, Effects:, Synchronization:, Postconditions:, Returns:, Throws:, Complexity:, Remarks:, and Error conditions: specified for the function invocations contained in the code sequence. The value returned from F is specified by F's Returns: element, or if F has no Returns: element, a non-void return from F is specified by the return statements in the code sequence. If F's semantics contains a Throws:, Postconditions:, or Complexity: element, then that supersedes any occurrences of that element in the code sequence.
```

etwa bei der Spezifikation der C++-Standardbibliotheksfunktion `sort()`:

```
28.7.1 Sorting [alg.sort]
28.7.1.1 sort [sort]
template<class RandomAccessIterator>
void sort(RandomAccessIterator first, RandomAccessIterator last);
template<class ExecutionPolicy, class RandomAccessIterator>
void sort(ExecutionPolicy&& exec,
         RandomAccessIterator first, RandomAccessIterator last);
template<class RandomAccessIterator, class Compare>
void sort(RandomAccessIterator first, RandomAccessIterator last,
         Compare comp);
template<class ExecutionPolicy, class RandomAccessIterator, class Compare>
void sort(ExecutionPolicy&& exec,
         RandomAccessIterator first, RandomAccessIterator last,
         Compare comp);
1 Requires: RandomAccessIterator shall satisfy the requirements of ValueSwappable (20.5.3.2). The
   type of *first shall satisfy the requirements of MoveConstructible (Table 23) and of MoveAssignable
   (Table 25).
2 Effects: Sorts the elements in the range [first, last).
3 Complexity:  $\mathcal{O}(N \log N)$  comparisons, where  $N = \text{last} - \text{first}$ .
```

Besser (vollständiger) sind die Spezifikationen der C++-Standardbibliothek (früher häufig STL genannt) in [Standard Template Library Programmer's Guide](#):

## sort

### Algorithms

Category: algorithms

### Function

Component type: function

#### Prototype

sort is an overloaded name; there are actually two sort functions.

```
template <class RandomAccessIterator>
void sort(RandomAccessIterator first, RandomAccessIterator last);
```

```
template <class RandomAccessIterator, class StrictWeakOrdering>
void sort(RandomAccessIterator first, RandomAccessIterator last,
          StrictWeakOrdering comp);
```

#### Description

sort sorts the elements in `[first, last)` into ascending order, meaning that if `i` and `j` are any two valid iterators in `[first, last)` such that `i` precedes `j`, then `*j` is not less than `*i`. Note: sort is not guaranteed to be stable. That is, suppose that `*i` and `*j` are equivalent: neither one is less than the other. It is not guaranteed that the relative order of these two elements will be preserved by sort. [1]

The two versions of sort differ in how they define whether one element is less than another. The first version compares objects using operator`<`, and the second compares objects using a [function object](#) comp.

#### Definition

Defined in the standard header [algorithm](#), and in the nonstandard backward-compatibility header [algo.h](#).

#### Requirements on types

For the first version, the one that takes two arguments:

- RandomAccessIterator is a model of [Random Access Iterator](#).
- RandomAccessIterator is mutable.
- RandomAccessIterator's value type is [LessThan Comparable](#).
- The ordering relation on RandomAccessIterator's value type is a *strict weak ordering*, as defined in the [LessThan Comparable](#) requirements.

For the second version, the one that takes three arguments:

- RandomAccessIterator is a model of [Random Access Iterator](#).
- RandomAccessIterator is mutable.
- StrictWeakOrdering is a model of [Strict Weak Ordering](#).
- RandomAccessIterator's value type is convertible to StrictWeakOrdering's argument type.

#### Preconditions

- `[first, last)` is a valid range.

## C++20 Codeverträge

Um solche „formalen“ Spezifikationen auch maschinenüberprüfbar zu machen, wurde für C++20 in [Contracts programming for C++20 \(J. Daniel Garcia\)](#) eine C++-Sprachergänzung für Codeverträge mit Hilfe der Attribute `expects`, `ensures` und `assert` vorgeschlagen:

```
template <typename Iter >
requires RandomAccessIterator<Iter >()
Iter bsearch(Iter p, Iter q, std::iterator_traits<Iter >::value_type w){
    [[expects default: p <= q]]
    [[expects audit: is_sorted(p, q)]]
    [[ensures result: !(result != q) || (*result == w
        && !(result != p) || *(result - 1) != w) ]]
    [[ensures axiom result:
        !(std::none_of(p, q, [] (auto v){return v == w;}) || result == q ]]
    {
        // ...
    }
```

[Extensible Templates: Via Inheritance or Traits?](#)

## C++ attributes

### Concepts Lite: Constraining Templates with Predicates

#### C++17 attributes:

Neue Systemattribute als Code-Metainformation:

Attribute	Description
<code>[[fallthrough]](C++17)</code>	Appears in a <code>switch</code> statement on a line of its own (technically as an attribute of a <code>null statement</code> ), immediately before a case label. Indicates that the fall through from the previous case label is intentional and should not be diagnosed by a compiler that warns on fallthrough.
<code>[[nodiscard]](C++17)</code>	Appears in a function declaration, enumeration declaration, or class declaration. If a function declared <code>nodiscard</code> or a function returning an enumeration or class declared <code>nodiscard</code> by value is called from a <code>discarded-value expression</code> other than a cast to void, the compiler is encouraged to issue a warning.
<code>[[maybe_unused]](C++17)</code>	Appears in the declaration of a class, a typedef, a variable, a nonstatic data member, a function, an enumeration, or an enumerator. If the compiler issues warnings on unused entities, that warning is suppressed for any entity declared <code>maybe_unused</code> .

Beispiel zum Attribut `maybe_unused`:

```
namespace {
    [[maybe_unused]] std::string createWindowsConfigFilePath(
        const std::string &relativePath);
    // TODO: Reuse this on BSD, MAC ...
    [[maybe_unused]] std::string createLinuxConfigFilePath(
        const std::string &relativePath);
}

std::string createConfigFilePath(const std::string &relativePath) {
#ifdef OS == "WINDOWS"
    return createWindowsConfigFilePath(relativePath);
#elif OS == "LINUX"
    return createLinuxConfigFilePath(relativePath);
#else
#error "OS is not yet supported"
#endif
}
//...
```

(aus [C++ attribute examples](#))

Besser mit der neuen C++17 bedingten Compilation:

```
/* workaround until operator== is constexpr
 * http://open-std.org/JTC1/SC22/WG21/docs/papers/2017/p0533r0.pdf
 */
constexpr bool equal(char const* lhs, char const* rhs)
{
    while (*lhs || *rhs)
        if (*lhs++ != *rhs++)
            return false;
    return true;
}
```

```
// aus: https://stackoverflow.com/questions/27490858/how-can-you-compare-two-character-strings-statically-at-compile-time//
```

```
template <class...>
struct False : std::bool_constant<false> { };
//template <
//struct False<float> : std::bool_constant<>true> { };
template <class T = int>
void ErrorAbort(const char* Msg){
    static_assert(False<T>{}, Msg);
}

std::string createConfigFilePath(const std::string &relativePath) {
    if constexpr(equal(OS, "WINDOWS")){
        return createWindowsConfigFilePath(relativePath);
    } else if constexpr(equal(OS, "LINUX")){
        return createLinuxConfigFilePath(relativePath);
    } else {
        ErrorAbort<>("OS not yet supported");
        // static_assert(false, "OS is not yet supported"); doesn't
        work
    }
}
}
```

where to specify maybe\_unused

`#error`

`static_assert`

getenv in C++

(„C++ attributes“ entsprechen „Java Annotationen“)

noreturn-Attribut:

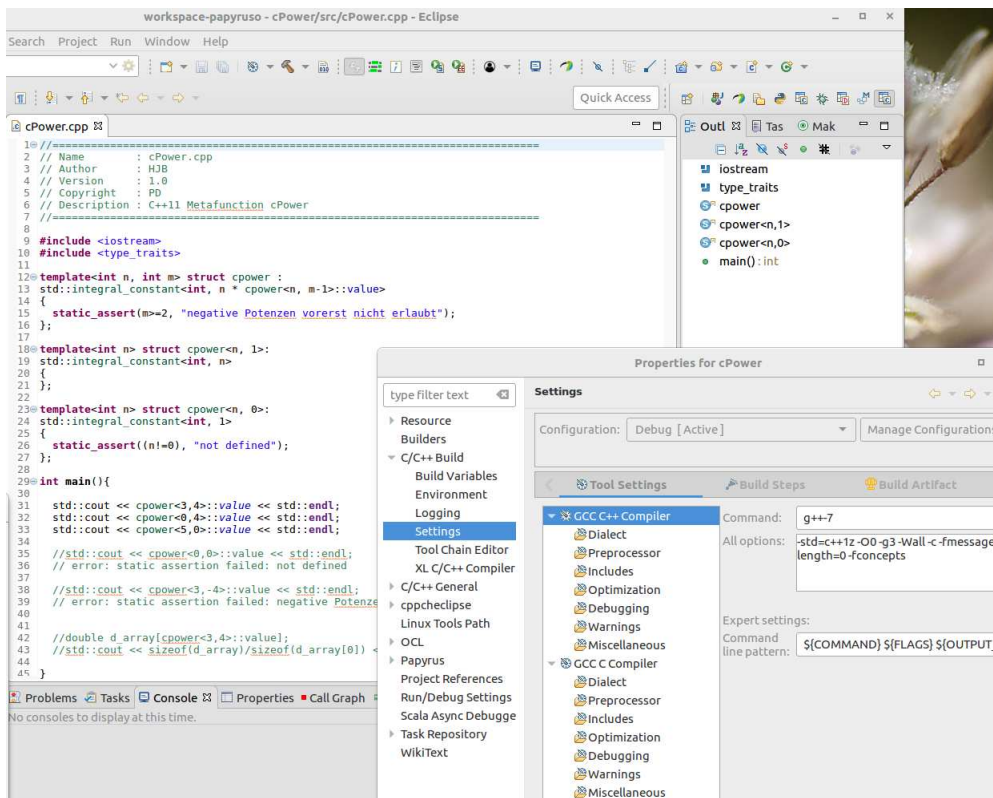
```
// 21.5, start and termination
[[noreturn]] void abort() noexcept;
int atexit(c_atexit_handler* func) noexcept;
int atexit(atexit_handler* func) noexcept;
int at_quick_exit(c_atexit_handler* func) noexcept;
int at_quick_exit(atexit_handler* func) noexcept;
[[noreturn]] void exit(int status);
[[noreturn]] void _Exit(int status) noexcept;
[[noreturn]] void quick_exit(int status) noexcept;
```



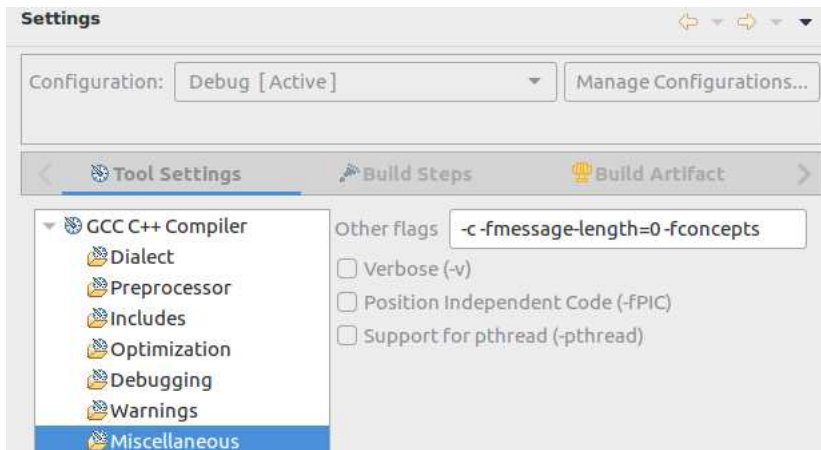
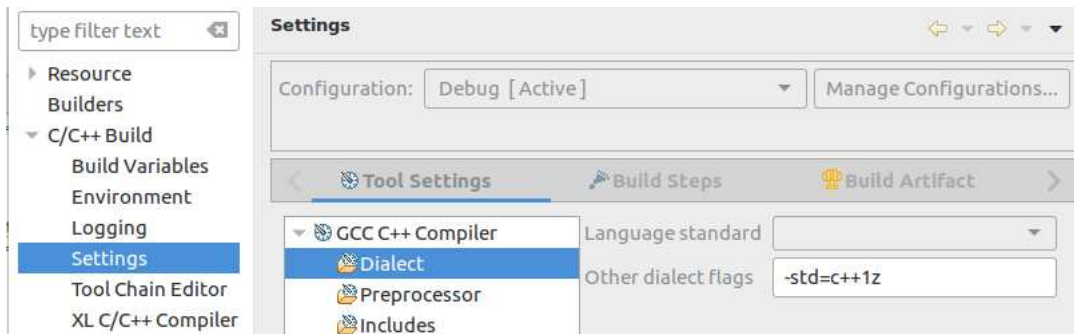
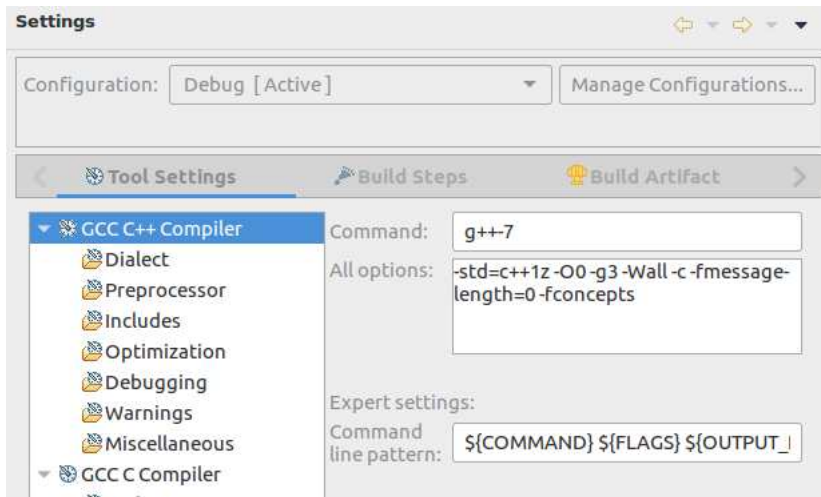
Wir benutzen die IDE **Eclipse**, auf den Ausbildungslaborrechnern der Fachgruppe Mathematik und Informatik als `eclipse-papyruso1` aufruf- und benutzbar:



### Hinweise zu den Compiler-Optionen für C++17:



`g++-7, -std=c++1z und -fconcepts`



Hinweise zur Installation von Eclipse auf Ihrem eigenen Linux-Rechner finden Sie im Abschnitt **IDE für C++-Programmierung: Eclipse ... Modeling mit CDT, UML, OCL, Linux Tools, ...** (Seite 30).

```

1 //=====
2 // Name      : New2C++.cpp
3 // Author    : HJB
4 // Version   :
5 // Copyright : PD
6 // Description : constexpr if- else
7 //=====
8
9 #include <iostream>
10 #include <cstdlib>
11 //using namespace std;
12
13 constexpr bool equal( char const* lhs, char const* rhs )
14 {
15     while (*lhs || *rhs)
16         if (*lhs++ != *rhs++)
17             return false;
18     return true;
19 }
20
21 template <class...>
22 struct False : std::bool_constant<false> { };
23 //template <>
24 //struct False<float> : std::bool_constant<>true> { };
25
26 template <class T = int>
27 void others(){
28     static_assert(False<T>{}, "OS not jet supported");
29 }
30 /* https://stackoverflow.com/questions/44059557/whats-the-right-way-to-call-static-asse
31
32 int main(int argc, const char* argv[]) {
33     std::cout << "Testrahmenprogramm:" << std::endl;
34
35     constexpr char OS[] = "LINUX";
36
37     if constexpr(equal(OS, "WINDOWS")){
38         std::cout << "Windows" << std::endl;
39     } else if constexpr(equal(OS, "LINUX")){
40         std::cout << "Linux" << std::endl;
41     } else {
42         others<>();
43         //static_assert(false, "OS not jet supported"); not allowed
44     };

```

Problems Tasks Console Properties Call Graph

CDT Build Console [New2C++]

Invoking: GCC C++ Compiler  
g++-7 -std=c++1z -O0 -g3 -Wall -c -fmessage-length=0 -Wno-attributes -Wno-attributes -fconcepts  
Finished building: ../src/New2C++.cpp

## Statische Code-Analyse (lint, cdt codan, cppcheck, ...):

Lint (software)

CDT/designs/StaticAnalysis

Cppcheck und Cppcheck with Eclipse

The screenshot shows the Eclipse IDE interface. The context menu for the 'cppcheck' option is open, displaying the following items:

- Run cppcheck (Umschalt+Strg+C)
- Clear cppcheck Markers (Umschalt+Strg+M)

The error list below shows the following messages:

Message	Resource	Path
Possible assignment in condition 'a = b'	vorbem-codan.1	/vorbe
Statement has no effect 'b+1'	vorbem-codan.1	/vorbe
Un- or ill-initialized variable found	vorbem-codan.1	/vorbe
Un- or ill-initialized variable found	vorbem-codan.1	/vorbe

```

1 //=====  

2 // Name      : vorbem-codan.cpp  

3 // Author    : HJB  

4 // Version   :  

5 // Copyright : PD  

6 // Description : Hello World in C++, Ansi-style  

7 //=====  

8  

9 #include <iostream>  

10  

11 int main() {  

12     int a;  

13     int b;  

14     if (a = b) return 0;  

15     b+1;  

16  

17     std::cout << "Testrahmenprogramm codan:" << std::endl;  

18     return 0;  

19 }  

20

```

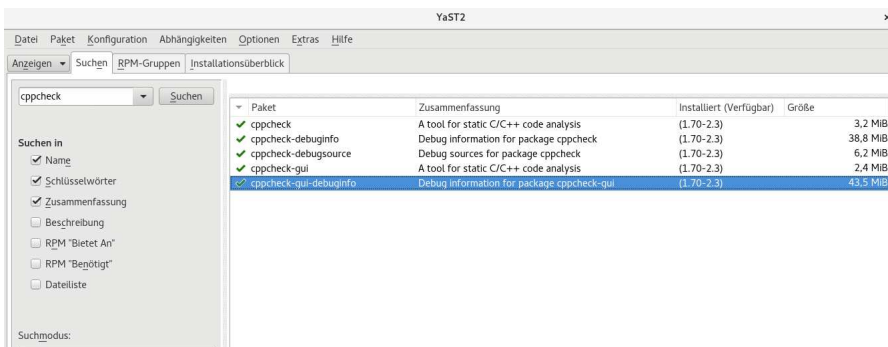
Problems 7 warnings, 0 others

Description	Resource	Path	Location	Type
Warnings (7 items)				
'b' is used uninitialized in this function [-Wuninitialized]	vorbem-codan	/vorbem-codan/src	line 14	C/C++ Problem
Possible assignment in condition 'a = b'	vorbem-codan	/vorbem-codan/src	line 14	Semantic Error
Statement has no effect 'b+1'	vorbem-codan	/vorbem-codan/src	line 15	Code Analysis Problem
statement has no effect [-Wunused-value]	vorbem-codan	/vorbem-codan/src	line 15	C/C++ Problem
suggest parentheses around assignment used as truth value [-Wparentheses]	vorbem-codan	/vorbem-codan/src	line 14	C/C++ Problem
Un- or ill-initialized variable found	vorbem-codan	/vorbem-codan/src	line 12	Elevation Suggestion
Un- or ill-initialized variable found	vorbem-codan	/vorbem-codan/src	line 13	Elevation Suggestion

Problems 1 error, 7 warnings, 0 others

Description	Resource	Path	Location	Type
Errors (1 item)				
[cppcheck error] Uninitialized variable: b	vorbem-codan	/vorbem-codan/src	line 14	cppcheck Problem
Warnings (7 items)				
'b' is used uninitialized in this function [-Wuninitialized]	vorbem-codan	/vorbem-codan/src	line 14	C/C++ Problem
Possible assignment in condition 'a = b'	vorbem-codan	/vorbem-codan/src	line 14	Semantic Error
Statement has no effect 'b+1'	vorbem-codan	/vorbem-codan/src	line 15	Code Analysis Problem
statement has no effect [-Wunused-value]	vorbem-codan	/vorbem-codan/src	line 15	C/C++ Problem
suggest parentheses around assignment used as truth value [-Wparentheses]	vorbem-codan	/vorbem-codan/src	line 14	C/C++ Problem
Un- or ill-initialized variable found	vorbem-codan	/vorbem-codan/src	line 12	Elevation Suggestion
Un- or ill-initialized variable found	vorbem-codan	/vorbem-codan/src	line 13	Elevation Suggestion

Das Programm cppcheck muß zusätzlich zu eclipse zum Beispiel mit Yast installiert sein:



Auch g++ besitzt in letzter Zeit verbesserte Hinweise auf „augenscheinliche“ Tippfehler:

- GCC 6's C and C++ frontends were able to offer suggestions for misspelled field names:

```
spellcheck-fields.cc:52:13: error: 'struct s' has no member named 'colour'; did you mean 'color'?  
return ptr->colour;  
                ^~~~~~  
                color
```

GCC 7 greatly expands the scope of these suggestions. Firstly, it adds fix-it hints to such suggestions:

```
spellcheck-fields.cc:52:13: error: 'struct s' has no member named 'colour'; did you mean 'color'?  
return ptr->colour;  
                ^~~~~~  
                color
```

The suggestions now cover many other things, such as misspelled function names:

```
spellcheck-identifiers.c:11:3: warning: implicit declaration of function 'gtk_widget_showall'; did you mean 'gtk_widget_show_all'? [-Wimplicit-function-declaration]  
gtk_widget_showall(w);  
^~~~~~~~~~~~~~~~~~  
gtk_widget_show_all
```

misspelled macro names and enum values:

```
spellcheck-identifiers.cc:85:11: error: 'MAX_ITEM' undeclared here (not in a function); did you mean 'MAX_ITEMS'?  
int array[MAX_ITEM];  
         ^~~~~~  
         MAX_ITEMS
```

misspelled type names:

```
spellcheck-typenames.c:7:14: error: unknown type name 'singed'; did you mean 'signed'?  
void test (singed char e);  
          ^~~~~~  
          signed
```

and, in the C frontend, named initializers:

```
test.c:7:20: error: 'struct s' has no member named 'colour'; did you mean 'color'?  
struct s test = { .colour = 3 };  
                  ^~~~~~  
                  color
```

## Weitere mögliche Ursachen von Software-Katastrophen:

```
1 //=====
2 // Name      : arrayNeu.cpp
3 // Author    : HJB
4 // Version   : 1.0
5 // Copyright : PD
6 // Description : array<.,.>-C++11-Test, Ansi-style
7 //=====
8
9 #include <iostream>
10 #include <array>
11 /* using namespace std; */
12
13 void f1()
14 {
15     std::array<double, 3> vec2 {1.0, 1.1, 1.2};
16     std::cout << vec2[3] << std::endl;           // 3, 10
17 }
18
19 int main() {
20     std::cout << "Test des C++11 arrays:" << std::endl;
21
22     std::array<double, 3> vec1 {1.0, 1.1, 1.2};
23     std::cout << vec1[2] << std::endl;
24     f1();
25
26     return 0;
27 }
28 |
```

Problems Tasks Console

<terminated> arrayNeu2 [C/C++ Application] /home/buhl/workspace-papyrus-neu/arrayNeu2/Debug/arrayNeu2 (08.09.14 10:29)  
Test des C++11 arrays:  
1.2  
6.90307e-310

C++ Accesses an Array out of bounds gives no error, why?

GCC 5.0: `-fsanitize=bounds`: enable instrumentation of array bounds and detect out-of-bounds accesses;

GCC 5.1 mit Offloading und Cilk-Plus-Support erschienen

Using gcc's 4.8.0 Address Sanitizer: leider für Indexüberprüfung recht ungeeignet  
`address-sanitizer` – how it works

What is the C++ compiler required to do with ill-formed programs according to the Standard?  
Undefined behavior

## C++14: Abschnitt 1.3.24 undefined behavior

### Index checking [edit]

---

Index checking means that, in all [expressions](#) indexing an array, the index value is checked against the bounds of the array (which were established when the array was defined), and if the index is out-of-bounds, further execution is suspended via some sort of error. Because using a number outside of the upper range in an array may cause the program to crash, or may introduce security vulnerabilities (see [buffer overflow](#)), index checking is a part of many [high-level languages](#).

Pascal, Fortran, Java have index checking ability. The [VAX](#) computer has an INDEX assembly instruction for array index checking which takes six operands, all of which can use any VAX addressing mode. The B6500 and similar [Burroughs](#) computers performed bound checking via hardware, irrespective of which computer language had been compiled to produce the machine code. A limited number of later [CPUs](#) have specialised instructions for checking bounds, e.g. The CHK2 instruction on the [Motorola 68000](#) series.

Many [programming languages](#), such as [C](#), never perform automatic bounds checking to raise speed. However, this leaves many [off-by-one errors](#) and [buffer overflows](#) uncaught. Many programmers believe these languages sacrifice too much for rapid execution.<sup>[*who?*]</sup> In his 1980 [Turing Award](#) lecture, [C. A. R. Hoare](#) described his experience in the design of [ALGOL 60](#), a language that included bounds checking, saying:

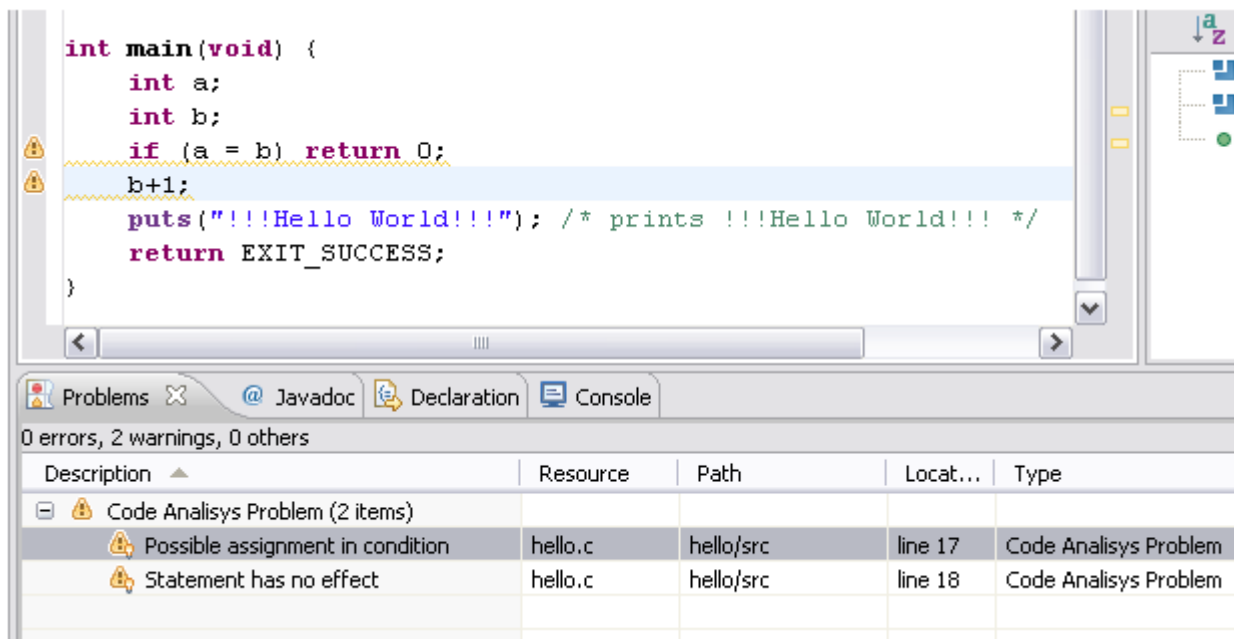
A consequence of this principle is that every occurrence of every subscript of every subscripted variable was on every occasion checked at run time against both the upper and the lower declared bounds of the array. Many years later we asked our customers whether they wished us to provide an option to switch off these checks in the interest of efficiency on production runs. Unanimously, they urged us not to—they already knew how frequently subscript errors occur on production runs where failure to detect them could be disastrous. I note with fear and horror that even in 1980, language designers and users have not learned this lesson. In any respectable branch of engineering, failure to observe such elementary precautions would have long been against the law.

Mainstream languages that enforce run time checking include [Ada](#), [C#](#), [Haskell](#), [Java](#), [JavaScript](#), [Lisp](#), [PHP](#), [Python](#), [Ruby](#), and [Visual Basic](#). The [D](#) and [OCaml](#) languages have run time bounds checking that is enabled or disabled with a compiler switch. [C#](#) also supports *unsafe regions*: sections of code that (among other things) temporarily suspend bounds checking to raise efficiency. These are useful for speeding up small time-critical bottlenecks without sacrificing the safety of a whole program.

(aus: [http://en.wikipedia.org/wiki/Bounds\\_checking#Index\\_checking](http://en.wikipedia.org/wiki/Bounds_checking#Index_checking) )

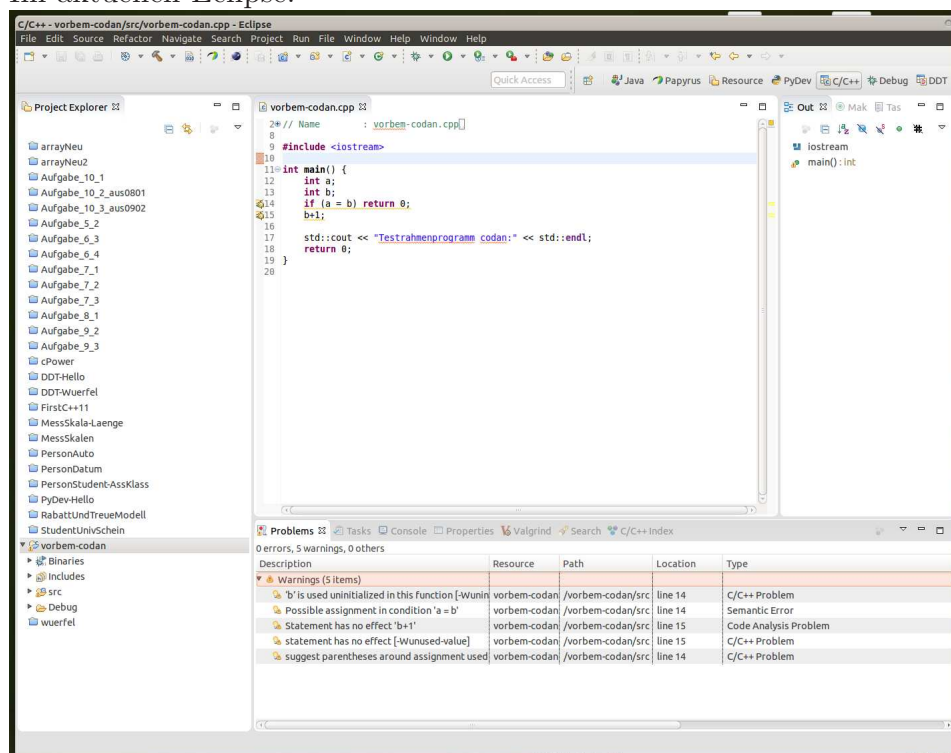


Hilfsmittel zur Vermeidung von Codefehlern:



(aus: CDT/designs/StaticAnalysis)

Im aktuellen Eclipse:



Siehe auch: Statische Code-Analyse, ...

## Chaos an Hannovers Geldautomaten (05.10.2003 13:00 Uhr)

Computerprobleme haben am Samstag alle 240 Geldautomaten der Sparkasse in der Stadt und Region Hannover lahm gelegt. Die Fusion der Stadt- und Kreissparkasse sollte am Wochenende auch technisch umgesetzt werden, sagte der Sprecher des Geldinstituts, Stefan Becker. Beim Hochfahren eines Server habe sich ein Fehler eingeschlichen, so dass die Geldautomaten nicht mehr funktionierten. Die Sparkasse öffnete stattdessen fünf Filialen, damit Kunden etwa in Einkaufszonen Bargeld abheben können.

(aus: <http://www.heise.de/newsticker/meldung/40834>)

## Software-Softwarequalitätsmerkmale (nach Balzert):

### A. Produktorientiert:

1. funktionale **Korrektheit** (benötigt Spezifikation)
2. funktionale Vollständigkeit
3. Robustheit gegenüber dem Benutzer
4. Benutzerfreundlichkeit
5. Effizienz in Laufzeit
6. Effizienz in Arbeitsspeicherbedarf
7. Effizienz in Plattenspeicherbedarf
8. Integrität (gegenüber unauthorisierten Änderungen)
9. Kompatibilität, Integrationsfähigkeit, Standards
10. **Hochverfügbarkeit**

### B. Projekt- bzw. teamarbeitsorientiert:

1. Verständlichkeit (des GUI, der Dokumentation, ...)
2. Überprüfbarkeit
3. Wartbarkeit
4. Änderbarkeit, Erweiterbarkeit
5. Portierbarkeit
6. Produktivität
7. Wiederverwendbarkeit

Siehe auch:

**Softwaretechnik I (A. Zeller),**  
**ISO 9126,**  
**FURPS**

## Relevanzmatrizen zur Beurteilung im Projekt wichtiger Merkmale

Jedem Qualitätsmerkmal kann man in Design- und Implementierungsphase eine mehr oder wenig wichtige Rolle zuordnen. Diese Bewertungszuordnung nennen wir Relevanzmatrix.

Sie kann auch bei der Beurteilung von Softwarekatastrophen eingesetzt werden, etwa:

```
+++ besonders wichtig
++ wichtig
+ beachtenswert
. neutral
- missachtet
-- stark missachtet
--- katastrophal missachtet
```

Ein Beispiel:

### Untersuchungsbericht zu Schiaparelli-Absturz

Korrektheit	--
Vollständigkeit	--
Robustheit	---
Benutzerfreundlichkeit	
Laufzeit-Effizienz	
Hauptspeicher-Effizienz	
Massenspeicher-Effizienz	
Integrität	
Kompatibilität, Integrationsfähigkeit, Standards	
Hochverfügbarkeit	
-----	
Verständlichkeit	---
Überprüfbarkeit	
Wartbarkeit	
Änderbarkeit/Erweiterbarkeit	
Portierbarkeit	
Produktivität	
Wiederverwendbarkeit	---

## Memory safety

siehe: [http://en.m.wikipedia.org/wiki/Memory\\_safety](http://en.m.wikipedia.org/wiki/Memory_safety)

- Buffer overflow - Out-of bound writes can corrupt the content of adjacent objects, or internal data like bookkeeping information for the heap or return addresses.
- Dynamic memory errors - Incorrect management of dynamic memory and pointers:
  - Dangling pointer - A pointer storing the address of an object that has been deleted.
  - Double frees - Repeated call to free though the object has been already freed can cause freelist-based allocators to fail.
  - Invalid free - Passing an invalid address to free can corrupt the heap. Or sometimes will lead to an undefined behavior.
  - Null pointer accesses will cause an exception or program termination in most environments, but can cause corruption in operating system kernels or systems without memory protection, or when use of the null pointer involves a large or negative offset.
- Uninitialized variables - A variable that has not been assigned a value is used. It may contain an undesired or, in some languages, a corrupt value:
  - Wild pointers arise when a pointer is used prior to initialization to some known state. They show the same erratic behaviour as dangling pointers, though they are less likely to stay undetected.
- Out of memory errors:
  - Stack overflow - Occurs when a program runs out of stack space, typically because of too deep recursion.
  - Allocation failures - The program tries to use more memory than the amount available. In some languages, this condition must be checked for manually after each allocation.

```
new  
std::nothrow  
try ... catch  
std::exception
```

## Move-Semantik:

(Austausch von Referenzen statt Verschieben von Datenblöcken, (längere) Datenblöcke so lange wie möglich wieder-/weiterverwenden, Datenfluß-zentriertes Konzipieren von Algorithmen)

```
class SocketManager
{
public:
    SocketManager() = default;
    SocketManager(IPAddress addr, Port p);
    ~SocketManager();
    SocketManager& operator=(SocketManager rhs);
    SocketManager(const SocketManager& src);

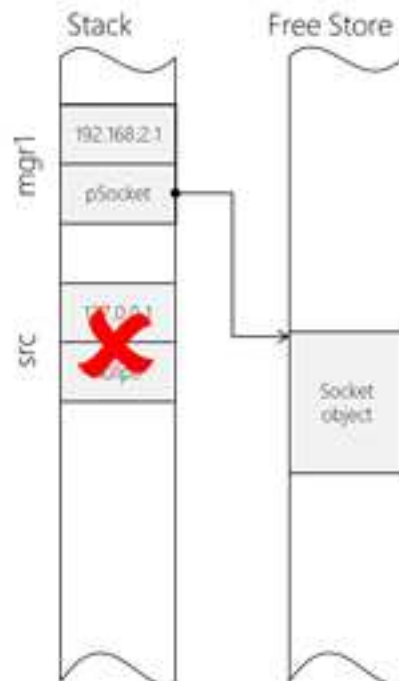
    SocketManager(SocketManager&& src);
private:
    IPAddress ip = 0x7F000001;
    Socket* pSocket = nullptr;

    void swap(SocketManager& rhs);
};

SocketManager::SocketManager(SocketManager&& src)
{
    ip = src.ip;
    pSocket = src.pSocket;

    src.ip = 0x7F000001;
    src.pSocket = nullptr;
}
```

*Overload by  
r-value  
reference*



(aus: <https://blog.feabhas.com/2015/01/the-rule-of-the-big-four-and-a-half-move-semantics-and-resource-management/>)

## Illustration of move semantics: Move Semantics and Resource Management

Introducing the Move Constructor and the Move Assignment

C++11 rvalues and move semantics confusion (return statement)

## RVO:

Return Value Optimization, RVO vs. std::move

copy elision

copy elision (Wikipedia)

## Modern C++ Style:

Scott Meyers: Effective Modern C++, 42 Specific Ways to Improve Your Use of C++11 and C++14

## Fehlende Benutzung aggressiver Laufzeit-Zusicherungen zur Qualitätsverbesserung von Software:

Heartbleed and Formal Methods

How to Prevent the next Heartbleed

Acceptance of Formal Methods: Lessons from Hardware Design — the FDIV bug

20 Jahre FDIV-Bug: Ein Prozessor-Rechenfehler macht Geschichte

20 Jahre FDIV-Bug: Die Hintergründe

Intel Skylake bug causes PCs to freeze during complex workloads

Oder mittels bei jedem Produktions- und Testlauf dauernd überprüfter Codeverträge und sofortiger Meldung bei Vertragsverletzung:

The screenshot shows the Eclipse IDE interface. The main editor displays the source code for `Wuerfel-nana.cpp`. The code includes a `main` function that creates a `Wuerfel` object `w` with the value `-3.5`. The program prints the surface area of the cube and then fails with an assertion error: `I{mySeite >= 0} failed; dumping core`. The Debug Console shows the stack trace, indicating the error occurred in `Wuerfel::Wuerfel()` at line 17. The Variables window shows the state of the `w` object, with `Seite` set to `-3.5`. The Outline window shows the project structure, including `iostream`, `cstdlib`, and `string`.

```
21 int main() {
22     cout << "Hier meldet sich das Testrahmenprogramm zur Klasse Wuerfel" << endl;
23
24     Wuerfel w(-3.5);
25     // assert( w.toString() == "1;6;1;1.73205" );
26
27     std::cout << "Oberflaeche: " << w.Oberflaeche() << std::endl;
28
29     std::cout << "w.toString()= " << w.toString() << std::endl;
30 }
```

Name	Type	Value
w	Wuerfel	{...}
Seite	double	-3.5

```
Wuerfel-nana [C/C++ Application] Wuerfel-nana
Hier meldet sich das Testrahmenprogramm zur Klasse Wuerfel
../src/Wuerfel.cpp:17: I{mySeite >= 0} failed; dumping core
```

Debug - Wuerfel-nana/src/Wuerfel.cpp - Eclipse

File Edit Source Refactor Navigate Search Pydev Project Run File Window Help Window Help

Quick Access C/C++ Debug Resource

Debug

Thread [1] 6689 [core: 0] (Suspended : Signal : SIGABRT)

- \_\_GI\_raise() at raise.c:56 0x7fff7249065
- \_\_GI\_abort() at abort.c:90 0x7fff724a4e8
- \_l\_default\_handler() at l.c:51 0x401b40
- Wuerfel:Wuerfel() at Wuerfel.cpp:17 0x401541
- main() at Wuerfel-nana.cpp:24 0x401145

(x) Variables Breakpoints Registers Modules

Name	Type	Value
this	Wuerfel * const	0x7fffffd350
mySeite	double	-3.5

\*Wuerfel.cpp Wuerfel-nana.cpp std:operator<< <std::char\_traits<< l.c

```

11 #include "wuerfel.h"
12 #include "double_adds.h"
13 #include "double_math.h"
14
15 Wuerfel::Wuerfel(double mySeite): Seite(mySeite)
16 {
17     REQUIRE(mySeite >= 0);
18     ENSURE(invariant());
19 }
20 END

```

Outline

- sstream
- limits
- cmath

Console Tasks Problems Executables Memory

Wuerfel-nana [C/C++ Application] Wuerfel-nana  
 Hier meldet sich das Testrahmenprogramm zur Klasse Wuerfel  
 ../src/Wuerfel.cpp:17: I(mySeite >= 0) failed; dumping core





# Vorbemerkungen – Softwarequalität heute

Produkthaftung auch für Software?

## Haftungsausschluß

Die Überlassung dieser Baupläne erfolgt ohne Gewähr. Der Planer gibt keine Garantie, Gewährleistung oder Zusicherung, daß diese Pläne für einen bestimmten Zweck geeignet sind, daß sie richtig sind oder daß ein Gebäude, das nach diesen Plänen gebaut wird, den Ansprüchen des jeweiligen Erwerbers genügt. Der Planer erklärt sich bereit, Ersatzkopien derjenigen Teile der Pläne zu liefern, die zum Zeitpunkt des Kaufs unleserlich sind. Darüber hinaus wird keinerlei Haftung übernommen. Der Erwerber dieser Pläne sollte beachten, daß in den entscheidenden Phasen des Baus und nach der Fertigstellung geeignete Tests durchzuführen sind und daß die üblichen Vorsichtsmaßnahmen zum Schutz des Lebens der Bauarbeiter zu treffen sind.

(Zitat: Robert L. Baber: *Softwarereflexionen*, Springer-Verlag, Seiten 1...10)

und in der Praxis:

...

## 2. Haftung

Wir werden immer bemüht sein, ihnen einwandfreie Software zu liefern. Wir können aber keine Gewähr dafür übernehmen, daß die Software unterbrechungs- und fehlerfrei läuft und daß die in der Software enthaltenen Funktionen in allen von Ihnen gewählten Kombinationen ausführbar sind. Für die Erreichung eines bestimmten Verwendungszweckes können wir ebenfalls keine Gewähr übernehmen. Die Haftung für unmittelbare Schäden, mittelbare Schäden, Folgeschäden und Drittschäden ist, soweit gesetzlich zulässig, ausgeschlossen. Die Haftung bei grober Fahrlässigkeit und Vorsatz bleibt hiervon unberührt, in jedem Fall ist jedoch die Haftung beschränkt auf den Kaufpreis.

(AGB, Punkt 9)

*Design by Contract* (Spezifikation durch Verträge = SdV)

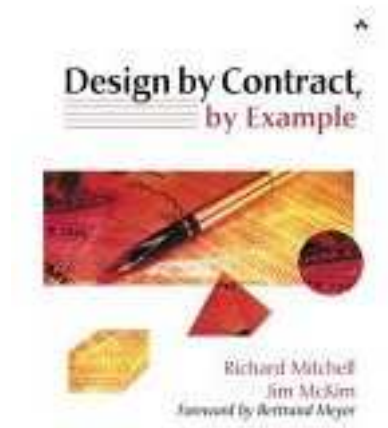


Abbildung 0.1.: *Design by Contract, by Example* von Richard Mitchell und Jim McKim

Vergleiche auch **SQS** und **SQA**.

In den ersten Kapiteln der Vorlesung wird Grundwissen zur Softwarequalität und -qualitätssicherung wiederholt.

Die dann behandelte Methodik DbC wurde zuerst in der Programmiersprache **Eiffel** thematisiert, ist jedoch heute in (fast) allen neuen Programmiersprachen nutzbar.

[http://en.wikipedia.org/wiki/Eiffel\\_\(programming\\_language\)](http://en.wikipedia.org/wiki/Eiffel_(programming_language))

[http://en.wikipedia.org/wiki/Design\\_by\\_contract](http://en.wikipedia.org/wiki/Design_by_contract)

Contracts programming for C++20 — Current proposal status

( Ein Eiffel Tutorial

Beispiel-Quellen für **EiffelStudio**:

[http://www.math.uni-wuppertal.de/~buhl/teach/exercises/PbC09/source\\_code.tar.gz](http://www.math.uni-wuppertal.de/~buhl/teach/exercises/PbC09/source_code.tar.gz))

## Ein Beispiel C++-Codevertrag mit Hilfe von Nana:

```
#define EIFFEL_CHECK CHECK_ALL
#include <set>
#include <vector>
#include <eiffel.h>
#include <nana.h>
...
void quicksort(double v[], int l, int h)
{
    REQUIRE(l <= h+1);
    ...
    ENSURE(A(int k=l, k<h, k++, v[k]<=v[k+1]));
}
void quicksort(double v[], int n)
{
    REQUIRE(n>=1);
    ID(multiset<double> v_old_contents(&v[0],&v[n]));
    ...
    ENSURE(A(int k=0, k<n-1, k++, v[k]<=v[k+1]));
    ID(multiset<double> v_contents(&v[0],&v[n]));
    ENSURE(v_old_contents == v_contents);
};

class name_list{ ...
void name_list::put(const string& a_name) // Push a_name into list
DO
    REQUIRE(/* name not in list */ !has(a_name));
    ID(set<string> contents_old(begin(),end()));
    ID(int count_old = get_count());
    ID(bool not_in_list = !has(a_name));
    ...
    ENSURE(has(a_name));
    ENSURE( (!not_in_list) || (get_count() == count_old + 1));
    ID(set<string> contents(begin(),end()));
    ENSURE( (!not_in_list) || (contents == contents_old + a_name));
END;
...
}
```

Auswahl der Überprüfungslevel durch:

```
#define EIFFEL_DOEND
#ifndef EIFFEL_CHECK
#define EIFFEL_CHECK CHECK_ALL
//
// CHECK_LOOP Makros CHECK() und folgende
// CHECK_INVARIANT Makros INVARIANT() und folgende
// CHECK_ENSURE Methode invariant() und folgende
// CHECK_REQUIRE Nachbedingungen und folgende
// CHECK_NO Vorgedingungen
#endif
#include "eiffel.h"
#include "nana.h"
```

nana-Manual

## Software Quality Attributes confirming ISO 9126-1

- Funktionalität
  - Angemessenheit
  - Richtigkeit/Sorgfalt
  - Interoperationalität/Kompatibilität
  - Regeltreue
- Ausfallsicherheit
  - Ausgereiftheit
  - Fehlertoleranz
  - Wiederherstellbarkeit
- Bedienbarkeit
  - Verständlichkeit
  - Erlernbarkeit
  - Funktionsfähigkeit
- Effizienz
  - zeitliche Effizienz
  - Ressourcenverbrauch
- Wartungsfreundlichkeit
  - Analysierbarkeit
  - Änderbarkeit
  - Stabilität
  - Testbarkeit
- Portabilität
  - Anpassbarkeit
  - Installierbarkeit
  - Konformität
  - Ersetzbarkeit

Alexa music device held a party on its own

The Sun: Cops raid music fan's flat ...

Smart Home: Nests Neujahrs-Bug lässt Kunden bibbern

MacBook: Facebook- und Messenger-App saugten Akkus leer

Homematic IP-Ausfall

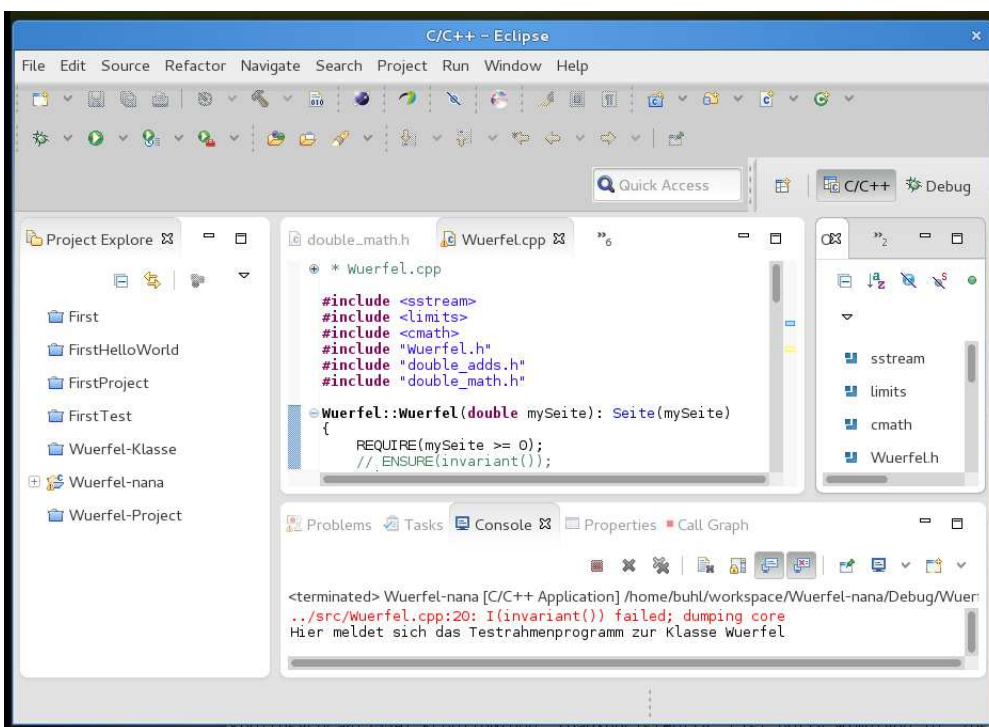
## C++-Entwicklungsumgebung

<http://www.eclipse.org/downloads/packages/release/Oxygen/1> (Eclipse Oxygen 1)

Eclipse für C/C++-Programmierer, 3. Auflage  
Creating Your First C++ Program in Eclipse

Eclipse-Modelling (inkl. Java) mit

- CDT (C/C++ Development Environment inklusive Eclipse standalone Debugger),
- UML (Papyrus mit Papyrus-Designer (CPP- und Java-Codeerzeugung),
- OCL-Tools (Object Constraint Language),
- CDT-Linux Tools (für Gcov, Gprof, Perf, Valgrind, ...),
- PyDev (Python),
- D Development Tools (mit gesondert installiertem dmd, dub),
- Scala,
- Kotlin,
- OcaIDE (Ocaml),
- ...
- CUTE (C++ Unit-Tests),
- cppcheclipse (mit gesondert installierte cppcheck)



## Benutzte freie CDT/UML2.5-/OCL2.4-Programmentwicklung

Hilfsmittel (Tools) zur „state of the art“-Entwicklung von C++-Anwendungen:  
Verfügbar (vorinstalliert) auf allen Ausbildungsklustern (CIP/IT/PI: I101, ...) der  
Fachgruppe Mathematik/Informatik der BUW als `eclipse-papyrus01`.

Hinweis zur Installation auf dem eigenen Linux-Notebook:

## IDE für C++-Programmierung: Eclipse Oxygen Modeling mit CDT, UML, OCL, Linux Tools, ...

Eclipse Downloads (Download Packages):



Installiere *Eclipse Modeling Tools*, (zur Zeit die Version Oxygen 1 durch Download der  
Datei `eclipse-modeling-oxygen-1-linux-gtk-x86_64.tar.gz` von  
<http://www.eclipse.org/downloads/>, installiere sie mittels:

```
/Downloads> gunzip eclipse-modeling-oxygen-1-linux-gtk-x86_64.  
tar.gz
```

```
buhl@rhea3:~/Downloads> ls -al ec*  
-rw-r--r-- 1 buhl users 495882240 22. Aug 09:05 eclipse-  
modeling-oxygen-1-linux-gtk-x86_64.tar
```

wechsle ins Zielverzeichnis für selbstinstallierte Software (etwa `$HOME/sw`) und entpacke  
`eclipse` dorthin:

```
~/sw> tar xf ~/Downloads/eclipse-modeling-oxygen-1-linux-gtk-  
x86_64.tar
```

```
~/sw> ls -al
```

```
drwxr-xr-x 9 buhl users 4096 19. Feb 09:36 eclipse
```

```
~/sw> mv eclipse eclipse-modeling-oxygen-1-linux-gtk-x86_64
```

```
~/sw> cd ~/bin
```

Erzeuge in `$HOME/bin` ein Startskript `$HOME/bin/eclipse-papyrus01` mit dem Inhalt:

```
#!/bin/sh
```

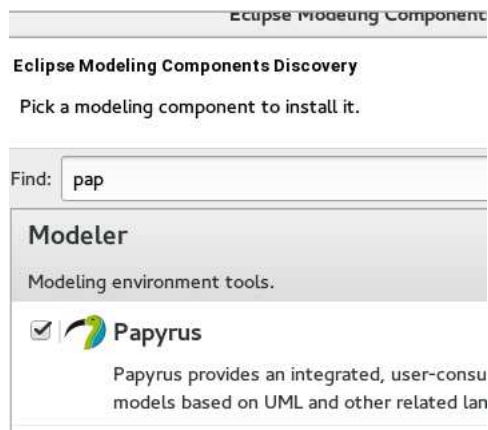
```
#
```

```
$HOME/sw/eclipse-modeling-oxygen-1-linux-gtk-x86_64/eclipse $*
```

und gib ihm Ausführbarkeitsrechte:

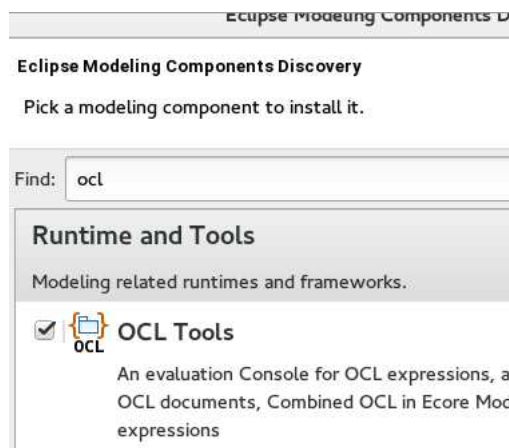
```
~/bin> chmod 755 $HOME/bin/eclipse-papyrus01
```

Ergänze dann unter Help, Install Modelling Components das UML-Tool Papyrus (für die Erstellung von UML-Modellen):



⋮

ergänze dann die OCL Tools zur Erstellung von formalen Constraints (Codeverträge an die UML-Komponenten):

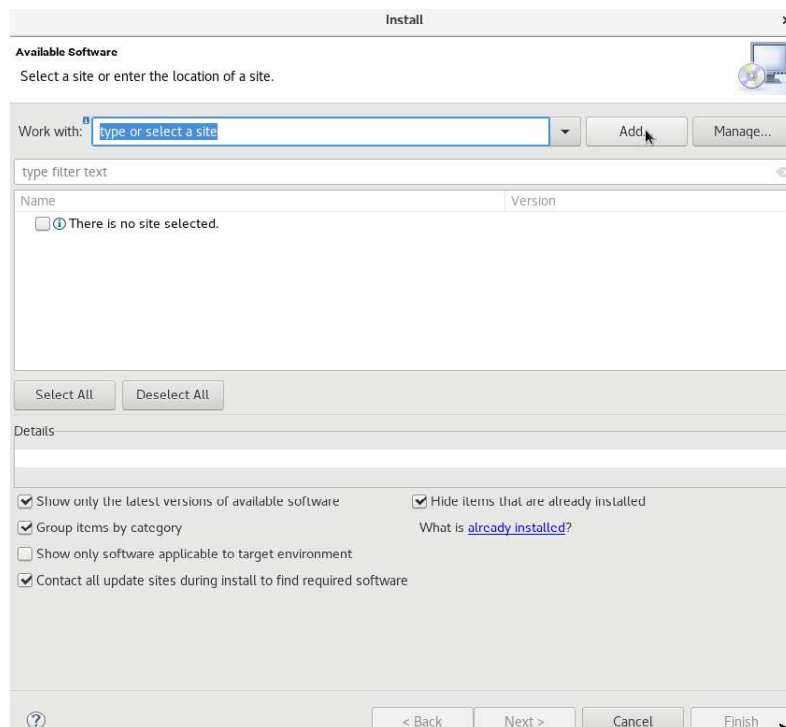
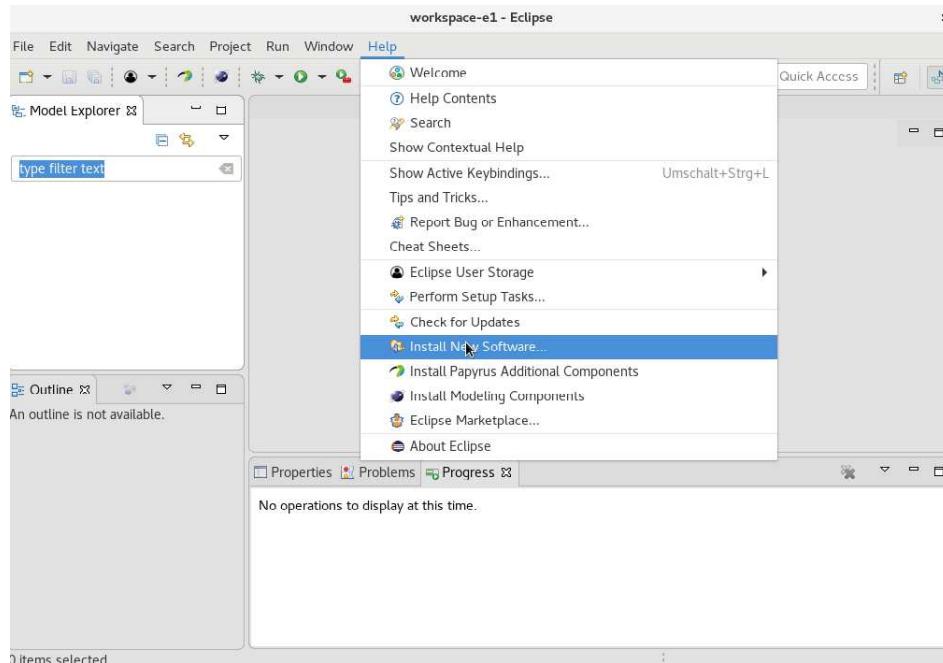


aktualisiere Papyrus unter Help, Check for updates, falls nötig,

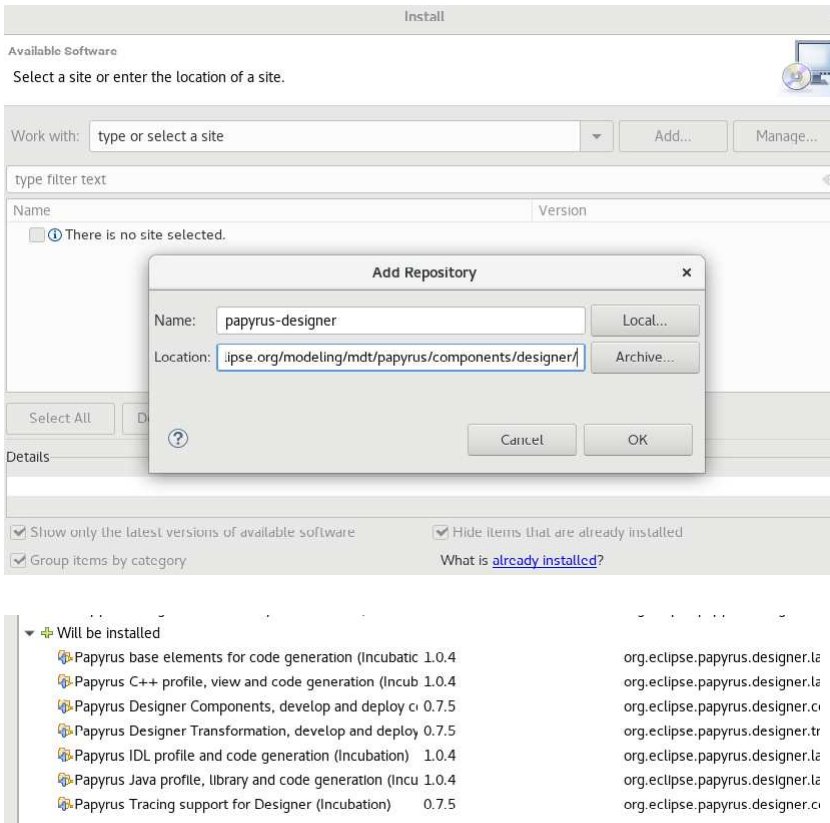
ergänze unter Help, Install New Software, Add den C/C++-Designer (zur Codeerzeugung unter CDT und JAVA):

papyrus-designer

<http://download.eclipse.org/modeling/mdt/papyrus/components/designer/>



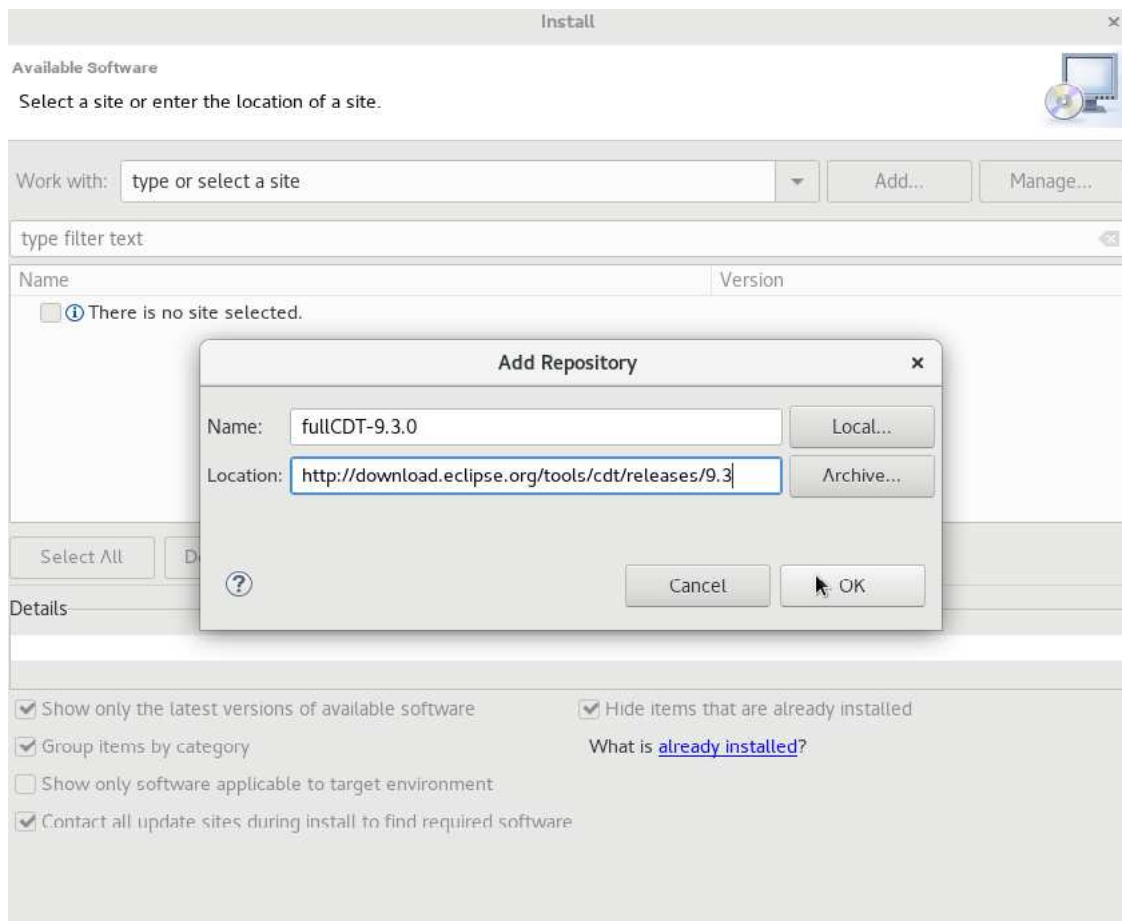




ergänze unter Help, Install New Software, Add

full CDT 9.3.0

<http://download.eclipse.org/tools/cdt/releases/9.3>

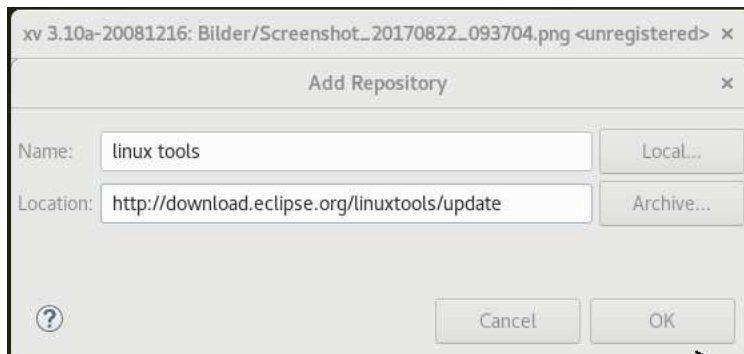


und wähle mittels **Select All** das volle CDT-Plugin aus und installiere es,

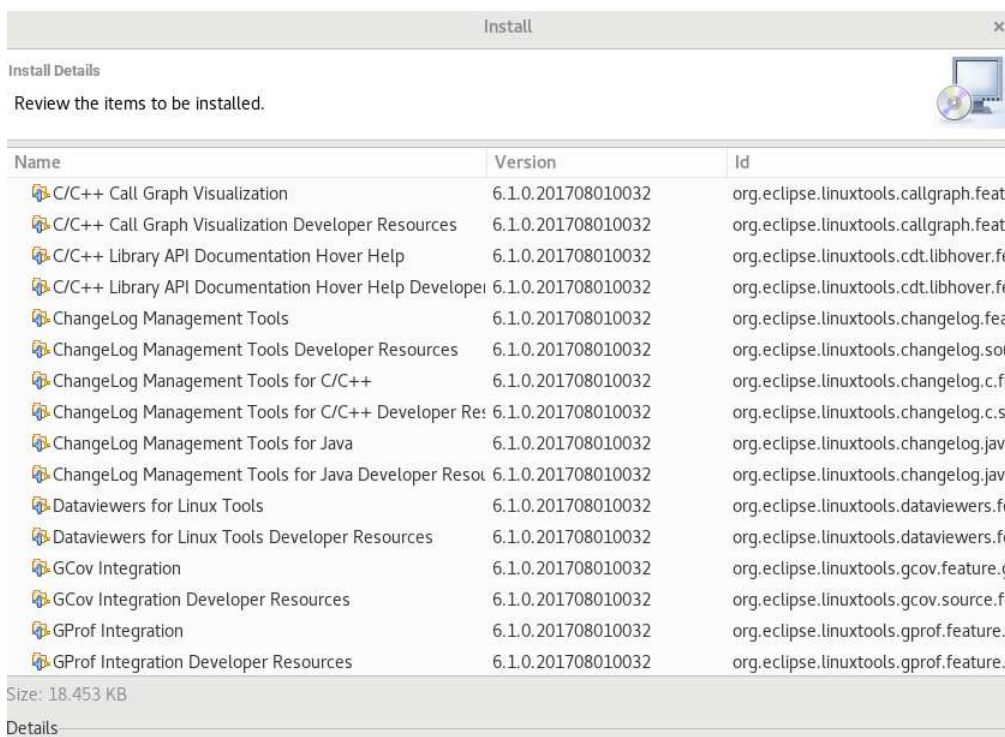
ergänze unter Help, Install New Software, Add

linuxtools

<http://download.eclipse.org/linuxtools/update>



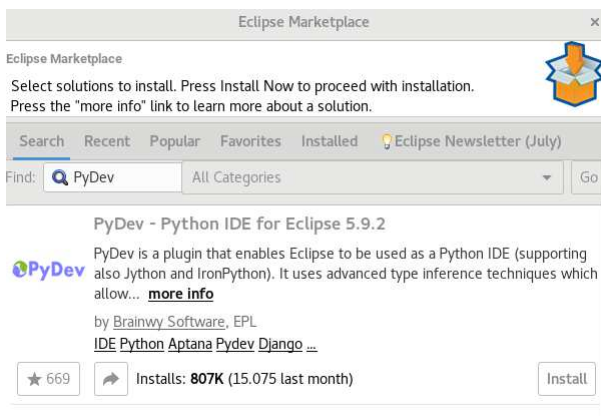
das Linuxtools-Repository und wähle **Select All** an:



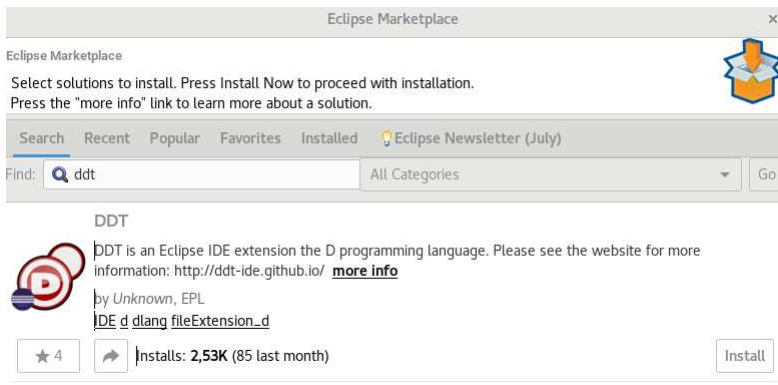
Die Linuxtools bieten Eclipse-Integration qualitätssteigernder Tools für die C++-Entwicklung:

- Callgraph
- ChangeLog
- GProf
- Gcov (oder lcov)
- Libhover
- Man Page
- LTTng
- OProfile
- Perf
- Systemtap
- Valgrind

Bei Bedarf kann man unter Help, Eclipse Marketplace schließlich noch die Python-Entwicklungsumgebung



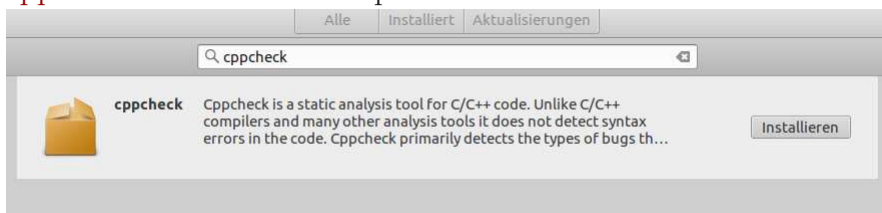
und D-Unterstützung (sofern auf Ihrer Maschine dmd und dub installiert sind)



und ... hinzustallieren.

(Vorinstalliert auf den Ausbildungsclustern der Fachgruppe als eclipse-papyruso sind zum Beispiel zusätzlich: Scala, Kotlin, OcaIDE (für Ocaml) sowie Cute.)

cppcheclipse (Eclipse-Marketplace Plugin für cppcheck) ist nach Installation von **cppcheck** ebenfalls sehr empfehlenswert:



Getting started with CDT development

CDT Documentation, Tutorials, ...

Eclipse CDT (C/C++ Development Tooling)

Eclipse für C/C++-Programmierer, dritte Auflage

Hinweis zu verfügbaren Softwareentwicklungssystemen:

GNU g++ für Linux

gcc7 vor den Toren

Compiler: GCC 7.1 kennt die Sprachfeatures von C++17

GCC, the GNU Compiler Collection

C++17: Standardbibliotheksänderungen

Cygwin für Windows, Cygwin  
mingw-64  
Windows 10 Linux-Subsystem  
C++17 Features In Visual Studio 2017 Version 15.3 Preview

Microsoft Imagine (früher MSDNAA): VisualStudio 201x für Windows

## Weitere Beispiele für Softwaredisfunktionalitäten

### Ein sahniger Brocken

(aus: *Die Zeit* vom 15.09.2005)

Begleitet von großem Werberummel hat die NASA den Kometen Tempel1 beschossen. Nun zeigen die Daten: Getroffen hat sie gut, gelernt hat sie wenig.

Auch wenn in den offiziellen Mitteilungen der NASA keine Rede davon ist - unter den versammelten Astronomen hat sich längst herumgesprochen, dass der Erfolg von *Deep Impact* nicht nur von aufgewirbeltem Feinstaub verdunkelt wurde. Ein Softwarefehler hat dazu geführt, dass die ersten - und besten - Bilder des Zusammenpralls im Datenspeicher des Begleitsateliten von späteren Aufnahmen überschrieben wurden.

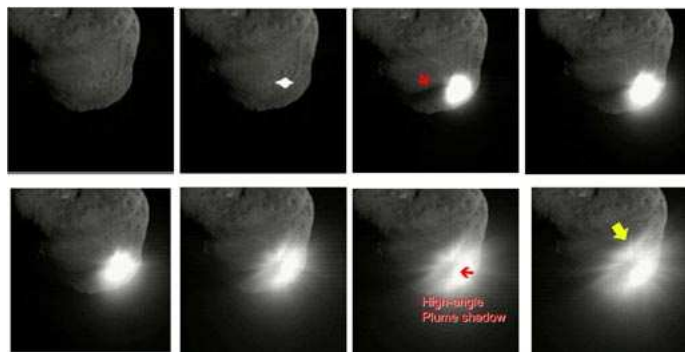


Abbildung 0.2.: Bilder von Deep Impact

Der vollständige Artikel: <http://www.zeit.de/2005/38/komet>

### USV-Software legt Server lahm

**APC**, Hersteller von unterbrechungsfreien Stromversorgungssystemen (USV), rät in einem Knowledgebase-Artikel dazu, alte Versionen der **PowerChute Business Edition-Software 6.X** umgehend durch die Version 7.X zu ersetzen.

Die Software zur Steuerung unterbrechungsfreier Stromversorgungen und zum sicheren Server-Shutdown hat Probleme mit einem auslaufenden Java-Runtime-Zertifikat. Dies führt dazu, dass die Windows-Server, auf denen die alte Version läuft, zum Teil mehrere Stunden für eine Ab- beziehungsweise Anmeldung benötigen. Die Dienste des Servers wie zum Beispiel Netzwerkfreigaben funktionieren allerdings trotz der Anmeldeprobleme weiterhin.

(aus <http://www.heise.de/newsticker/meldung/62344>)

## Chaos an Hannovers Geldautomaten (05.10.2003 13:00 Uhr)

Computerprobleme haben am Samstag alle 240 Geldautomaten der Sparkasse in der Stadt und Region Hannover lahm gelegt. Die Fusion der Stadt- und Kreissparkasse sollte am Wochenende auch technisch umgesetzt werden, sagte der Sprecher des Geldinstituts, Stefan Becker. Beim Hochfahren eines Server habe sich ein Fehler eingeschlichen, so dass die Geldautomaten nicht mehr funktionierten. Die Sparkasse öffnete stattdessen fünf Filialen, damit Kunden etwa in Einkaufszonen Bargeld abheben können.

(aus: <http://www.heise.de/newsticker/meldung/40834>)

## THERAC 25

Selten sind solch schädliche Vorfälle so gut dokumentiert worden wie im Fall des „THERAC 25“, eines computergestützten Bestrahlungsgerätes. Dabei handelt es sich um ein Bestrahlungsgerät, welches in zwei „Modi“ arbeitet: im „X-Modus“ wird ein Elektronenstrahl von 25 Millionen Elektronen-Volt durch Beschuß einer Wolframscheibe in Röntgenstrahlen verwandelt; im „E-Modus“ werden die Elektronen selbst, allerdings „weicher“ mit erheblich reduzierter Energie als Korpuskelstrahlung erzeugt. Je nach therapeutischer Indikation wird die geeignete Strahlungsart eingestellt; in beiden Fällen kann der Bestrahlungsverlauf, nach Modus, Intensität und Bewegungskurve der Strahlungsquelle, mit einem Bildschirm-„Menü“ eingegeben werden.

Als mehrere Patienten berichteten, sie hätten bei Behandlungsbeginn das Gefühl gehabt, „ein heißer Strahl“ durchdringe sie, wurde dies vom Hersteller als „unmöglich“ zurückgewiesen. Erst nach dem Tod zweier Patienten sowie massiven Verbrennungen bei weiteren Personen kam heraus, daß neben dem X- sowie E-Modus mit niedriger Elektronenintensität infolge Programmierfehler ein unzulässiger dritter Zustand auftrat, nämlich direkt wirkende, 25 Millionen Elektronen-Volt „heiße“ Elektronen.

Dies geschah immer dann, wenn ein vorgegebenes „Behandlungsmenü“ mittels Curser-Taste modifiziert wurde. Um aufwendige Umprogrammierung zu vermeiden, wollte der kanadische Hersteller die Benutzung der Curser-Taste verbieten bzw. diese ausbauen und die Tastenlücke mit Klebeband abdichten lassen! Es ist zu befürchten, daß der Fall „THERAC 25“ kein Einzelfall ist. Zumeist ist es mangels entsprechender Vorsorge in computergesteuerten Medizingeräten schwerlich möglich, schädliches Systemverhalten später aufzuklären.



## Berliner Magnetbahn

Computer spielen in allen gesellschaftlichen Bereichen eine immer größere Rolle. Angesichts der von fehlerhafter Software ausgehenden Gefahr wird versucht, die Sicherheit von computergesteuerten Systemen so weit wie möglich zu garantieren.

### **Softwarefehler: Kleine Ursache, große Wirkung**

Fünf - Null, tippt der Operator in die Tastatur und erwartet, daß die Magnetschwebbahn auf 50 Stundenkilometer beschleunigen würde. Doch nichts geschah. Wieder tippt er fünf - null und vergaß diesmal nicht die „Enter“-Taste zu betätigen, mit der die Daten erst in den Rechner abgeschickt werden. Die insgesamt eingegebene Tastenfolge „fünf - null - fünf - null“ interpretiert der Rechner als Anweisung, auf unsinnige 5050 Stundenkilometer zu beschleunigen. Dies konnte die Bahn zwar nicht, aber immerhin wurde sie so schnell, daß sie nicht mehr rechtzeitig vor der Station gebremst werden konnte. Es kam zum Crasch mit Personenschaden – so geschehen vor zwei Jahren bei einer Probefahrt der Berliner M-Bahn.

Vernünftigerweise hätte die den Computer steuernde Software die Fehlerhaftigkeit der Eingabe „5050“ erkennen müssen. Schon dieses Beispiel mangelnder Software zeigt, von welcher Bedeutung das richtige Verhalten von Computerprogrammen sein kann. Nicht nur bei Astronauten, die mit softwaregesteuerten Raumfähren ins All starten, hängt heute Leben und Gesundheit von Software ab. Computerprogramme erfüllen mittlerweile in vielen Bereichen sicherheitsrelevante Aufgaben.

## Elektronik-Fehler führt zu Überhitzung bei Volvo-PKW

Kaum ein KFZ-Hersteller, der nicht mit Elektronik, Software und Hightech-Ausstattung das Autofahren komfortabler und die Wartung in der Werkstatt einfacher machen will. Doch die Tücken der Technik lassen für manchen Kunden den PKW zum IT-Sicherheitsrisiko werden. Nachdem vor kurzem erst Softwarefehler bei Mercedes-Dieseln für Aufsehen sorgten, können nun Defekte in der elektronischen Steuerung der Motorkühlung bei Volvo-Personenwagen zur Überhitzung führen.

Der Fehler tritt bei den Modellen S60, S80, V70 und XC70 aus den Baujahren 2000 und 2001 auf, erklärte Volvo, einzelne Modelle aus dem Jahr 1999 seien ebenfalls betroffen. Die fehlerhaft arbeitende Elektronik hat Bosch an Volvo geliefert – wer für den Fehler, der vor allem bei langsamer Fahrt bei hohen Außentemperaturen zur Überhitzung führen kann, verantwortlich ist, steht laut Volvo noch nicht fest. Insgesamt 460.000 Fahrzeuge weltweit ruft der schwedische Hersteller daher in die Werkstätten zurück. Laut dpa erhalten in Deutschland rund 40.000 Besitzer eines Volvo-PKW eine Aufforderung zum Werkstattbesuch – der für die Halter zumindest kostenlos bleibt.

(aus: <http://www.heise.de/newsticker/meldung/51019>)

## The Patriot Missile

The Patriot missile defense battery uses a 24 bit arithmetic which causes the representation of real time and velocities to incur roundoff errors; these errors became substantial when the patriot battery ran for 8 or more consecutive hours.

As part of the search and targeting procedure, the Patriot radar system computes a "Range Gate" that is used to track and attack the target. As the calculations of real time and velocities incur roundoff errors, the range gate shifts by substantial margins, especially after 8 or more hours of continuous run.

The following data on the effect of extended run time on patriot operations from Appendix II of the report would be of interest to numerical analysts anywhere.

HOURS	REAL TIME (seconds)	CALCULATED TIME (seconds)	INACCURACY (seconds)	APPROXIMATE SHIFT IN RANGE GATE (meters)
0	0	0	0	0
1	3600	3599.9966	.0034	7
8	28800	28799.9725	.0275	55
20a	72000	71999.9313	.0687	137
48	172800	172799.8352	.1648	330
72	259200	259199.7528	.2472	494
100b	360000	359999.6667	.3333*	687

Table 0.1.: Divergence in the Range Gate of a PATRIOT MISSILE

a: continuous operation exceeding 20 hours-target outside range gate

b: Alpha battery [at Dhahran] ran continuously for about 100 hours

\* corrected value [GAO report lists .3433]

On February 21, 1991 the Partiot Project Office send a message to all patriot sites stating that very long run times "could cause a shift in the range gate, resulting in the target being offset". However the message did not specify "what constitutes very long run times". According to the Army officials, they presumed that the users would not run the batteries for such extended periods of time that the Patriot would fail to track targets. "Therefore, they did not think that more detailed guidance was required".

The air fields and seaports of Dhahran were protected by six Patriot batteries. Alpha battery was to protect the Dhahran air base.

On February 25, 1991, Alpha battery had been in operation for over 100 consecutive hours. That was the day an incomming Scud struck an Army barracks and killed 28 American soldiers.

On February 26, the next day, the modified software, which compensated for the inaccurated time calculation, arrived in Dhahran.

### **Kontenabrufverfahren startet wegen Softwareproblemen als Provisorium**

Das automatische Kontenabrufverfahren nach dem „Gesetz zur Förderung der Steuerehrlichkeit“, das ab dem 1. April die Abfrage der Kontostammdaten für einige Behörden möglich macht, startet mit Anlaufproblemen. Sie liegen vor allem darin begründet, dass die entsprechende Abfragesoftware der Stammdaten, die ab November 2003 zum Zwecke der Terroristenfahndung entwickelt wurde, nicht richtig skaliert. Diese Software wurde auf ca. 2000 Abfragen pro Tag durch die Polizeifahnder ausgelegt. Mit mehr als täglichen 50.000 Abfragen, die von Finanzämtern, Bafög- oder Sozialämtern ab dem 1. April erwartet werden, ist die Software hoffnungslos überfordert. Für die 18 bis 20 Millionen Konten, die jährlich nach dem Willen des Gesetzgebers gesucht werden sollen, wird derzeit eine völlig neue Schnittstellenspezifikation entwickelt und ein komplett neues Programm geschrieben. Bis dieses Programm für die automatische Abfrage durch die Sachbearbeiter fertig ist, muss die Abfrage wie bisher manuell erfolgen.

Bei dieser manuellen Abfrage reichen Polizeibehörden und Strafverfolger ihre Anfragen auf Papier oder per Fax oder E-Mail bei der Bundesanstalt für Finanzdienstleistungsaufsicht (BaFin) ein und bekommen die gewünschten Kontodaten auf demselben Wege zurück. Dieses Verfahren soll durch eine Suchmaske ersetzt werden, die jede Behörde aufrufen kann – wenn die dahinter liegende Abfragesoftware die Datenmengen bewältigen kann.

(aus: <http://www.heise.de/newsticker/meldung/58096>)

### **Buffer Overflow im Linux-Kernel**

Paul Starzetz von isec hat Details zu einer neuen Lücke im Linux-Kernel veröffentlicht, mit der ein Angreifer Programme mit Root-Rechten ausführen kann. Anders als bei vergangenen Veröffentlichungen von Starzetz, wurden die Hersteller aber offenbar nicht vorab informiert, etwa über die geschlossene Mailing-Liste Vendor-Sec. Nach seinen Angaben würde die Linux-Community Veröffentlichungen ohne Embargos von Distributoren bevorzugen. Um aber die Regeln der so genannten Responsible Disclosure einzuhalten, veröffentlicht er diesmal keinen Exploit-Code.

Der Fehler findet sich wieder einmal im Linux ELF-Binary-Loader, in dem Starzetz in der Vergangenheit bereits mehrere Lücken aufdeckte. Diesmal ist ein Buffer Overflow in der Funktion `elf_core_dump` schuld, der beim Aufruf einer

weiteren Funktion (`copy_from_user`) mit einer negativen Längenangabe auftritt. Starzetz hat nach eigenen Angaben die Lücke bereits durch ein präpariertes ELF-Binary demonstrieren können, das mit Kernel-Privilegien lief. Ein Proof-of-Concept-Programm ist seinem Advisory beigefügt, das aber nur den Kern des Problems demonstriert.

(aus: <http://www.heise.de/newsticker/meldung/59498>)

## **Auch Superhirne können irren - das Risiko Computer**

Lenkwaffen, Flugsteuerungen, Diagnosegeräte, Verkehrsleitsysteme, Dateien, Produktions-Steuerung – überall hat der Computer das Kommando übernommen. Doch nicht überall gibt er die richtigen Befehle. Mancher Irrtum schon hatte tödliche Folgen. Das Vertrauen in das elektronische Superhirn ist angeschlagen.

Sollten US-Kriegsschiffe, die mit dem computergestützten Waffensystem „Aegis“ ausgerüstet sind, in Zukunft wieder in Spannungsgebieten kreuzen, werden die verantwortlichen Offiziere dort mit der Angst leben, daß sich die Ereignisse des 3. Juli 1988 wiederholen könnten: Damals folgte der Kapitän des Kreuzers „Vincennes“, von elektronischen Befehlen unter Entscheidungsdruck gesetzt, der Logik des Computers, dessen Abtastsystem ein Verkehrsflugzeug mit einer Kampfmaschine verwechselte. Er gab den verhängnisvollen Befehl zum Abfeuern der Raketen. Alle 290 Insassen des iranischen Airbus kamen dabei ums Leben. ...

Aus anderer Quelle:

Auch der erste KI-Unfall, bei dem das „künstlich intelligente“ AEGIS-System des US-Kreuzers „Vincennes“ im Sommer 1988 einen zivilen Airbus mit einem MIG-Militärjet verwechselte, dürfte bei heutigem Kenntnisstand durch einen Konzeptfehler mitverursacht worden sein. Aus der „Sicht“ des einzelnen AEGIS-Systems werden alle Signale, die auf einem Richtstrahl innerhalb einer 300 Meilen umfassenden Überwachungszone entdeckt werden, einem einzelnen Objekt zugeordnet. So können ein Militär- und ein Zivil-Jet nur durch ein räumlich getrenntes System unterschieden werden. Offenbar hat das AEGIS-System aber weder Inkonsistenzen der Daten (militärische und zivile Transponder-Kennung) noch die unvollständige räumliche Auflösung dem verantwortlichen Kommandeur übermittelt, der im Vertrauen auf die Datenqualität den Befehl zum Abschluß von fast 300 Zivilisten gab. Offensichtlich ist in Streßsituationen eine menschliche Plausibilitätskontrolle nicht nur bei derart komplexen Systemen erschwert. Aus einem bis dahin fehlerfreien Funktionieren wird induktiv auf korrektes Verhalten im Ernstfall geschlossen. Daher sind besondere Hinweise auf inkonsistente und

unvollständige „Datenlagen“ und gegebenenfalls Sperren gegen automatische Prozeduren zwingend erforderlich.

## Explosion der Ariane 5

<http://www.ima.umn.edu/~arnold/disasters/ariane5rep.html>

## Neueste Risikoinformationen/Softwareprobleme

... findet man unter: <http://catless.ncl.ac.uk/Risks>:

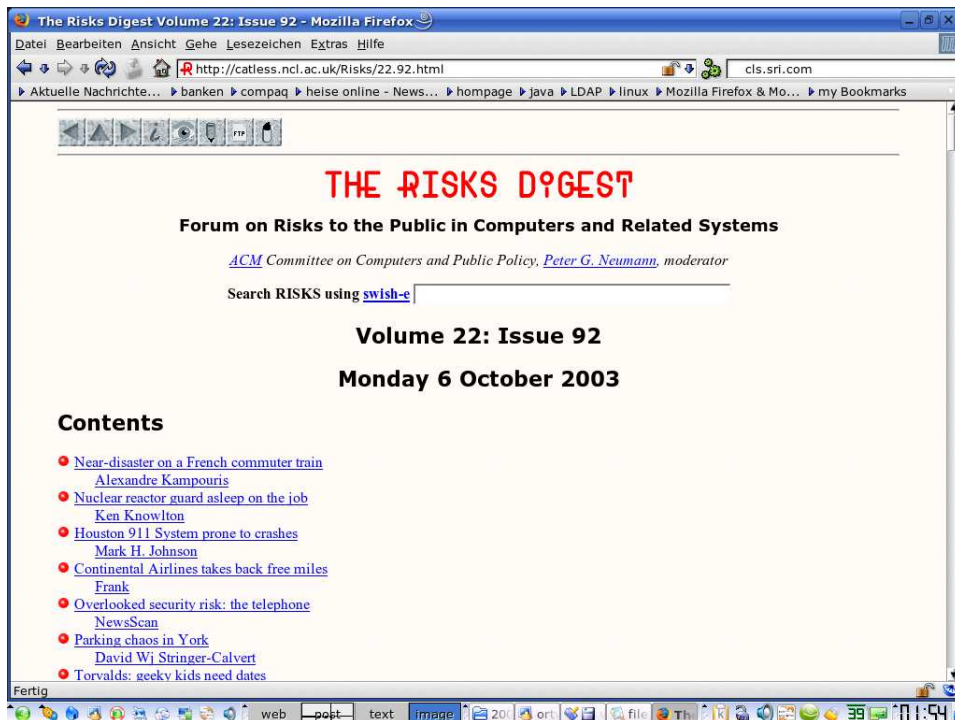


Abbildung 0.3.: <http://catless.ncl.ac.uk/Risks/22.92.html>

## Codeverträge / Contracting:

Design by contract

An introduction to Design by Contract

Contracts programming for C++20



# 1. Softwarequalität

## 1.1. Komponententests/Unit-Tests

Modultest/Komponententest/Unittest  
Unit-Test-Suites

CxxTest unit testing framework for C++: User Guide  
Test example  
Assert-Makros

Eclipse plug-in for C++ unit testing with CUTE  
CUTE users guide  
CUTE example: string reverse(), StringUtils.h, StringUtils.cpp, CUTE-Test in Aktion

### JUnit



Eclipse JUnit 5 Support  
Java Test Annotation

C++11 attribute specifier sequence

D programming language Unit Tests

Python unittest — Unit testing framework

Extreme programming  
test-driven development (TDD)

## 1.2. Refactoring

Refactoring

Refactoring in Eclipse CDT

Refactoring 2004, 2008, 2011

## 1.3. Ariane 5 Failure - Full Report

ARIANE 5, Flight 501 Failure, Report by the Inquiry Board

The Explosion of the Ariane 5:

*The internal SRI\* software exception was caused during execution of a data conversion from 64-bit floating point to 16-bit signed integer value. The floating point number which was converted had a value greater than what could be represented by a 16-bit signed integer.*

Ariane 5: 16 bit signed integer overflow

Ada Programming/Exceptions

Integer overflow

## 1.4. Bug plötzlicher unbeabsichtigter Automobilbeschleunigung, ...

Toyota uncontrolled acceleration

Toyota vehicle recall

plötzliche unbeabsichtigte Automobilbeschleunigung

Smartes Pedal

NASA-Gutachten

Hält Toyota Blackbox-Daten zurück?

US-Gericht: Motorelektronik von Toyota schuld an Unfall

Toyota Case: Single Bit Flip That Killed



## Aktuelle Rückrufaktion

Probleme bei der Airbag-Software — Audi ruft 850.000 A4 zurück

## Fehlende Hochverfügbarkeit

Notrufnummer wegen Softwarebug nicht erreichbar

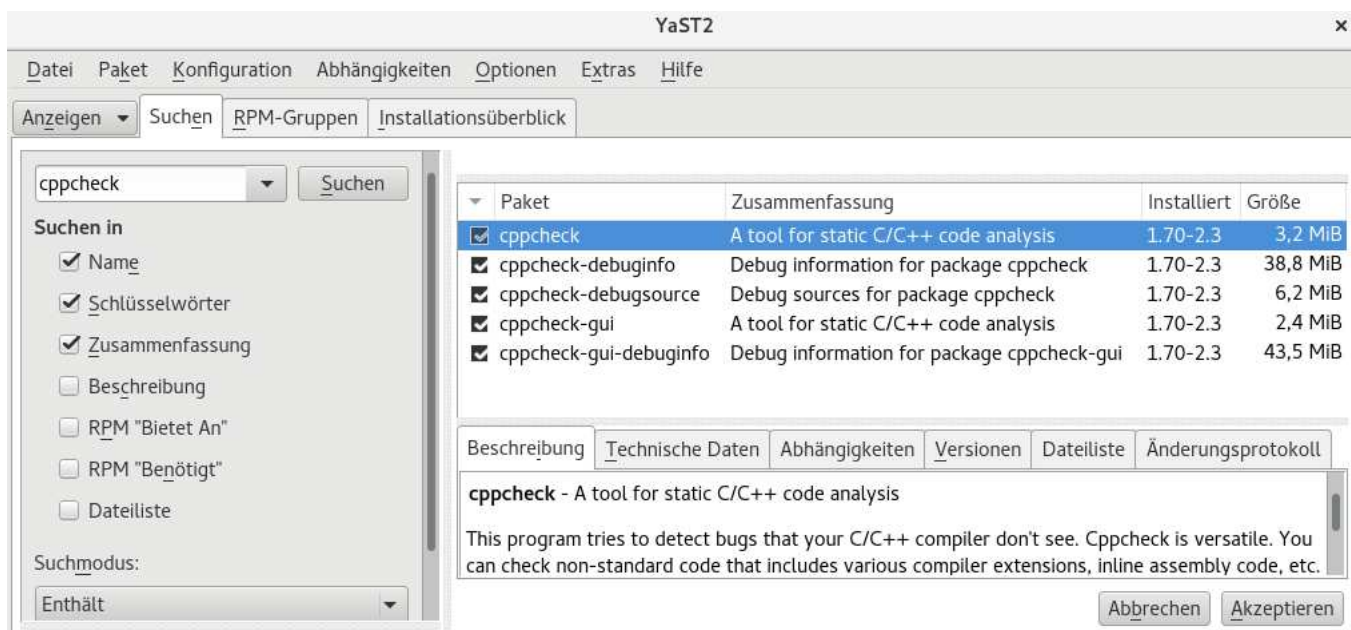
Großstörung bei der Telekom

DSL-Großstörung: 100.000 Betroffene auch in Großbritannien

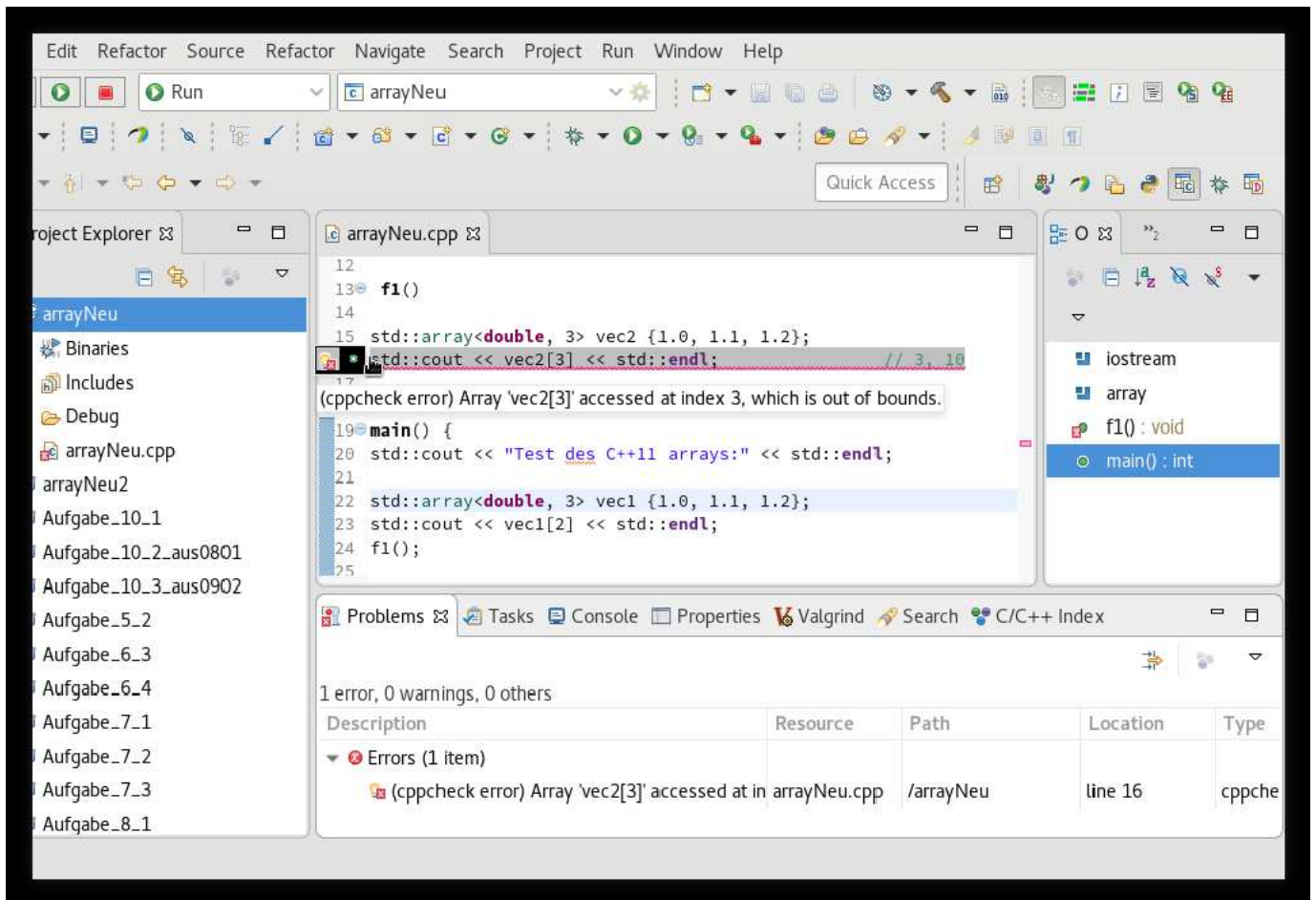
# 1.5. cppcheck CodAn-Tool

Cppcheck — A tool for static C/C++ code analysis (aktuelle Version: 1.81)

Download und Installation von cppcheck (aktuelle Version: 1.70) unter Opensuse Leap 42.2:



cppcheck in Aktion:



Siehe auch [cppcheclipse](#) auf Seite 37.

## 1.6. Exceptions/Trap/Ausnahmebehandlung

[Ausnahmebehandlung](#)  
[Exception handling](#)  
[Exception safety](#)  
[Coding by exception](#)

### Java:

```
$ Exception in thread "Thread-0" java.lang.NullPointerException
    at
org.eclipse.ui.internal.ide.IDEWorkbenchPlugin$1.run(IDEWorkbenchPlugin.
java:356)
    at org.eclipse.swt.widgets.Display.timerProc(Display.java:5171)
    at org.eclipse.equinox.launcher.JNIBridge._takedown_splash(Native
Method)
    at
org.eclipse.equinox.launcher.JNIBridge.takeDownSplash(JNIBridge.java:170)
    at org.eclipse.equinox.launcher.Main.takeDownSplash(Main.java
:2217)
    at org.eclipse.equinox.launcher.Main$SplashHandler.run(Main.java
:115)
...

```

[Java Exception Hierarchy](#)  
[Exception Handling](#)  
[Exceptions](#)  
[How to throw exception in java with example](#)  
[Exceptions in Java](#)  
[How to Specify and Handle Exceptions in Java](#)  
[Do you \(really\) write exception safe code?](#)  
[Java Exception Interview Questions and Answers](#)

### C++:

[std::exception](#)  
[C++ Exception Handling](#)  
[Exceptions and Error Handling](#)  
[Exception Safety: Concepts and Techniques](#)  
[Exception handling mistakes and how to avoid them.](#)  
[Katastrophenschutz mit try und catch](#)  
[Exceptions and Error Handling](#)  
[How to free memory in try-catch blocks?](#)  
[Creating custom exceptions in C++](#)

## 1.7. Spezifikation einer abstrakten Datenkapsel

### 1.7.1. Axiomatische Spezifikation

TYPES		
	STACK[X]	
FUNCTIONS		
empty:	STACK[X]	→ BOOLEAN
new:		→ STACK[X]
push:	X x STACK[X]	→ STACK[X]
pop:	STACK[X]	→ STACK[X]
top:	STACK[X]	→ X
PRECONDITIONS		
	pre pop (s: STACK[X])	= (not empty(s))
	pre top (s: STACK[X])	= (not empty(s))
AXIOMS		
for all x:X, S : STACK[X]:	empty(new())	
	not empty (push(x,S))	
	top (push(x,S))=x	
	pop (push(x,S))=S	
Vollständigkeit + Widerspruchsfreiheit (+ Unabhängigkeit)		

### 1.7.2. Beschreibende (denotationale) Spezifikation

$Queue = Qelem^*$	
$q_0 = [ ]$	
ENQUEUE ( $e : Qelem$ )	
<b>ext wr</b>	$q : Queue$
<b>post</b>	$q = \overleftarrow{q} \hat{\sim} [e]$
DEQUEUE() $e : Qelem$	
<b>ext wr</b>	$q : Queue$
<b>pre</b>	$q \neq [ ]$
<b>post</b>	$\overleftarrow{q} = [e] \hat{\sim} q$
ISEMPTY() $r : \mathbb{B}$	
<b>ext rd</b>	$q : Queue$
<b>post</b>	$r \Leftrightarrow (len\ q = 0)$

„mathematische“ Modellierung  
mit Hilfe von  
Folgen, Mengen, ...  
vergleiche **VDM**

Aktuellere VDM-Versionen benutzen von Programmierern besser nutzbare reine ASCII/Unicode VDM-Versionen ohne Formelschreibweise: [Stackspezifikation](#)

```
public AddExpertToSchedule: Period * Expert ==> ()
AddExpertToSchedule(p,ex) ==
  schedule(p) := if p in set dom schedule
                then schedule(p) union {ex}
                else {ex};
```

and the RemoveExpertFromSchedule operation can be expressed as:

```
public RemoveExpertFromSchedule: Period * Expert ==> ()
RemoveExpertFromSchedule(p,ex) ==
  let exs = schedule(p) in
    schedule := if card exs = 1
                then {p} <:-: schedule
                else schedule ++ {p |-> exs \ {ex}}
```

Overture

Abstract data type

Example: Associative array

Abstrakter Datentyp

Heute wird alternativ als geeignete denotationelle Spezifikationsprache die **OCL** immer beliebter:

Listing 1.1: OCL-Spezifikation Datum

```
context Datum
inv tagGueltig: tag >= 1 and tag <= 31
inv:      monat >=1 and monat <= 12
inv:      jahr >= 1600 and jahr <= 2500
```

oder verbessert:

Listing 1.2: genauere OCL-Spezifikation Datum

```
context Datum — virtuelle Methode = Hilfsmethode / OclHelper
static def: gueltigesDatum( t : Integer, m : Integer, j :
  Integer ) : Boolean =
  1920 <= j and j <= 2500 and
  1 <= m and m <= 12 and
  1 <= t and
  t <=
    if      Set{4, 6, 9, 11}->includes(m)
    then 30
  else if Set{1, 3, 5, 7, 8, 10, 12}->includes(m)
    then 31
  else if ((j.mod(4)=0) and
            not(j.mod(100)=0)) or
            (j.mod(400)= 0)
    then 29 — Schaltjahr
    else 28

    endif
  endif
  endif

context Datum
inv: gueltigesDatum(tag, monat, jahr)

context Datum::Datum( t : Integer, m : Integer, j : Integer ):
  Datum
pre: gueltigesDatum(t, m, j)
post: result.oclIsNew()
post: result.tag = t
post: result.monat = m
post: result.jahr = j
```

### 1.7.3. Spezifikation durch Codeverträge

siehe [SdV](#) (Übungsblatt 3, Aufgabe 2f.) und Rest dieser Veranstaltung.

Feature	ISE Eiffel 5.4	D	C++ Proposal
keywords	require, ensure, do, require else, ensure then, invariant, old, result, variant	in, out, body, invariant, assert, static	precondition, postcondition, invariant, oldof
on failure	throws exception	throws exception	defaults to <code>terminate()</code> , defaults can be customized, might throw
expression copying in postconditions	yes, <code>old</code> keyword	no	yes, <code>oldof</code> keyword
subcontracting	yes	yes	yes, but only considers postconditions
contracts on abstract functions	yes	no (planned)	yes
arbitrary code contracts	yes	yes	no, must be const correct
function code ordering	pre -> body -> post	pre -> post -> body	pre -> post -> body
compile-time assertions	no	yes	no
loop invariants	yes	no	no
loop variants	yes	no	no
const-correct	no	no	yes
invariant calls	<ul style="list-style-type: none"> <li>• end of "constructor"</li> <li>• around public functions</li> </ul>	<ul style="list-style-type: none"> <li>• end of constructor</li> <li>• around public functions</li> <li>• start of destructor</li> </ul>	<ul style="list-style-type: none"> <li>• as in D</li> <li>• when a function exits due to an exception</li> </ul>
disabling of checks during assertions	yes	no	yes, but not in preconditions
when public func. call public func.	disable all checks	disable nothing	disable nothing
removable from object code	yes	yes	yes

## 1.8. iContract (Java)

[iContract, Seite 25: Beispiel](#)

[iContract: Design by Contract in Java](#)

[iContract: Design by Contract in Java, Fortsetzung](#)

[iContract instrumentation and usage](#)

## 1.9. D contracts

[Contract Programming](#)

[Contract programming](#)

Ein Beispiel:

```

import std.stdio;
import std.math;
import std.conv;
import core.exception;
import std.process;
import std.string;
import std.path;

pure nothrow @safe bool withinEpsilonOf(real left, real right,
    real delta){
    return fabs(left-right)<=delta;
}

pure nothrow @safe bool approximatelyEqualTo(real left, real
    right, real factor = 1.0){
    return fabs(left-right)<=real.epsilon*factor*fmax(fabs(left),
        fabs(right));
}

pure nothrow @safe real sqrtm(real x, int m)
    in{assert((m>1)&&(x>=0.0), "Vorbedingung sqrtm verletzt
        !");}
    out(result){assert(approximatelyEqualTo(pow(result,m),x
        ,2.0),
        "Genauigkeit konnte nicht
        erreicht werden!");}

    body{
        if (x==0) return 0;
        real approx = x/m;
        real better = approx*(m-1+x/pow(approx,m))/m;
        while (better != approx) {
            approx = better;
            better = approx*(m-1+x/pow(approx,m))/m
                ;
        }
        return approx;
    }

pure nothrow @safe real myown_sqrt(real x)
    in{assert(x >= 0.0, "Argument ist negativ!");}
    out(result){assert(approximatelyEqualTo(result*result, x, 1.0)
        ,
        "Genauigkeit konnte nicht erreicht
        werden!");}

```



```

body{
    real xold, xi, result;
    if (x == 0.0) {
        result = 0.0;
    } else {
        xi = x / 2.0;
        do{
            //writeln(xi);
            xold = xi;
            xi = 0.5 * (xi + x/xi);
        } while (xi != xold);
        result = xi;
    }
    return result;
}

/*!
 * @mainpage 3-dimensionale Wuerfel
 *
 * date 10. Februar 2013
 * version 0.9
 * Copyright Public Domain
 * author HJB
 */

/*!
 * 3D-Wuerfel
 *
 */
class Wuerfel{

    private real seite;

    public void Seite(real w)
        in{assert(w >= 0.0); assert(!isInfinity(seite));}
        out{assert(seite == w);}
        body{seite = w;}

    public real Seite() const
        out(result){assert(result == seite);}
        body{return seite;};

    this(real mySeite = 1.0)
        in{assert(mySeite >= 0.0); assert(!isInfinity(seite));}

```

```

    out{assert(seite==mySeite);}
    body{
        this.seite=mySeite;
    }

~this(){
    writeln("Wuerfel destruiert!\n");
}

/*!
 * @invariant{seite >= 0 && !isNaN(seite) && !isInfinity(seite)}
 */
const invariant() {
    assert(seite >= 0.0);
    assert(!isNaN(seite) && !isInfinity(seite));
}

override string toString() const {
    return text("Wuerfel(", seite, "; ", Oberflaeche(), "; ",
                Volumen(), "; ", Raumdiagonale(), ") ");
}

void verdoppleWuerfel()
    // out{assert(approximatelyEqualTo(pow(old(seite),3),pow(
    //   seite,3),600.0));}
    //out{assert(approximatelyEqualTo(pow(seite@pre,3),pow(
    //   seite,3),600.0));}
    body{
        real old_volumen = Volumen();
        seite *= pow(2.0, 1.0/3.0);
        assert(approximatelyEqualTo(Volumen(),
            old_volumen*2.0, 600.0));
    }

unittest{
    auto w = new Wuerfel();
    assert(w.Seite == 1.0);
    real old_volumen = w.Volumen();
    w.verdoppleWuerfel();
    assert(approximatelyEqualTo(w.Volumen(), 2.0 *
        old_volumen, 600.0));
}

```

```

    assert (approximatelyEqualTo(w. Oberflaeche ()
        ,9.52440631180919723957,1.00));
    assert (approximatelyEqualTo(w. Volumen () ,2.0,600.0));
    assert (approximatelyEqualTo(w. Raumdiagonale ()
        ,2.1822472719434427256,1.0));

    //writefln ("%1.20f",w. Raumdiagonale ());

    w = new Wuerfel (2.1);
    assert (approximatelyEqualTo(w. Oberflaeche () ,26.46,2.00))
        ;
    assert (approximatelyEqualTo(w. Volumen () ,9.261,2.0));
    assert (approximatelyEqualTo(w. Raumdiagonale ()
        ,3.6373066958946421055,1.0));

    //writefln ("%1.20f",w. Raumdiagonale ());
}

real Oberflaeche () const
    out(result){
        assert (approximatelyEqualTo(result , 6.0 * seite * seite ,
            1.0));
    } body {
        return 6.0 * pow(seite ,2);
    }

real Volumen () const
    out(result){
        assert (approximatelyEqualTo(result ,seite*seite*seite ,1.0));
    } body {
        return pow(seite ,3);
    }

real Raumdiagonale () const
    out(result){
        assert (approximatelyEqualTo(result ,seite*myown_sqrt (3.0)
            ,600.0));
    } body {
        return seite * sqrt (3.0);
    }
}

int main(char [][] args)
{

```

```

try{
    writeln(" Testlauf  Wuerfel:\n");

    auto w1= new Wuerfel();
    writeln(w1.Seite);
    writeln(w1); writeln();

    w1.verdoppleWuerfel();
    writeln(w1.Seite);
    writeln(w1); writeln();

    w1.Seite=2.1;
    writeln(w1.Seite);
    writeln(w1); writeln();

    writeln(myown_sqrt(3.0)); writeln();

    writeln(sqrtm(512.0,9));
    writeln(sqrtm(0.000001,6));
    //writeln(sqrtm(-1.0,2));
    writeln(sqrtm(0.0,3)); writeln();

    writeln(real.epsilon, " ", 600.0*real.epsilon);
    writeln(double.epsilon); writeln();

} catch (AssertError e){
    writeln(" Fehlerabbruch:");
    writeln("   in Zeile ", e.line, " von Datei ",
            stripExtension(e.file), ".d");
    writeln(" Backtrace:");
    writeln(e);
    auto file = stripExtension(baseName(e.file));
    auto filename = "src/" ~ file ~ ".d";
    system("emacs" ~ " " + " ~ to!string(e.line) ~ " " ~
            filename ~ "&");
}

writeln("Programm endet.");
return 0;
}

```

## 1.10. Prinzipien der ordnungsgemäßen Programmerstellung

1. Konstruktive Voraussicht und methodische Restriktion
2. **Strukturierung**
3. **C++/JAVA** Modularisierung
4. Lokalität
5. Integrierte Dokumentation
6. Standardisierung
7. Funktionale und informelle Bindung
8. Schmale Datenkopplung
9. Vollständige Schnittstellenspezifikation
10. Lineare Kontrollstrukturen
11. Verbalisierung

## 1.11. sanitize-Optionen in gcc 5

**GCC 5 Release Series Changes, New Features and Fixes:**

UndefinedBehaviorSanitizer gained a few new sanitization options:

- `-fsanitize=float-divide-by-zero`: detect floating-point division by zero;
- `-fsanitize=float-cast-overflow`: check that the result of floating-point type to integer conversions do not overflow;
- `-fsanitize=bounds`: enable instrumentation of array bounds and detect out-of-bounds accesses;
- `-fsanitize=alignment`: enable alignment checking, detect various misaligned objects;
- `-fsanitize=object-size`: enable object size checking, detect various out-of-bounds accesses.

**GCC 5.1 mit Offloading und Cilk-Plus-Support erschienen**

## 1.12. Debug-Hilfspakete in OpenSuse

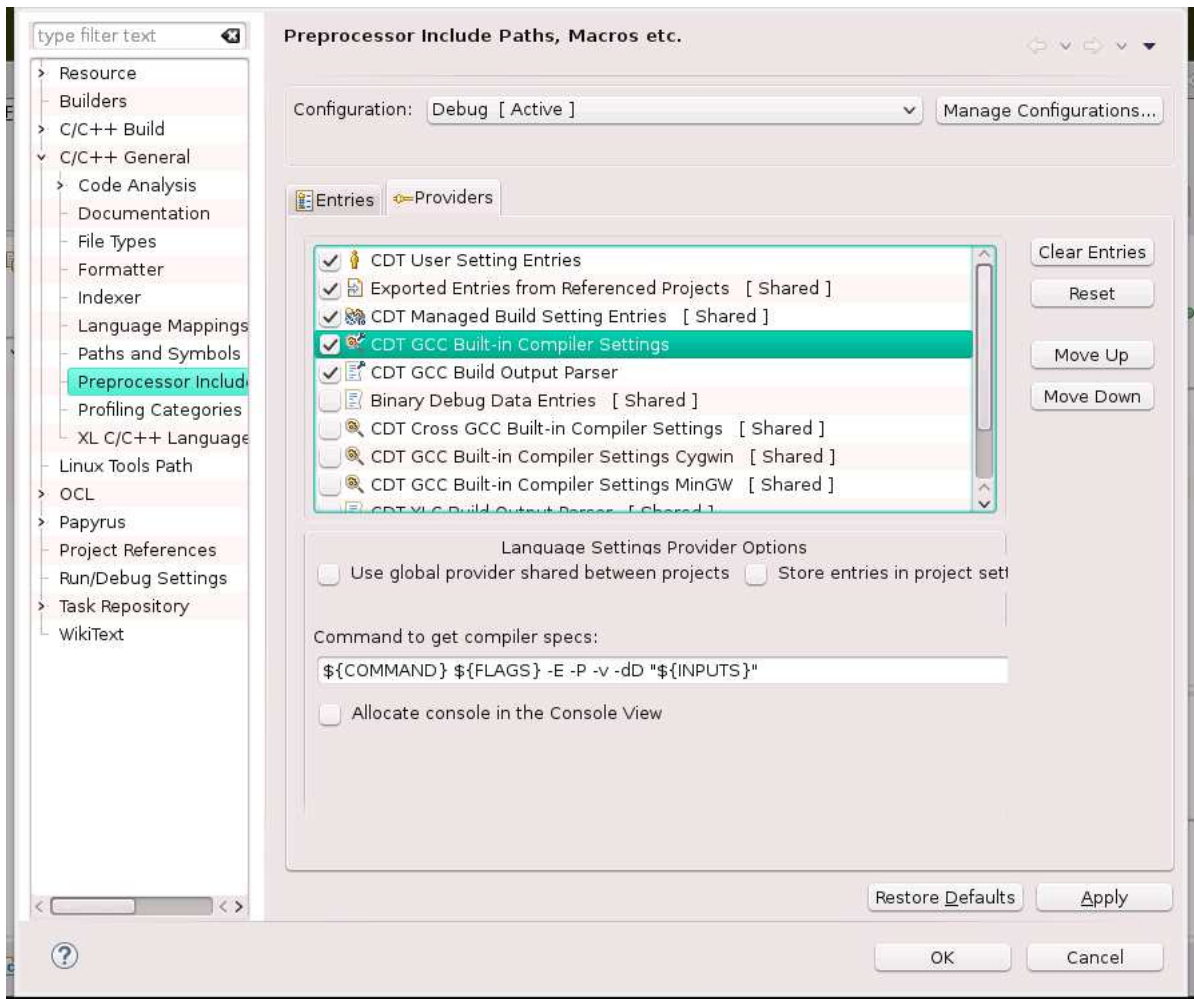
### 1.12.1. debuginfo, debugsource in OpenSUSE

▼	Paket	Zusammenfas	Installiert (V	Größe
<input checked="" type="checkbox"/>	gcc48-32bit	The GNU C ...	4.8.5-21.1	9,
<input checked="" type="checkbox"/>	gcc48-c++	Der GNU C...	4.8.5-21.1	14,
<input checked="" type="checkbox"/>	gcc48-gij	Java Byteco...	4.8.5-21.1	118
<input checked="" type="checkbox"/>	gcc48-info	Documentat...	4.8.5-21.1	2,
<input checked="" type="checkbox"/>	gcc48-java	Der GNU Ja...	4.8.5-21.1	14,
<input type="checkbox"/>	cross-aarch64-gcc48-icecream-backend	Der GNU C...	(4.8.5-21.2)	12,
<input type="checkbox"/>	cross-armv6hl-gcc48-icecream-backend	Der GNU C...	(4.8.5-21.2)	14,
<input type="checkbox"/>	cross-armv7hl-gcc48-icecream-backend	Der GNU C...	(4.8.5-21.2)	14,
<input type="checkbox"/>	cross-hppa-gcc48-icecream-backend	Der GNU C...	(4.8.5-21.2)	12,
<input type="checkbox"/>	cross-i386-gcc48-icecream-backend	Der GNU C...	(4.8.5-21.2)	13,
<input type="checkbox"/>	cross-ia64-gcc48-icecream-backend	Der GNU C...	(4.8.5-21.2)	12,
<input type="checkbox"/>	cross-ppc-gcc48-icecream-backend	Der GNU C...	(4.8.5-21.2)	14,
<input type="checkbox"/>	cross-ppc64-gcc48-icecream-backend	Der GNU C...	(4.8.5-21.2)	14,
<input type="checkbox"/>	cross-ppc64le-gcc48-icecream-backend	Der GNU C...	(4.8.5-21.2)	14,
<input type="checkbox"/>	cross-s390-gcc48-icecream-backend	Der GNU C...	(4.8.5-21.2)	12,
<input type="checkbox"/>	cross-s390x-gcc48-icecream-backend	Der GNU C...	(4.8.5-21.2)	12,
<input type="checkbox"/>	gcc48-ada	Auf GCC (G...	(4.8.5-21.1)	70,
<input type="checkbox"/>	gcc48-ada-32bit	Auf GCC (G...	(4.8.5-21.1)	31,
<input type="checkbox"/>	gcc48-ada-debuginfo	Debug infor...	(4.8.5-21.1)	143,
<input checked="" type="checkbox"/>	gcc48-c++-debuginfo	Debug infor...	(4.8.5-21.1)	75,
<input checked="" type="checkbox"/>	gcc48-debuginfo	Debug infor...	(4.8.5-21.1)	72,
<input checked="" type="checkbox"/>	gcc48-debugsource	Debug sour...	(4.8.5-21.1)	112,

▼ Paket	Zusammenfassung	Installiert (V)	Größe
<input checked="" type="checkbox"/> glibc	Die Standard Shared...	2.19-22.1	6,3 MiB
<input checked="" type="checkbox"/> glibc-32bit	Die Standard Shared...	2.19-22.1	3,4 MiB
<input checked="" type="checkbox"/> glibc-devel	Enthält Dateien und ...	2.19-22.1	3,0 MiB
<input checked="" type="checkbox"/> glibc-devel-32bit	Enthält Dateien und ...	2.19-22.1	304,3 KiB
<input checked="" type="checkbox"/> glibc-extra	Extra binaries from ...	2.19-22.1	23,8 KiB
<input checked="" type="checkbox"/> glibc-info	Info-Dateien zur GN...	2.19-22.1	910,3 KiB
<input checked="" type="checkbox"/> glibc-locale	Lokalisierungsdaten ...	2.19-22.1	114,2 MiB
<input checked="" type="checkbox"/> glibc-locale-32bit	Lokalisierungsdaten ...	2.19-22.1	6,1 MiB
<input checked="" type="checkbox"/> linux-glibc-devel	Linux headers for us...	4.1-1.1	4,3 MiB
<input checked="" type="checkbox"/> nss-mdns	Host Name Resoluti...	0.10-64.1	138,1 KiB
<input checked="" type="checkbox"/> nss-mdns-32bit	Host Name Resoluti...	0.10-64.1	63,7 KiB
<input checked="" type="checkbox"/> glibc-debuginfo	Debug information f...	(2.19-22.1)	21,9 MiB
<input checked="" type="checkbox"/> glibc-debuginfo-32bit	Debug information f...	(2.19-22.1)	11,6 MiB
<input checked="" type="checkbox"/> glibc-debugsource	Debug sources for p...	(2.19-22.1)	34,3 MiB
<input checked="" type="checkbox"/> glibc-devel-debuginfo	Debug information f...	(2.19-22.1)	262,9 KiB
<input checked="" type="checkbox"/> glibc-devel-debuginfo-32bit	Debug information f...	(2.19-22.1)	
<input type="checkbox"/> glibc-devel-static	C library static librar...	(2.19-22.1)	33,0 MiB
<input type="checkbox"/> glibc-devel-static-32bit	C library static librar...	(2.19-22.1)	20,0 MiB
<input checked="" type="checkbox"/> glibc-extra-debuginfo	Debug information f...	(2.19-22.1)	47,0 KiB
<input type="checkbox"/>			

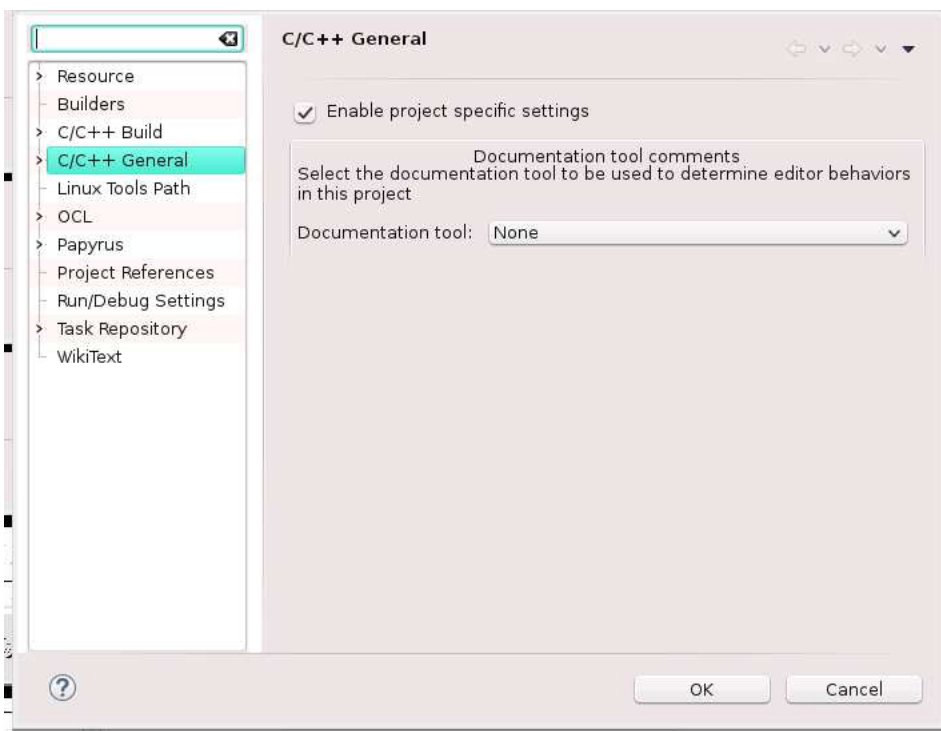
## 1.12.2. Umstellung auf projektspezifische Codechecks

Abstellen der shared-Einstellung:

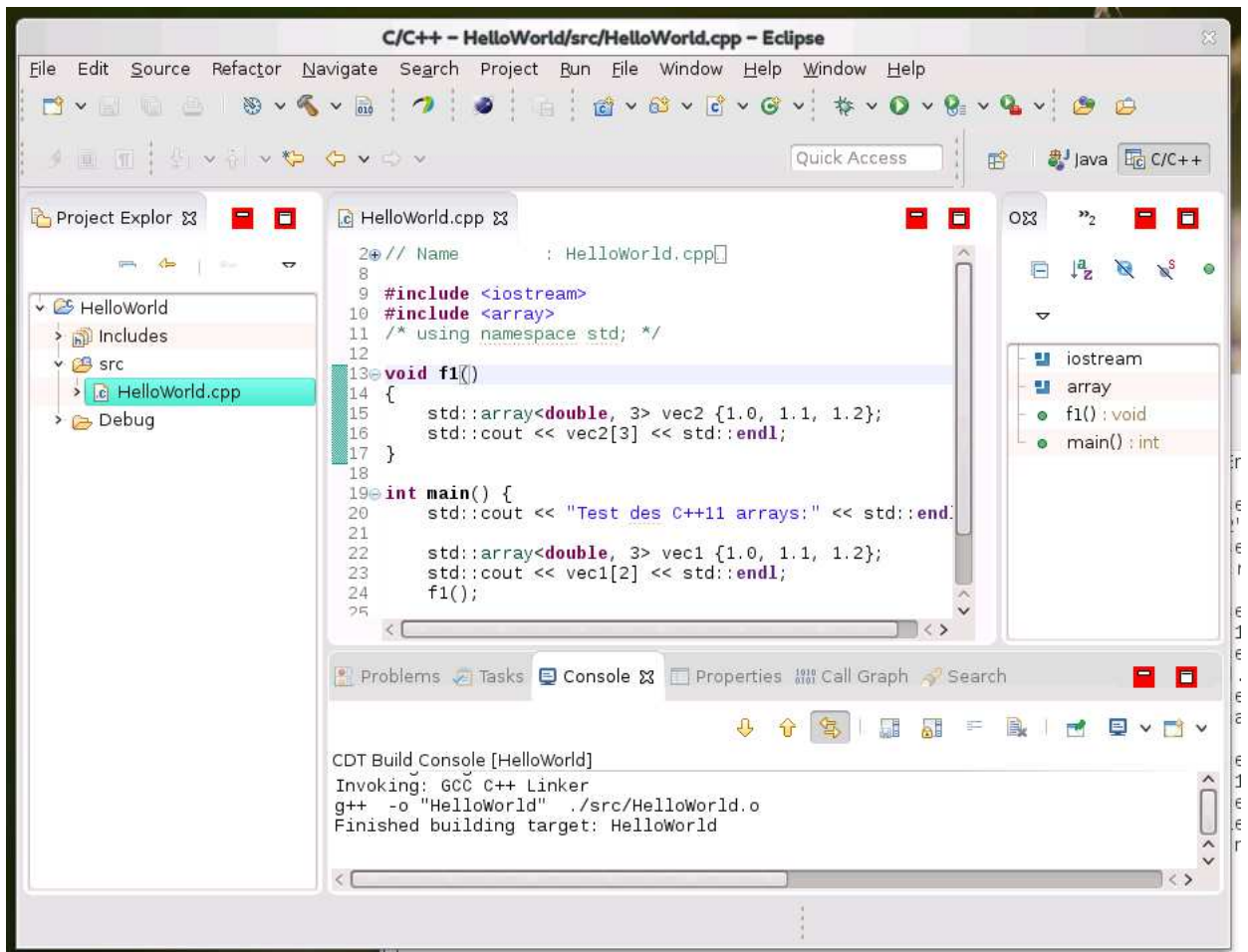




Einschalten der projektspezifischen Einstellungen:



... und auch der Code-Checker kennt den C++-Dialekt:



## 1.13. Modularisierung

### 1.13.1. Prinzipien der Moduarisierung

1. Module sollten **syntaktischen Einheiten** der Programmiersprache entsprechen.
2. Module sollten **mit möglichst wenigen anderen Modulen** „kommunizieren“.
3. „Kommunizierende“ Module sollten so **wenig** wie möglich **Informationen (Daten) austauschen**.
4. Jeder **Datenausch** zweier „kommunizierender“ Module muß **offensichtlich** in der Modulspezifikation (und nicht indirekt) kenntlich gemacht werden.
5. Alle **Daten** eines Moduls sollten **nur diesem bekannt** sein (außer im Falle einer gezielten Exportierung an möglichst wenige Nachbarmodule).
6. Ein Modul sollte **abgeschlossen und offen** sein.

### 1.13.2. Typen der Modularisierung

1. modulare **Zerlegbarkeit** (z.B. Top-Down-Design)
2. modulare **Zusammenfügbarkeit** (z.B. UNIX-Filter)
3. modulare **Verständlichkeit** (d.h. jede Modulbeschreibung selbsterklärend)
4. modulare „**Stetigkeit**“  
Kleine Spezifikationsänderungen wirken sich nur in **wenigen** Modulen aus. (Z.B. dyn. Felder, symbolische Konstanten, ...)
5. modularer „**Schutz**“  
Fehler/Ausnahmebedingungen bleiben in ihrer Auswirkung auf nur **wenige** Module beschränkt. (Z.B. direkte Konsistenzüberprüfung von Tastatureingaben, ...)

```

Module in C++:
// File_1.cpp:
export Lib: // Module definition header.
             // Must precede all declarations.
    import std;
public :
    namespace N {
        struct S {
            S() { std::cout << "S()\n"; }
        };
    }

// File_2.cpp:
import Lib;
int main() {
    N::S s;
}

```

<http://www.open-std.org/jtc1/sc22/wg21/docs/papers/2007/n2316.pdf> und

“Heading for a separate TR

These topics are deemed too important to wait for another standard after C++0x before being published, but too experimental to be finalised in time for the next Standard. Therefore, these features will be delivered by a technical report at the earliest opportunity.“

(Vgl. [Features originally planned but removed ...](#))

modules TS vs. `#include`

`constexpr if` (siehe auch Seite 7f.)

Java 9 Features with Examples

Java 9 Tutorial: Wir bauen einen Paketshop mit Jigsaw

9 new features in Java 9

5 Java-9-Features

C++?? Current Status

28.11.2016: Letzte Änderungen bei C++17 und erste Arbeiten an C++20

C++20

„leere Listen-Objekte vs. nicht initialisierten Listen-Objekten“

list initialization (since C++11)

„char\*“ vs. „const char\*“ in C++03/11

dynamic exception as a failed experiment

## 1.14. Natürlichsprachige Codeverträge (Kommentare und Verbalisierung)

iOS-11-Datumsfehler ... (wieder um immer wieder)  
iPhones stürzen ab

### 1.14.1. Fallstricke umgangssprachlicher Spezifikation

„Informelle Beschreibung“: Auf einem Parkplatz stehen PKW's und Motorräder. Zusammen seien es  $n$  Fahrzeuge mit insgesamt  $m$  Rädern. Bestimme die Anzahl  $P$  der PKW's.

(Überlegen Sie sich Vorbedingungen und Nachbedingungen der Softwarelösung dieses Problems.)

„Lösung“: Sei

$P$  := Anzahl der PKW's

$M$  := Anzahl der Motorräder

$$\left\{ \begin{array}{l} P + M = n \\ 4P + 2M = m \end{array} \right\} \Leftrightarrow \left\{ \begin{array}{l} M = n - P \\ P = \frac{m-2n}{2} \end{array} \right\} \Leftrightarrow \left\{ \begin{array}{l} M = \frac{4n-m}{2} \\ P = \frac{m-2n}{2} \end{array} \right\}$$

„Algorithmus“:

```
M = (4 * n - m) / 2;
```

```
P = (m - 2 * n) / 2;
```

```
std::cout << P << " PKWs und " << M << " Motorräder " << std::endl;
```

**Problem:** \*\*\*\*Null-Euro-Rechnung, Null-Euro-Mahnung,...

$$(m, n) = (9, 3) \Rightarrow P = 1\frac{1}{2}$$

$$(m, n) = (2, 5) \Rightarrow P = -4$$

Vor der Entwicklung eines Algorithmus ist zunächst für das Problem eine *Spezifikation* bestehend aus

1. Definitionsbereich der Eingabegrößen,
2. Wertebereich der (Hilfs-)Variablen und Ausgabegrößen sowie
3. für die Lösung wichtigen (meist booleschwertige) Eigenschaften (insbesondere funktionaler Zusammenhang zwischen Eingabe- und Ausgabegrößen, Konsistenzbeziehungen, ...)

anzufertigen.

Besser ist also:

**Eingabe:**  $m \in \{0, 1, \dots, \text{numeric\_limits}\langle\text{int}\rangle::\text{max}()\}$ ,  
 $n \in \{0, 1, \dots, \text{numeric\_limits}\langle\text{int}\rangle::\text{max}() / 2\}$

oder sogar  $n \in \{0, 1, \dots, \text{numeric\_limits}\langle\text{int}\rangle::\text{max}() / 4\}$ ,  
will man M wirklich mittels „ $M = (4 * n - m) / 2$ ;“ statt dem weniger häufig zu nicht  
erkannten Overflows föhrenden „ $M = (2 * n - m / 2)$ ;“ in der Implementierung be-  
rechnen.

**Vorbedingungen:**  $m$  gerade,  $2n \leq m \leq 4n$

**Ausgabe:**  $P \in \{0, 1, \dots, \text{numeric\_limits}\langle\text{int}\rangle::\text{max}()\}$ , falls die Nachbedingung  
erfüllt ist (sonst „keine Lösung“)

**Nachbedingung:** Ein  $(P, M) \in \{0, 1, \dots, \text{numeric\_limits}\langle\text{int}\rangle::\text{max}()\}^2$  mit

$$\begin{aligned}P + M &= n \\4P + 2M &= m\end{aligned}$$

Problems with natural language

Can you write an unambiguous specification in a natural language like English?

Is Software Design is a Trial and Error Approach

## 1.15. Codeverträge

### 1.15.1. REQUIRE(), ENSURE(), ID() und invariant()

- genaue Spezifikation der Methoden des Moduls:
  - **Vorbedingungen** (preconditions) einer Methode sind Bedingungen, die vor dem Aufruf einer Methode erfüllt sein müssen, damit sie ausführbar ist. Vorbedingungen sind boolsche Ausdrücke über den Abfragen des Moduls und den Parametern der Methode.
  - **Nachbedingungen** (postconditions) einer Methode sind Bedingungen, die nach dem Aufruf einer Methode erfüllt sind; sie beschreiben, welches Ergebnis ein Methodenaufruf liefert oder welchen Effekt er erzielt. Nachbedingungen sind boolsche Ausdrücke über den Abfragen des Moduls und den Parametern der Methode, erweitert um ein Gedächtniskonstrukt, das die Werte von Ausdrücken vor dem Methodenaufruf liefert. Im Einzelnen:
    - \* Spezifikation des Funktionsergebnisses
    - \* genaue Spezifikation der Werte der Referenz- und der dereferenzierten Pointer-Parameter nach Beendigung der Methode
    - \* Spezifikation der Werte aller Attribute des Moduls nach Beendigung der Methode (häufig werden hier nicht einzeln genannte Attribute als nicht verändert angenommen)
- Definition der erlaubten Stati (Werte aller Attribute) des Moduls zu jedem (beobachtbaren) Zeitpunkt zur Laufzeit des Moduls.

Sie werden durch Invarianten beschrieben. **Invarianten** eines Moduls sind allgemeine unveränderliche Konsistenzbedingungen an den Zustand des Moduls, die vor und nach dem Aufruf jeder (öffentlichen) Methode gelten. Formal sind Invarianten boolsche Ausdrücke über den Abfragen des Moduls; inhaltlich können sie z.B. Geschäftsregeln (business rules) ausdrücken.

(vergleiche: <http://informatik.karlheinz-hug.de/artikel/ForumWI01%20SdV.pdf>)

### Ein Beispiel in C++ mit Hilfe von Nana:

```
#define EIFFEL_CHECK CHECK_ALL
#include <set>
#include <vector>
#include <eiffel.h>
#include <nana.h>
...
void quicksort(double v[], int l, int h)
{
    REQUIRE(l <= h+1);
    ...
    ENSURE(A(int k=l, k<h, k++, v[k]<=v[k+1]));
}
void quicksort(double v[], int n)
{
    REQUIRE(n>=1);
    ID(multiset<double> v_old_contents(&v[0],&v[n]));
    ...
    ENSURE(A(int k=0, k<n-1, k++, v[k]<=v[k+1]));
    ID(multiset<double> v_contents(&v[0],&v[n]));
    ENSURE(v_old_contents == v_contents);
};

class name_list{ ...
void name_list::put(const string& a_name)    // Push a_name
    into list
DO
    REQUIRE(/* name not in list */ !has(a_name));
    ID(set<string> contents_old(begin(),end()));
    ID(int count_old = get_count());
    ID(bool not_in_list = !has(a_name));
    ...
    ENSURE(has(a_name));
    ENSURE( (!not_in_list) || (get_count() == count_old + 1));
    ID(set<string> contents(begin(),end()));
    ENSURE( (!not_in_list) || (contents == contents_old + a_name)
    );
END;
...
}
```



Auswahl der Überprüfungslevel durch:

```
#define EIFFEL_DOEND
#ifndef EIFFEL_CHECK
#define EIFFEL_CHECK CHECK_ALL
//
//          CHECK_LOOP          Makros CHECK() und folgende
//          CHECK_INVARIANT     Makros INVARIANT() und folgende
//          CHECK_ENSURE        Methode invariant() und folgende
//          CHECK_REQUIRE       Nachbedingungen und folgende
//          CHECK_NO            Vorgedingungen
#endif
#include "eiffel.h"
#include "nana.h"
```

Zum [nana-Manual](#)

Contracts programming for C++20 — Current proposal status:

Siehe Seite 6 unten.

## 1.15.2. Klassifikation der Klassenmethoden gemäß SdV

- const-Methoden (Abfragen/Queries/Observatoren/Getter) teilt man in wesentliche und abgeleitete solche ein.
- Die wesentlichen Observatoren erlauben eine vollständige Spezifizierung des Zustands eines Klassenexemplars.
- Sie (und nur sie) werden nicht durch Nachbedingungen spezifiziert. Sie dienen vielmehr dazu, abgeleitete Observatoren und Modifikatoren (das sind nicht-const-Methoden) in ihren Nachbedingungen näher zu bestimmen.
- Dazu werden die abgeleiteten Observatoren durch eine Nachbedingung unter Benutzung einer oder mehrerer wesentlicher Observatoren spezifiziert.
- Modifikatoren werden durch eine Nachbedingung unter Benutzung aller wesentlicher Observatoren spezifiziert, um den exakten Zustand des Exemplars am Ende des Modifikatoraufrufs anzugeben.
- Verzichte (evtl.) in Nachbedingungen von Modifikatoren darauf, explizit zu spezifizieren, was sich nicht ändert (in der Annahme, dass alles nicht explizit genannte als *ungeändert* zu gelten hat). Leider ist nicht immer klar, was *ungeändert* zu bedeuten hat: Mindestens dann sollten Frameregeln (Rahmenbedingungen) explizit spezifizieren, was nach Aufruf des Modifikators *gleich* ist wie vorher.
- Explizite Spezifikation aller Rahmenbedingungen können bei programminterner Überprüfung der Nachbedingungen fehlerhafte Implementierungen aufdecken!
- Schreibe für jede Methode eine Vorbedingung mit Hilfe von
  - Abfragen und
  - Bedingungen an Methodenparameter.

Hier (bei den Vorbedingungen) dürfen auch abgeleitete Abfragen, die eventuell effizienter sein können als eine sonst nötige Kombination mehrerer wesentlicher Abfragen, benutzt werden.

- Sorge dafür, dass bei Erfülltsein der Vorbedingungen auf jeden Fall die Nachbedingungen ebenfalls erfüllt sind (oder — in Ausnahmefällen — eine Exception ausgelöst wird).
- Sorge dafür, dass die Abfragen in Vorbedingungen effizient berechnet werden (evtl. durch Hinzufügen weiterer effizienter abgeleiteter Abfragen). Vergesse nicht, die evtl. hinzugefügten neuen abgeleiteten Abfragen durch Nachbedingungen (und Vorbedingungen) zu spezifizieren.
- Nutze Invarianten um die Abhängigkeit von Abfragen zu spezifizieren (Konsistenzbeziehungen).

- Untersuche alle Abfragen paarweise auf Redundanzen und formuliere solche explizit als Invarianten.
- Wann immer Abfrage-Ergebnisse oder Methoden-Parameter eingeschränkte Wertebereiche besitzen, formuliere dies explizit in Form von
  - Vorbedingungen,
  - Nachbedingungenoder
  - Invarianten.
- Schreibe die Nachbedingungen von virtuellen (also überschreibbaren) Methoden immer in der Form  
Vorbedingung implies Nachbedingung  
(`Ensure(!Vorbedingung) || Nachbedingung`), um die Redefinition in Kindklassen konfliktfrei zu ermöglichen.

## 1.16. Der Sicherheits-Supergau Spectre/Meltdown

Meltdown and Spectre — Vulnerabilities in modern computers leak passwords and sensitive data  
Meltdown and Spectre: what we know about the vulnerabilities in CPUs?

- SpectreAttack, 3.1.2018, CVE-2017-5715, 5715/2017  
in OpenSuse für AMD behoben, 4.1.2018
- MeltdownAttack, 3.1.2018, CVE-2017-5754, 5754/2017  
in OpenSuse "Kernel Address Isolation" behoben, 4.1.2018
- Windows 10 Patch KB4056892, 7.1.2018  
Windows-Prüfwerkzeug, 1/2018  
Microsoft ...: Details zu Patches und Leistungseinbußen, 9.1.2018
- iOS 11.2.2, 8.1.2018
- Prozessorlücke: Auch Qualcomm-CPUs sind anfällig, 6.1.2018
- Die Sicherheitshinweise und Updates von HW- und SW-Herstellern, 8.1.2018
- Meltdown und Spectre sind ein Security-Supergau, 5.1.2018
- "Meltdown" und "Spectre": Updaten, updaten, updaten, 5.1.2018
- Bug: Browser-Hersteller reagieren auf Meltdown und Spectre, 4.1.2018
- Intel-Chef stieß hunderttausende Aktien ab, Börsenkurs sackt ab, 4.1.2018
- US-Senatoren fordern Prüfung des Aktienverkaufs von Intel-Chef, 10.1.2018
- Intel patcht ab 2013 produzierte Prozessoren, bestätigt Performance-Auswirkung, 9.1.2018
- Weißes Haus: NSA wusste nichts von Computerchip-Schwachstelle, 6.1.2018
- De Maizière will eigene Schlüsseltechnologien stärken, 5.1.2018
- Erste Klagen gegen Intel, Performanceprobleme kochen hoch, 5.1.2018
- FAQ Meltdown/Spectre
- ...

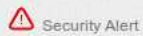
# 5

## Patches Available All SUSE Products

Updates for your SUSE products are now available. We recommend that you regularly install updates in order to keep your products running at peak performance, remove known security vulnerabilities and software defects, improve stability and add the latest functionality.

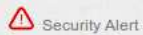
In order to access these updates, we recommend you regularly synchronize your systems using SUSE Manager, or the Subscription Management Tool (SMT), our recommended systems management solutions. If needed, you can download patches by clicking the links below.

Please note: You must have registered your SUSE product in order to access and download patches.



### **kvm 13397**

SUSE Linux Enterprise Point of Service 11 SP3 for x86  
SUSE Linux Enterprise Server 11 SP3 LTSS for x86-64  
SUSE Linux Enterprise Server 11 SP3 LTSS for x86  
SUSE Linux Enterprise Server 11 SP3 LTSS for s390x



### **kernel-firmware 30**

SUSE Linux Enterprise Server 12 SP1 LTSS for x86-64  
SUSE Linux Enterprise Server for SAP Applications 12 SP1 for x86-64  
SUSE Linux Enterprise Server for SAP Applications 12 SP1 for ppc64le  
SUSE OpenStack Cloud 6 for x86-64

## Sicherheit

- Security update for ucode-intel
- Security update for the Linux Kernel
- Security update for kernel-firmware
- Security update for gdk-pixbuf
- Security update for evince
- Security update for Mozilla Thunderbird
- Security update for ImageMagick
- Security update for ImageMagick

## empfohlen

Patch-Kategorie anzeigen:

Patch-Beschreibung

other secrets.

This problem is mitigated by adding speculative fencing on affected code paths throughout the Linux kernel.

- CVE-2017-5715 / "SpectreAttack": Local attackers on systems with modern CPUs featuring branch prediction could use mispredicted branches to speculatively execute code patterns that in turn could be made to leak other non-readable content in the same address space, an attack similar to CVE-2017-5753.

This problem is mitigated by disabling predictive branches, depending on CPU architecture either by firmware updates and/or fixes in the user-kernel privilege boundaries.

Please also check with your CPU / Hardware vendor on updated firmware or BIOS images regarding this issue.

As this feature can have a performance impact, it can be disabled using the "nospec" kernel commandline option.

- CVE-2017-5754 / "MeltdownAttack": Local attackers on systems with modern CPUs featuring deep instruction pipelining could use code patterns in userspace to

## 1.17. Wiederverwendbarkeit in höheren Programmiersprachen

Vermeide es, das Rad immer wieder neu zu erfinden!

1. **Algorithmen (Programme)** lösen i. allg. eine Klasse von Problemen, die durch Eingabewerte parametrisiert sind.  
[Wiktionary: Algorithmus](#)  
[Eigenschaften eines Algorithmus](#)  
[nondeterminism in Prolog](#)  
[Nondeterministic algorithm](#)
  
2. **Constraints:** Unterbereichstypen, ...  
[Constraints in Programmiersprachen](#)  
[Why don't Java, C# and C++ have ranges?](#)  
[MODULA-2 subrange types](#)  
[Pascal/ADA-Subrange Types](#)  
[A Subrange type for C++](#)
  
3. **Unterprogramme** (Funktionen, Prozeduren, Operatoren) lösen eine Klasse von Problemen: Gemäß dem Prinzip der methodischen Restriktion sind dabei die einzelnen Parameter jeweils Werte des Wertebereiches eines festen Typs.  
[Prototype](#)  
[Deklaration](#)  
[C functions without prototypes](#)  
[C and defining a function prototype with no parameters](#)  
[declaring functions](#)  
[Descriptions of function semantics \(Page 420\)](#)  
[Problems with natural language](#)  
[exceptions specification](#)  
[exception class](#)  
[C++ std exception hierarchy](#)  
[Features removed or deprecated in C++11](#)  
[undefined/unspecified behavior \(3.4.3f.\) of C11](#)  
[Translation limits \(5.2.4.1\) of C11](#)  
[implementation limits \(C++11\)](#)  
[Limitations of Java language](#)

attributes, lambda expressions, =deleted, =default, overrides, final, type inference (auto), .

4. **Unterprogramme mit konformen Feldparametern** (in Pascal bzw. open-array-Parameter in Modula2) erlauben es Parametern, einer Klasse von Feldern anzugehören (variable Dimension);

```
PROCEDURE EuklNorm (v:ARRAY OF REAL): REAL;
```

5. **Dynamische Felder / Teilfeld-Selektoren/Array slicing** erlauben einen in der Dimension noch nicht festgelegten Feldtyp bzw. Projektion auf ein Teilfeld:

```
TYPE vector = ARRAY[*] OF REAL;  
a := t[*..2];  
... t[min..k] ...
```

6. **Polymorphismus (virtual functions)** und **Überladen von Funktionen**:

```
writeln(x : real);           k := i * j;  
writeln(i : integer);       z := x * y;  
...
```

Siehe [Default parameters](#),

[Function overloading](#), [Explicit overrides and final](#),  
[Explicitly defaulted and deleted special member functions](#),  
[vararg parameter lists](#) und [type inference](#).

7. **Unterprogramme als Parameter** anderer Unterprogramme erlauben Algorithmen für eine Klasse von Unterprogrammen gleicher Signatur:

```
function Bisection (function f(x : real) : real;  
                   xLeft, xRight      : real;  
                   success             : boolean  
                   ) : real;
```



8. **Generizität** (Typen als Parameter) ermöglicht Parametrisierung nach Typen:

```
generic
  type T is private;
procedure swap (x, y : in out T) is
  t : T
begin
  t := x; x := y; y := t
end swap
  :
procedure int_swap is new swap (INTEGER);
procedure real_swap is new swap (REAL);
```

Ada generic procedure swap, see page 86

Ada generic model

templates with "... parameters

**Eingeschränkte Generizität** schränkt die aktuellen Typ-Parameter ein:

```
generic
  type T is private;
  with function "<=>" (a, b : T) return BOOLEAN is <>;
function minimum (x, y : T) return T is
begin
  if x <= y then return x;
  else return y
  end if
end minimum
```

(Ähnliches kann durch Textprozessoren oder die typunsichere Benutzung des typungebundenen Zeigers ADDRESS erreicht werden oder in Java mittels **constrained generic classes**:

```
class ListObject <T extends Comparable<T>>{ ... }
```

Difference between a Java interface and a Java abstract class)

Generics in Java

Java und C++

Irreführende unspezifische Fehlermeldungen bei der Nutzung von (uneingeschränkten) C++-Templates:

```

In file included from /usr/include/c++/4.5/algorithm:63:0,
      from bad_error_eg.cpp:3:
/usr/include/c++/4.5/bits/stl_algo.h: In Funktion >>void std::__insertion_sort(RandomAccessIterator, RandomAccessIterator) [with RandomAccessIterator = _
/usr/include/c++/4.5/bits/stl_algo.h:3358:4: instantiated from >>void std::__inplace_stable_sort(RandomAccessIterator, RandomAccessIterator) [with Rand
/usr/include/c++/4.5/bits/stl_algo.h:5415:2: instantiated from >>void std::stable_sort(_RAIter, _RAIter) [with _RAIter = __gnu_cxx::__normal_iterator<std:
bad_error_eg.cpp:8:41: instantiated from here
/usr/include/c++/4.5/bits/stl_algo.h:2103:4: Fehler: no match for >>operator<< in >>__i__gnu_cxx::__normal_iterator<Iterator, Container>::operator* [wit
/usr/include/c++/4.5/bits/stl_algo.h: In Funktion >>void std::__merge_without_buffer(BidirectionalIterator, BidirectionalIterator, BidirectionalIterator,
/usr/include/c++/4.5/bits/stl_algo.h:3364:7: instantiated from >>void std::__inplace_stable_sort(RandomAccessIterator, RandomAccessIterator) [with Rand
/usr/include/c++/4.5/bits/stl_algo.h:5415:2: instantiated from >>void std::stable_sort(_RAIter, _RAIter) [with _RAIter = __gnu_cxx::__normal_iterator<std:
bad_error_eg.cpp:8:41: instantiated from here
/usr/include/c++/4.5/bits/stl_algo.h:2963:4: Fehler: no match for >>operator<< in >>__middle__gnu_cxx::__normal_iterator<Iterator, Container>::operator*
/usr/include/c++/4.5/bits/stl_algo.h: In Funktion >>void std::__unguarded_linear_insert(RandomAccessIterator) [with RandomAccessIterator = __gnu_cxx::__no
/usr/include/c++/4.5/bits/stl_algo.h:2111:6: instantiated from >>void std::__insertion_sort(RandomAccessIterator, RandomAccessIterator) [with RandomAcc
/usr/include/c++/4.5/bits/stl_algo.h:3358:4: instantiated from >>void std::__inplace_stable_sort(RandomAccessIterator, RandomAccessIterator) [with Rand
/usr/include/c++/4.5/bits/stl_algo.h:5415:2: instantiated from >>void std::stable_sort(_RAIter, _RAIter) [with _RAIter = __gnu_cxx::__normal_iterator<std:
bad_error_eg.cpp:8:41: instantiated from here
/usr/include/c++/4.5/bits/stl_algo.h:2064:7: Fehler: no match for >>operator<< in >>__val < __next__gnu_cxx::__normal_iterator<Iterator, Container>::ope
In file included from /usr/include/c++/4.5/vector:61:0,
      from bad_error_eg.cpp:1:
/usr/include/c++/4.5/bits/stl_algobase.h: In Funktion >>ForwardIterator std::lower_bound(ForwardIterator, ForwardIterator, const Tp&) [with ForwardIter
/usr/include/c++/4.5/bits/stl_algo.h:2975:4: instantiated from >>void std::__merge_without_buffer(BidirectionalIterator, BidirectionalIterator, Bidirec
/usr/include/c++/4.5/bits/stl_algo.h:3364:7: instantiated from >>void std::__inplace_stable_sort(RandomAccessIterator, RandomAccessIterator) [with Rand
/usr/include/c++/4.5/bits/stl_algo.h:5415:2: instantiated from >>void std::stable_sort(_RAIter, _RAIter) [with _RAIter = __gnu_cxx::__normal_iterator<std:
bad_error_eg.cpp:8:41: instantiated from here
/usr/include/c++/4.5/bits/stl_algobase.h:976:4: Fehler: no match for >>operator<< in >>__middle__gnu_cxx::__normal_iterator<Iterator, Container>::operator
In file included from /usr/include/c++/4.5/algorithm:63:0,
      from bad_error_eg.cpp:3:
/usr/include/c++/4.5/bits/stl_algo.h: In Funktion >>Filter std::upper_bound(Filter, Filter, const Tp&) [with Filter = __gnu_cxx::__normal_iterator<std::com
/usr/include/c++/4.5/bits/stl_algo.h:2982:4: instantiated from >>void std::__merge_without_buffer(BidirectionalIterator, BidirectionalIterator, Bidirec
/usr/include/c++/4.5/bits/stl_algo.h:3364:7: instantiated from >>void std::__inplace_stable_sort(RandomAccessIterator, RandomAccessIterator) [with Rand
/usr/include/c++/4.5/bits/stl_algo.h:5415:2: instantiated from >>void std::stable_sort(_RAIter, _RAIter) [with _RAIter = __gnu_cxx::__normal_iterator<std:
bad_error_eg.cpp:8:41: instantiated from here
/usr/include/c++/4.5/bits/stl_algo.h:2461:4: Fehler: no match for >>operator<< in >>__val < __middle__gnu_cxx::__normal_iterator<Iterator, Container>::o
/usr/include/c++/4.5/bits/stl_algo.h: In Funktion >>OIter std::merge(_IIter1, _IIter1, _IIter2, _IIter2, _OIter) [with _IIter1 = std::complex<float>*, _IIT
/usr/include/c++/4.5/bits/stl_algo.h:2838:4: instantiated from >>void std::__merge_adaptive(BidirectionalIterator, BidirectionalIterator, Bidirectional
/usr/include/c++/4.5/bits/stl_algo.h:3315:7: instantiated from >>void std::__stable_sort_adaptive(RandomAccessIterator, RandomAccessIterator, Pointer,
/usr/include/c++/4.5/bits/stl_algo.h:5417:2: instantiated from >>void std::stable_sort(_RAIter, _RAIter) [with _RAIter = __gnu_cxx::__normal_iterator<std:
bad_error_eg.cpp:8:41: instantiated from here
/usr/include/c++/4.5/bits/stl_algo.h:5299:4: Fehler: no match for >>operator<< in >>__first2__gnu_cxx::__normal_iterator<Iterator, Container>::operator*
/usr/include/c++/4.5/bits/stl_algo.h: In Funktion >>BidirectionalIterator3 std::__merge_backward(BidirectionalIterator1, BidirectionalIterator1, Bidirec
/usr/include/c++/4.5/bits/stl_algo.h:2847:4: instantiated from >>void std::__merge_adaptive(BidirectionalIterator, BidirectionalIterator, Bidirectional
/usr/include/c++/4.5/bits/stl_algo.h:3315:7: instantiated from >>void std::__stable_sort_adaptive(RandomAccessIterator, RandomAccessIterator, Pointer,
/usr/include/c++/4.5/bits/stl_algo.h:5417:2: instantiated from >>void std::stable_sort(_RAIter, _RAIter) [with _RAIter = __gnu_cxx::__normal_iterator<std:
bad_error_eg.cpp:8:41: instantiated from here
/usr/include/c++/4.5/bits/stl_algo.h:2740:4: Fehler: no match for >>operator<< in >>* __last2 < __last1__gnu_cxx::__normal_iterator<Iterator, Container>
/usr/include/c++/4.5/bits/stl_algo.h: In Funktion >>OIter std::merge(_IIter1, _IIter1, _IIter2, _IIter2, _OIter) [with _IIter1 = __gnu_cxx::__normal_iterat
/usr/include/c++/4.5/bits/stl_algo.h:3163:4: instantiated from >>void std::__merge_sort_loop(RandomAccessIterator1, RandomAccessIterator1, RandomAccess
/usr/include/c++/4.5/bits/stl_algo.h:3261:4: instantiated from >>void std::__merge_sort_with_buffer(RandomAccessIterator, RandomAccessIterator, Pointer
/usr/include/c++/4.5/bits/stl_algo.h:3312:4: instantiated from >>void std::__stable_sort_adaptive(RandomAccessIterator, RandomAccessIterator, Pointer,
/usr/include/c++/4.5/bits/stl_algo.h:5417:2: instantiated from >>void std::stable_sort(_RAIter, _RAIter) [with _RAIter = __gnu_cxx::__normal_iterator<std:
bad_error_eg.cpp:8:41: instantiated from here
/usr/include/c++/4.5/bits/stl_algo.h:5299:4: Fehler: no match for >>operator<< in >>__first2__gnu_cxx::__normal_iterator<Iterator, Container>::operator*
/usr/include/c++/4.5/bits/stl_algo.h: In Funktion >>OIter std::merge(_IIter1, _IIter1, _IIter2, _IIter2, _OIter) [with _IIter1 = std::complex<float>*, _IIT
/usr/include/c++/4.5/bits/stl_algo.h:3163:4: instantiated from >>void std::__merge_sort_loop(RandomAccessIterator1, RandomAccessIterator1, RandomAccess
/usr/include/c++/4.5/bits/stl_algo.h:3263:4: instantiated from >>void std::__merge_sort_with_buffer(RandomAccessIterator, RandomAccessIterator, Pointer
/usr/include/c++/4.5/bits/stl_algo.h:3312:4: instantiated from >>void std::__stable_sort_adaptive(RandomAccessIterator, RandomAccessIterator, Pointer,
/usr/include/c++/4.5/bits/stl_algo.h:5417:2: instantiated from >>void std::stable_sort(_RAIter, _RAIter) [with _RAIter = __gnu_cxx::__normal_iterator<std:
bad_error_eg.cpp:8:41: instantiated from here
/usr/include/c++/4.5/bits/stl_algo.h:5299:4: Fehler: no match for >>operator<< in >>* __first2 < * __first1<

```

zum Beispiel in

```
#include <vector>
#include <complex>
#include <algorithm>

int main()
{
    std::vector<std::complex<float>> v;
    std::stable_sort(v.begin(), v.end());
}
```

obwohl der Algorithmus `stable_sort()` schon unzählige Male zuvor problemlos benutzt wurde.

Natürlich sollte `stable_sort()` nur benutzt werden, wenn der `value_type` des zu sortierenden Containers einen operator `<=` besitzt, aber wie kann man das in der C++-Quelle der STL maschinell überprüfbar codieren?

Ansätze zur eingeschränkten Generizität in C++ (2017?):

[Concepts-Lite](#)

[Concepts Lite: Constraining Template Arguments with Predicates](#)

[Concepts Lite: Constraining Templates with Predicates](#)

[A Concept Design for the STL](#)

```
template<Sortable Cont>
void sort(Cont& container);
```

```
list<int> lst = ...;
sort(lst); // Error
```

```
error: no matching function for call to 'sort(list<int>&)'
    sort(l);
      ^
```

```
note: candidate is:
```

```
note: template<Sortable T> void sort(T)
    void sort(T t) { }
```

```
note: template constraints not satisfied because
```

```
note:   'T' is not a/an 'Sortable' type [with T = list<int>] since
```

```
note:     'declval<T>()[n]' is not valid syntax
```

```

template<typename T>
constexpr bool Sortable()
{
    return ...;    // Returns true when T is a
                  // permutable container whose
                  // elements can be totally ordered
}

```

(aus: [Concepts Lite, Constraining Template Arguments with Predicates](#))

oder genauer (aus [A Concept Design for the STL, Seite 67](#), wenn concepts endlich existieren):

```

concept Sortable<ForwardIterator I> =
    TotallyOrdered<ValueType<I>> && Permutable<I>;

```

### 9. Patternmatching:

[Structured binding](#) (siehe auch Beispiel auf Seite 15 von

[H.-J. Buhl: Materialsammlung generische Programmierung](#))

Pattern matching in C++20?

### Hinweise:

First-Class-Objekt

First-class citizen

about first, second and third class ...

## 1.18. Erste einfache Code-Contracts in Eiffel: Vorbedingungen und Klassen-Invarianten

```
class
    WUERFEL

inherit
    DOUBLEMATH

create
    make

feature — Access

    seite : REAL_64

    make( s : REAL_64 ) is

        require
            argument_nonnegative: s >= 0.0
        do
            seite := s
        end

feature — Status report

    Oberflaeche : REAL_64 is
        — berechnet die Oberflaeche
        do
            Result := (6 * seite ^ 2)
        end

    Volumen : REAL_64 is
        — berechnet das Volumen
        do
            Result := seite ^ 3
        end

invariant
    seite_nonnegative: seite >= 0.0

end
```

## 1.19. Vererbung und Codeverträge (Fortsetzung von 1.15)

„A derived class ... should not strengthen any precondition or weaken any postcondition of a member function it overrides.“  
(aus: [Derived classes](#), [Public inheritance](#))

- a) Vorbedingungen können in einer Kindklasse abgeschwächt werden.
- b) Nachbedingungen in einer Kindklasse müssen stärker (oder gleich) sein als diejenigen der Elterklasse.
- c) Invarianten in der Kindklasse müssen ebenfalls stärker (oder gleich) als in der Elterklasse sein.

Dann ist ein echtes *Subcontracting* realisiert.

Bemerkung: Es reicht die Kindnachbedingung im Falle des Eintreffens der Eltervorbedingung stärker als die Elternachbedingung zu realisieren. Im Falle „Kindvorbedingung **and not** Eltervorbedingung“ darf die Kindnachbedingung frei gewählt werden.

Das ist unbedingt zu beachten, will man in Nachkommen eine benutzerfreundlichere Methodenvariante mit allgemeineren Vorbedingungen realisieren.

**tem and control the use of inheritance by Eiffel's assertion redefinition rule. This rule is intended to ensure that child classes have stronger class invariants, weaker feature preconditions and stronger feature postconditions. It can be**

[Eiffel Contracts and Inheritance](#)

[Eiffel Inheritance: require else, ensure then, invaraiant](#)

## 1.20. Integrierte Entwicklungsumgebungen zur Codequalitätssteigerung (am Beispiel: PyDev), IDEs zur SQA

WiederverwendbareSoftware-Teil2.pdf

*Qualitätssteigernde Eigenschaften der Nutzung des PyDev-Plugins für die Python-Programmentwicklung:*

- Anzeige suspekter Codestellen in der ersten Spalte des Python-Quellcodeeditor-Fensters, statische Codeanalyse
- Markierung der Sprachkonstrukte mit suspekten Codestellen (durch Unterschlängeln)
- In den Code integrierte Dokumentationsspezialkommentare zur automatischen Synchronisation geänderten Codes mit der automatisch generierten Dokumentation
- im Python-Texteditorfenster automatisch oder nach Tastendruck aufklappende Fenster, die die automatisch aus den Dokumentationsspezialkommentaren erzeugten Dokumentation an der aktuellen Editorstelle einblenden (**hover**)
- Pro Klasse ein Testrahmenprogramm, das durch Betätigung des Eclipse-Run-Knopfes gestartet wird
- Codeabdeckungschecks
- zeilenweises Debuggen fehlerhaften Codes
- interaktive Komponententest-Klassenanwendungsbeispiele (mit Spezifikation der zu erwartenden Ergebnisse) im Klassen-Kommentar, die im Testrahmenprogramm automatisch (und regressiv) ausgeführt werden
- Codeverträge (`contract.checkmod()`) mit Invarianten, Vorbedingungen und Nachbedingungen = bei jedem Produktionslauf mitlaufende Komponententests mit (in Form von Nachbedingungen) eingebauten Testorakeln, dynamische Codeanalyse, aggressive Asserts
- ...

## 1.21. Softwarequalitäts-Mängel

Smartphone-Apps zur Steuerung von Industrie-Anlagen sind voller Sicherheitslücken

bessere Kontrolle bei Künstlicher Intelligenz

KI und ML

Mehr Transparenz bei Algorithmen

Weniger schlecht Programmieren: Programmierstil

## 1.22. Ein erstes C++-Projekt mit Umbrello und Doxygen

UML 2 Class Diagram

UML 2 für Studenten

Getting Started with UML2

Umbrello

UML2 Class Diagram

Umbrello Handbuch

(alternativ: eclipse modelling mit papyrus oder VisualParadigm oder Together oder ...)

Übertragung des Würfelbeispiels in ein C++-Projekt:

<b>Wuerfel</b>
# Seite : double
+ Wuerfel(mySeite : double = 1.0)
+ ~Wuerfel()
+ toString() : string
+ Oberflaeche() : double
+ Volumen() : double
+ Raumdiagonale() : double



## Umbrello automatische Quelltexterzeugung liefert:

Wuerfel.h:

```
#ifndef WUERFEL_H
#define WUERFEL_H

#include <string>

/**
 * Namespace
 */
/**
 * Class Wuerfel
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 */
class Wuerfel {
/**
 * Public stuff
 */
public:
/**
 * Fields
 */
/**
 * Constructors
 */
/**
 * Empty Constructor
 */
Wuerfel ( ) { }
/**
 *
 */
Wuerfel (double mySeite=1.0) {

}
/**
 *
 */
~Wuerfel () {

}
/**
 * Accessor Methods
 */
/**
 *
 */
double get_seite ( ) {
    return seite;
}
```

```

}
/**
 *
 */
void set_seite (double value ) {
    seite = value;
}
/**
 * Operations
 */
/**
 *
 */
string toString () const {

}
/**
 *
 */
double Oberflaeche () const {

}
/**
 *
 */
double Volumen () const {

}
/**
 *
 */
double Raumdiagonale () const {

}
/**
 * Protected stuff
 */
protected:
/**
 * Fields
 */
double seite;
/**
 *
 */
/**
 * Constructors
 */
/**
 * Accessor Methods
 */
/**
 * Operations
 */
/**
 * Private stuff
 */
private:
/**
 * Fields
 */
/**
 *
 */
/**
 * Constructors

```

```

    */
    /**
    * Accessor Methods
    */
    /**
    * Operations
    */
};
#endif //WUERFEL_H

```

Wuerfel.cpp:

```

#include "wuerfel.h"
/**
 * Constructors/Destructors
 */
/**
 * Methods
 */

```

Die automatische Ergänzung von Getter/Setter-Methoden ist schon sehr schön, aber warum wird der Getter `get_seite ()` nicht als **const**-Methode modelliert?

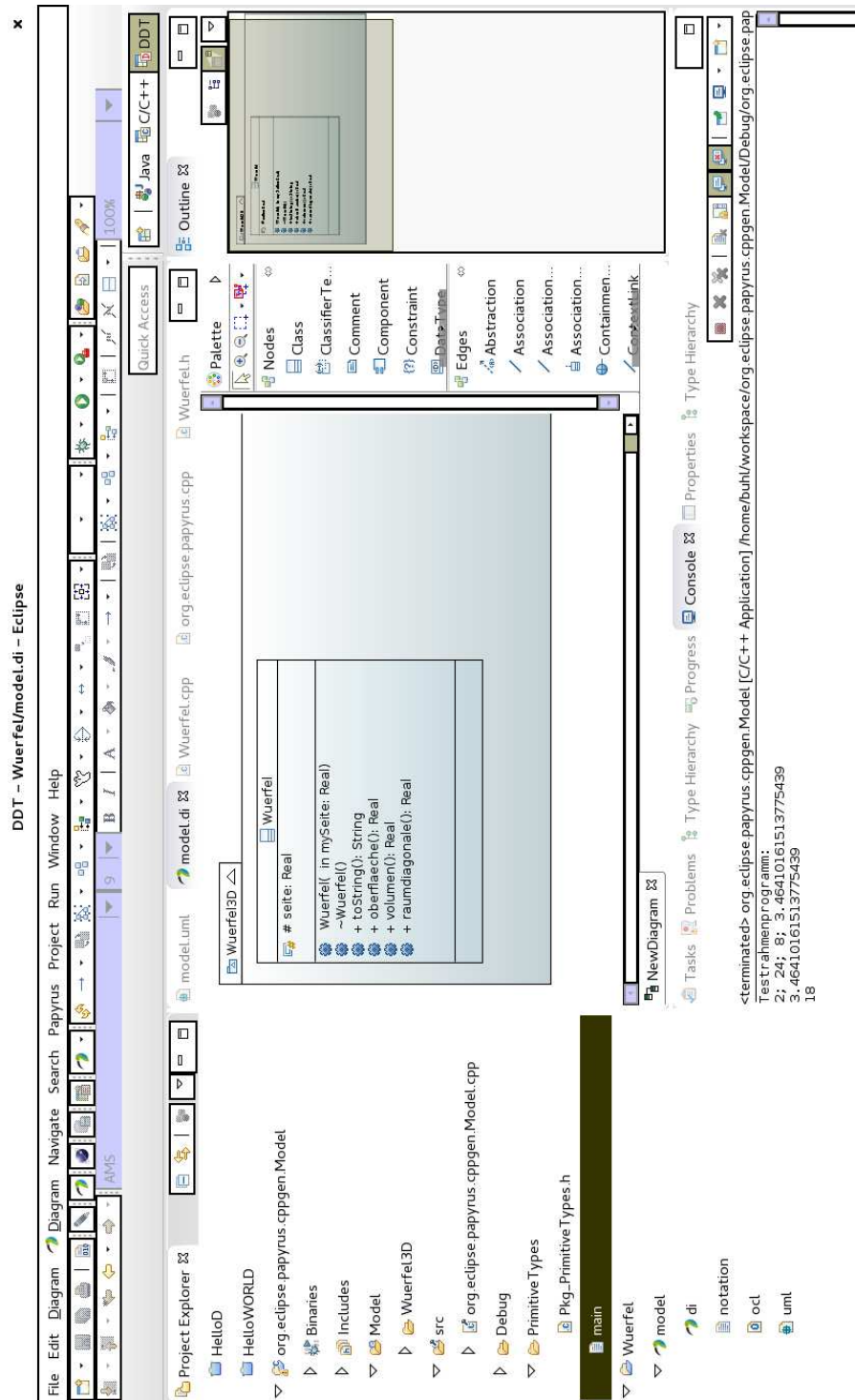
Die Aufteilung des erzeugten Codes in Header- und Implementierungsdatei sollte besser beeinflussbar sein, `std::string` sollte automatisch aus `string` erzeugt werden, die Dateinamen der Header- und Implementierungsdateien sollten großgeschriebene Klassennamen (UML2-Konvertion) enthalten, bei Existenz eines Konstruktors mit Defaultparameter solle nicht zusätzlich ein leerer Defaultkonstruktor erzeugt werden, der dann beim Übersetzen zu Namens-Mehrdeutigkeiten führt, im UML-Klassendiagramm werden die **const**-Methoden noch nicht mit einer solchen Signatur angezeigt, ...

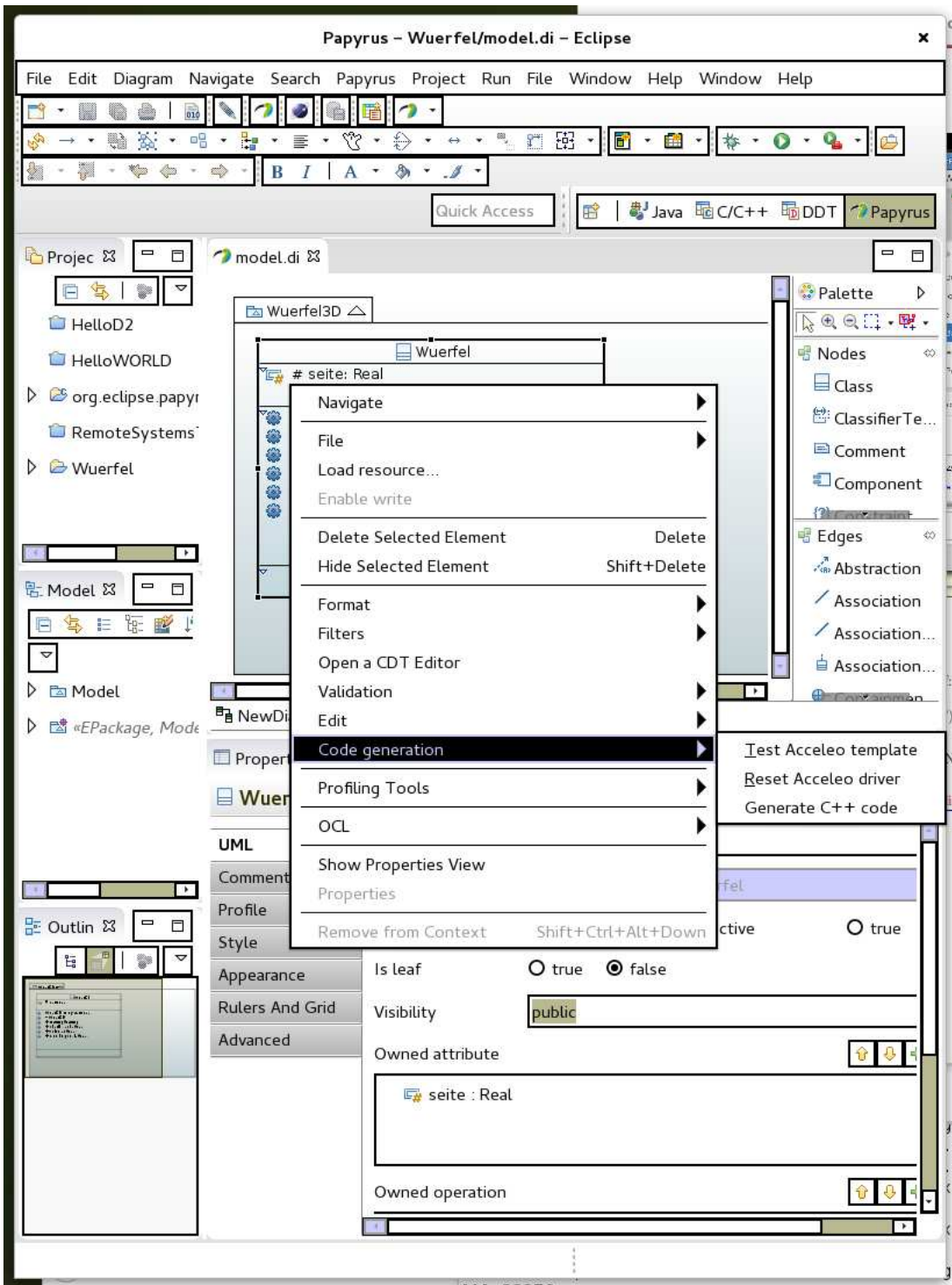
Es ist also noch viel Feinarbeit in Umbrello und den Codeerzeuger zu stecken!

Ebenfalls interessant ist die Erzeugung von UML-Diagrammen aus C++-Quellcode:

Quelltext von [ActionScript](#), [Ada](#), [C++](#), [C#](#), [D](#), [IDL](#), [Java™](#), [Javascript](#), [MySQL](#), und [Pascal](#) einlesen.

Seit Herbst 2014 funktioniert auch eclipse-papyrus (zur Erzeugung von UML-Diagrammen) mit C++-Quellcode-Erzeugung (zur Installation von eclipse-papyrus siehe Seite 30...38):





The screenshot displays the Eclipse IDE interface. The top menu bar includes File, Edit, Source, Refactor, Navigate, Search, Project, Run, Window, and Help. The Project Explorer on the left shows the project structure:

- org.eclipse.papyrus.cppgen.Model
  - Binaries
  - Includes
  - Model
    - Wuerfel3D
      - src
        - org.eclipse.papyrus.cppgen.Model.cpp
        - Debug
        - Primitive Types
          - Pkg\_Primitive Types.h
          - main
          - Wuerfel
          - model
            - di
            - notation
            - ocl
            - uml

The main editor window shows the source code for `Wuerfel.cpp` in the `org.eclipse.papyrus.cpp` package. The code is as follows:

```

20 // Code generated by Papyrus C++
4
5 #define Model_Wuerfel3D_Wuerfel_BODY
6 /******
7  /*****
8  Wuerfel class body
9  *****/
10 *****/
11 // include associated header file
12 #include "Model/Wuerfel3D/Wuerfel.h"
13
14 // Derived includes directives
15
16 namespace Wuerfel3D {
17
18 // static attributes (if any)
19
20 /**
21  *
22  * @param mySeite
23  */
24 Wuerfel::Wuerfel(Real /*in*/mySeite): seite(mySeite) {
25 }
26
27 /**
28  *
29  */
30 Wuerfel::~Wuerfel() {
31 // at the moment do nothing
32 }
33

```

The Outline view on the right shows the following elements:

- Model\_Wuerfel3D\_Wuerfel\_
- Model/Wuerfel3D/Wuerfel.h
- Wuerfel3D
  - Wuerfel::Wuerfel(Real)
  - Wuerfel::~Wuerfel()
  - Wuerfel::toString() : Strin
  - Wuerfel::oberflaeche() : R
  - Wuerfel::volumen() : Real
  - Wuerfel::raumdiagonale()

The Console view at the bottom shows the following output:

```

<terminated> org.eclipse.papyrus.cppgen.Model [C/C++ Application] /home/buhl/workspace/org.eclipse.papyrus.cppgen.Model/Debug/org.eclipse.pap
Testrahmenprogramm:
2; 24; 8; 3.46410161513775439
3.46410161513775439
18

```



The image shows an IDE interface with a Project Explorer on the left and a code editor on the right.

**Project Explorer:**

- org.eclipse.papyrus.cpp
  - Binaries
  - Includes
  - Model
    - Wuerfel3D
    - src
      - org.eclipse.papyrus.cppgen.Model.cpp
      - Debug
      - Primitive Types
        - Pkg\_Primitive Types.h
        - main

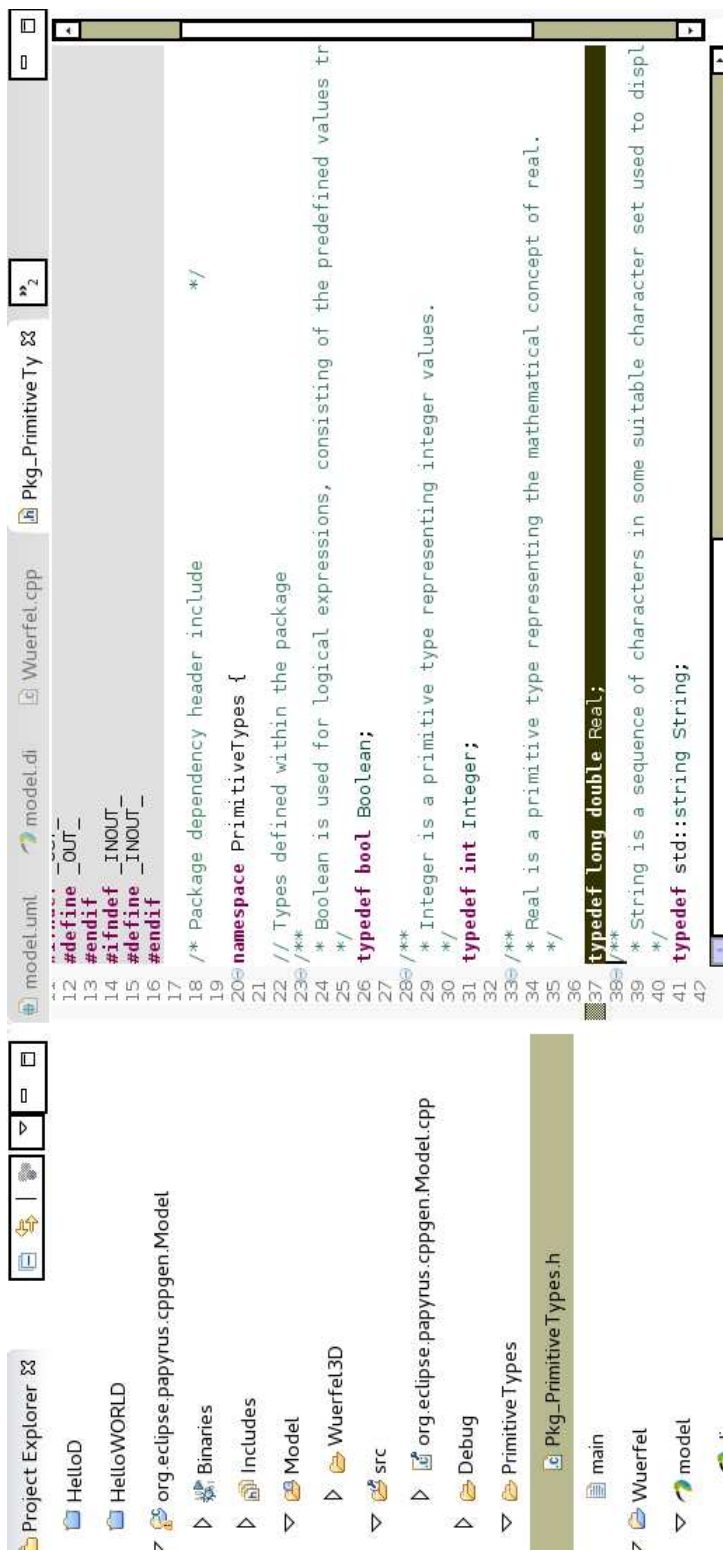
**Code Editor (Wuerfel.cpp):**

```

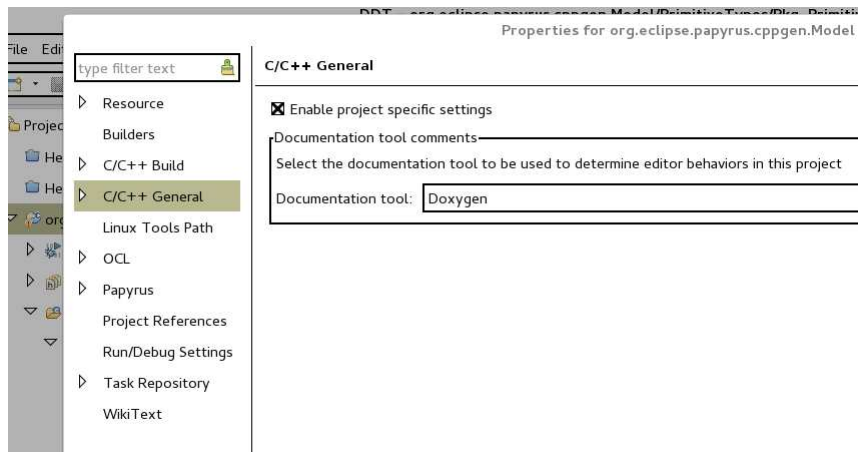
model.uml      model.di      org.eclipse.papyrus.cpp  Wuerfel.h
20 // Name      : Model.cpp
8
9 #include <iostream>
10 #include <sstream>
11 #include <limits>
12 using namespace std;
13 #include "Model/Wuerfel3D/Wuerfel.h"
14
15 int main() {
16     cout << "Testrahmenprogramm:" << endl; // prints Testrahmenprogramm:
17
18     Wuerfel3D::Wuerfel w1(2.0);
19     cout << w1.toString() << endl;
20
21     cout.precision(numeric_limits<Real>::digits10);
22     cout << w1.raumdiagonale() << endl;
23     cout << numeric_limits<Real>::digits10;
24
25     return 0;
26 }
27
  
```



Die Datei Pkg\_Primitive Typed.h muss ein bisschen modifiziert werden (Real sollte als Alias auf double oder long double gelegt werden, String statt auf char\* besser auf std::string, ...)



```
12 #define _OUT_
13 #endif
14 #ifndef _INOUT_
15 #define _INOUT_
16 #endif
17
18 /* Package dependency header include */
19
20 namespace PrimitiveTypes {
21
22 // Types defined within the package
23 /**
24  * Boolean is used for logical expressions, consisting of the predefined values tr
25  */
26 typedef bool Boolean;
27
28 /**
29  * Integer is a primitive type representing integer values.
30  */
31 typedef int Integer;
32
33 /**
34  * Real is a primitive type representing the mathematical concept of real.
35  */
36
37 typedef long double Real;
38 /**
39  * String is a sequence of characters in some suitable character set used to displ
40  */
41 typedef std::string String;
42
43 }
```



### Papyrus/Codegen/Cpp description

Papyrus Software Designer mit „side-by-side view of model and code“

## 1.23. Code-integrierte Dokumentation (mit autom. Dokumentationserzeugung)

Nun zur doxygen-Dokumentation unserer kleinen Klasse:

### Wuerfel-Klasse 0.9

The screenshot shows the Doxygen documentation for the `Wuerfel` class. At the top, there is a navigation bar with tabs for 'Hauptseite', 'Klassen', and 'Dateien'. A search box labeled 'Suchen' is on the right. Below the navigation bar, there are sub-tabs for 'Aufistung der Klassen', 'Klassen-Verzeichnis', and 'Klassen-Elemente'. The main content area is titled 'Wuerfel Klassenreferenz' and includes a search bar for 'Öffentliche Methoden | Geschützte Attrib'. The code snippet shows `#include <Wuerfel.h>` and 'Aufstellung aller Elemente'. The 'Öffentliche Methoden' section lists: `Wuerfel (double mySeite=1.0)`, `~Wuerfel ()`, `double get_seite ()`, `void set_seite (double value)`, `std::string toString () const`, `double Oberflaeche () const`, `double Volumen () const`, and `double Raumdiagonale () const`. The 'Geschützte Attribute' section lists `double seite`. The 'Ausführliche Beschreibung' section states 'Definiert in Zeile 52 der Datei Wuerfel.h.'. The 'Beschreibung der Konstruktoren und Destruktoren' section shows the constructor `Wuerfel::Wuerfel ( double mySeite = 1 . 0 )` with an 'inline' button and 'Definiert in Zeile 57 der Datei Wuerfel.h.'.

## Beschreibung der Konstruktoren und Destruktoren

`Wuerfel::Wuerfel ( double mySeite = 1.0 )` inline

Definiert in Zeile 57 der Datei `Wuerfel.h`.

`Wuerfel::~Wuerfel ( )` inline

Definiert in Zeile 60 der Datei `Wuerfel.h`.

## Dokumentation der Elementfunktionen

`double Wuerfel::get_seite ( )` inline

Definiert in Zeile 64 der Datei `Wuerfel.h`.

`double Wuerfel::Oberflaeche ( ) const` inline

Definiert in Zeile 74 der Datei `Wuerfel.h`.

`double Wuerfel::Raumdiagonale ( ) const` inline

Definiert in Zeile 80 der Datei `Wuerfel.h`.

`void Wuerfel::set_seite ( double value )` inline

Definiert in Zeile 67 der Datei `Wuerfel.h`.

`std::string Wuerfel::toString ( ) const` inline

Beachte dabei die speziellen dokumentationserzeugenden Kommentare gemäß [Doxygen-Syntax](#), die die automatische Codeerzeugung von Umbrello schon eingebaut hat.

Ein Aufruf von `doxygen` erzeugt aus dem Quelltext automatisch Dokumentation im pdf und/oder html-Format. Vergleiche dazu:

[Doxywizard usage](#)

[Doxygen: Generate documentation from source code](#)

[Doxygen Manual](#)

Hinweis zum Formatieren der pdf-Dokumentation:

```
cd documentation/latex
```

```
make pdf
```

```
evince refman.pdf
```

**Bemerkung:** Die `texlive`-Pakete `texlive-sectsty`, `texlive-tocloft`, `texlive-xtab` und `texlive-multirow` müssen eventuell nachinstalliert werden, da sie für Doxygen-Referenzmanuals benötigt werden.

Software-Dokumentationswerkzeug

Seite 22: Dokumentationsmethoden für Software-Bibliotheken

PHPDoc

Doxygen/Doxywizard

Javadoc

AdaBrowse

Document!X

Doc-O-Matic

Haddock: A Haskell Documentation Tool

Seite 28: Vergleich der Dokumentationswerkzeuge

## 1.24. Ungarische Notation und deren nicht unbedingt codequalitätssteigernde Auswirkungen

Ungarische Notation  
Kritikpunkte  
Namenskonventionen

## 1.25. Eclipse CDT in Zusammenarbeit mit Umbrello/Papyrus, Doxygen, ...

Erste Eclipse C++-Projekte:

<http://max.berger.name/howto/cdt/ar01s04.jsp#helloworld>

Umbrello in eclipse starten:

Run

External Tools

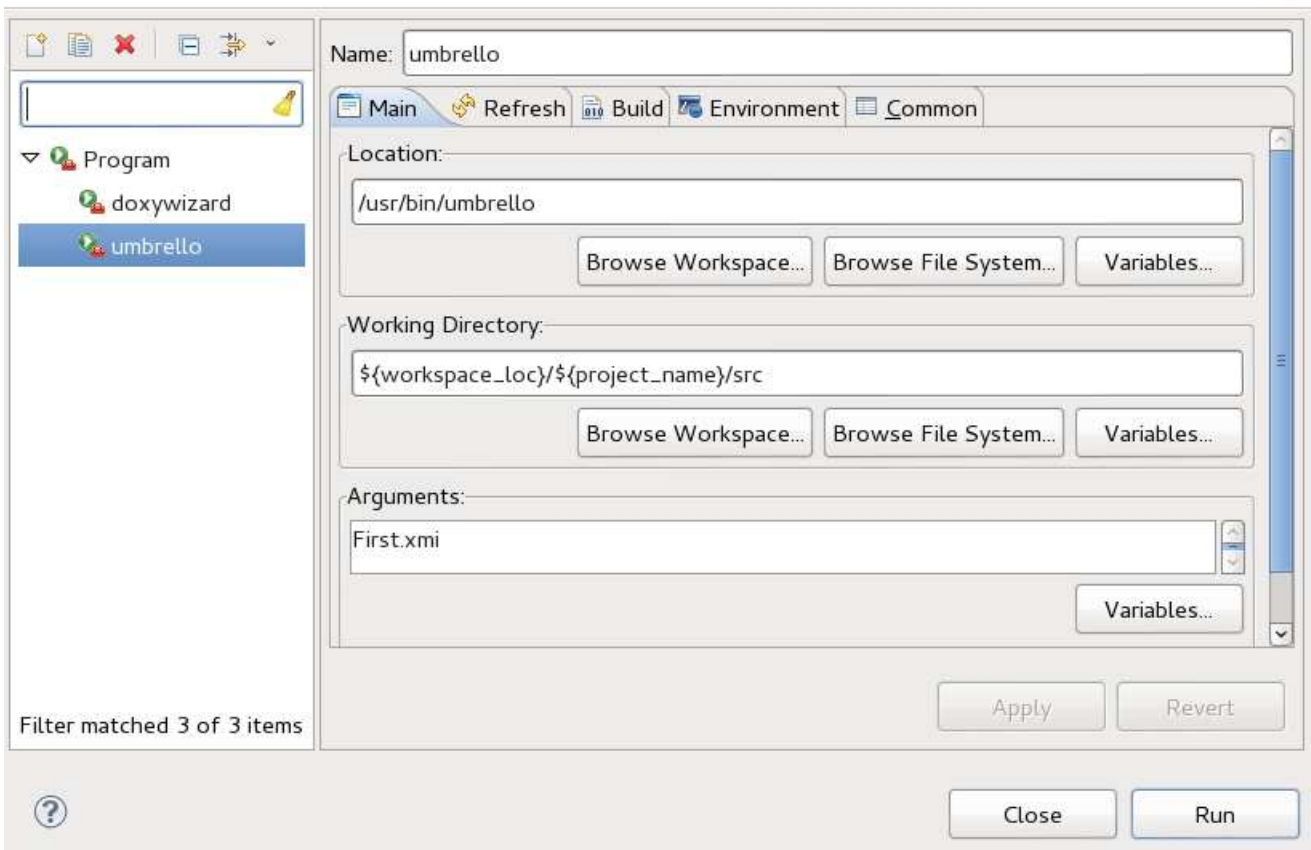
External Tools Configuration

New launch configuration

Für ein neues (aktuelles) Projekt First etwa:

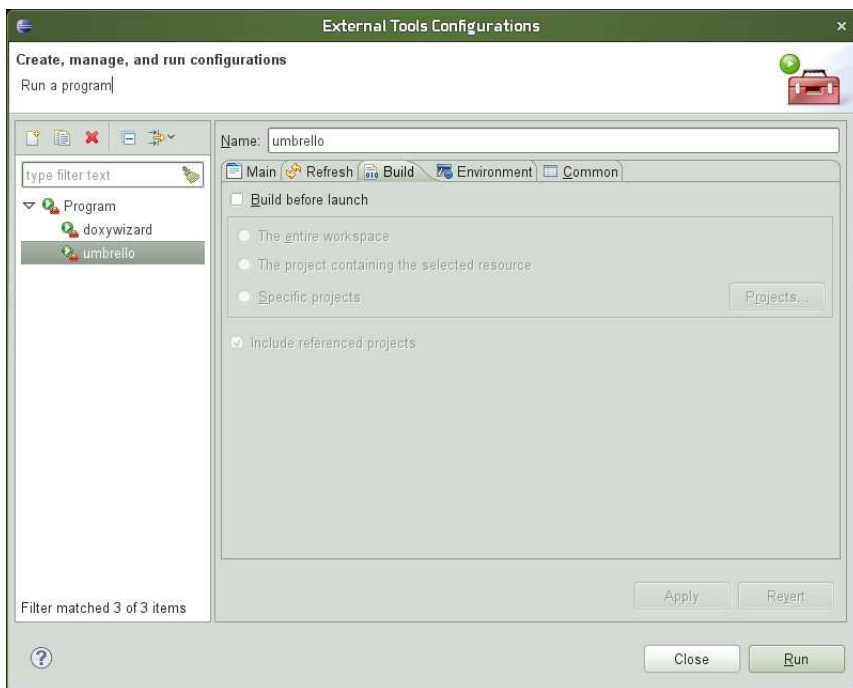
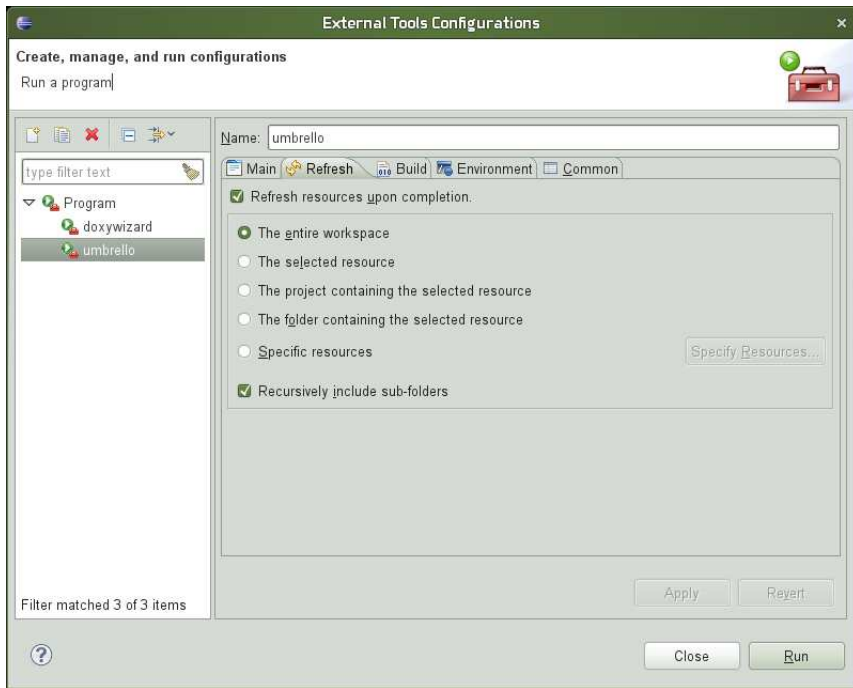
Create, manage, and run configurations

Run a program



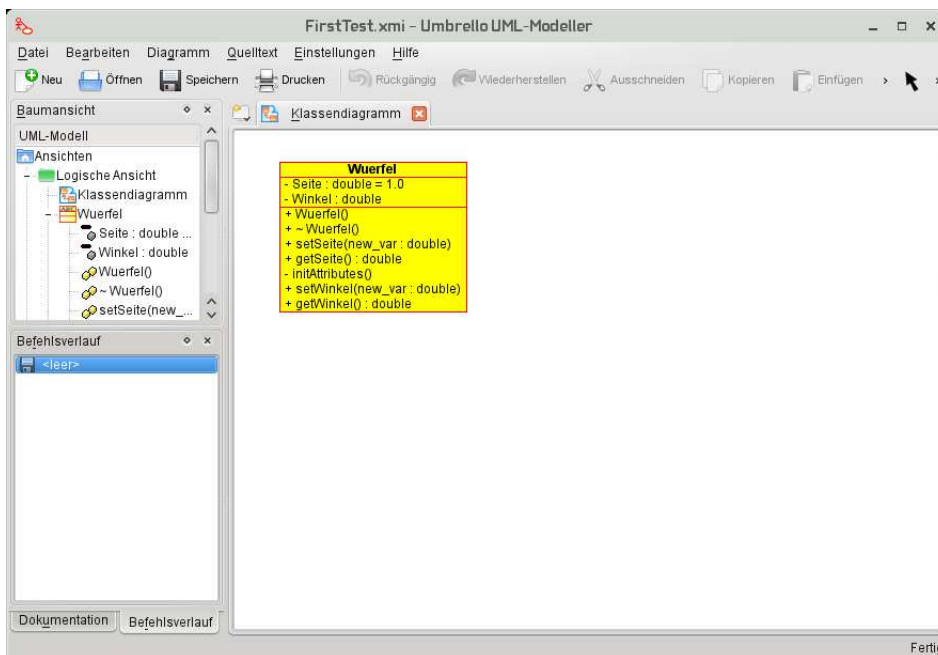
(Vergessen Sie nicht, im Eclipse-Projektexplorer Ihr aktuelles Projekt anzuklicken und dadurch auszuwählen.)

[External tools variables](#)





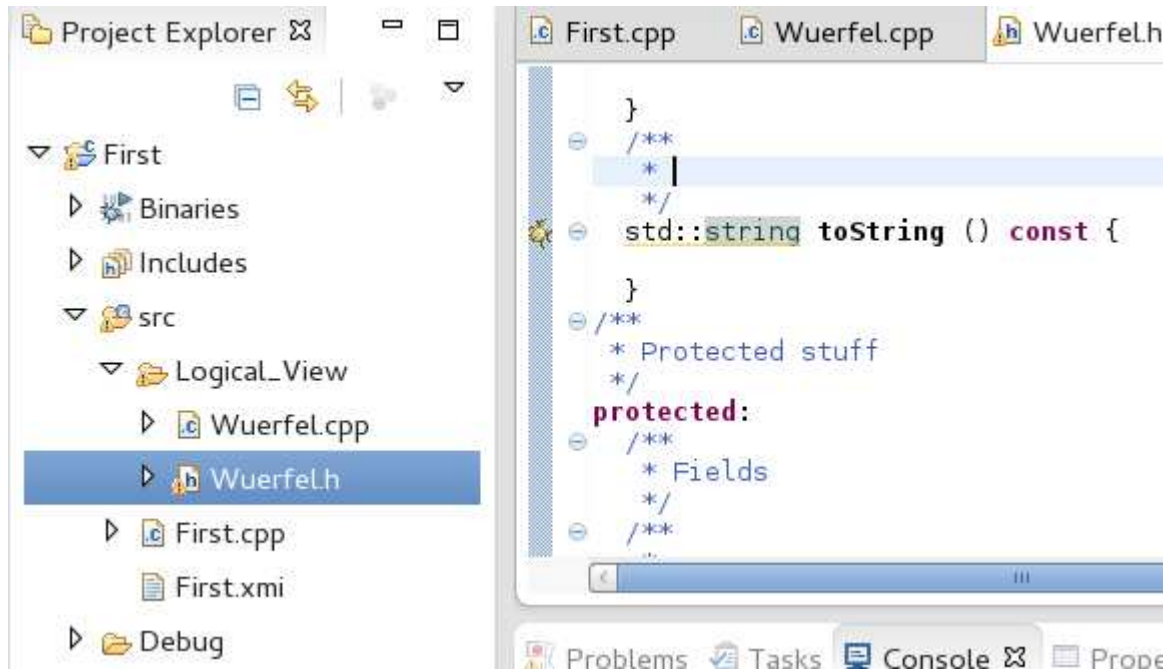
Nach Betätigung des Run-Knopfes können Sie ein UML-Modell konzipieren



und mittels des Umbrello-Assistenten für Quelltext-Generierung nach Angabe des Ordners für die Quelldateien



die Quelldateien in Eclipse unmittelbar im Projekt-src-Unterknoten LogicalView benutzen:



Selbst das Quelltext-Klassen(re)importieren von in Eclipse geänderten Klassen zurück ins Umbrello-UML-Modell funktioniert einwandfrei.

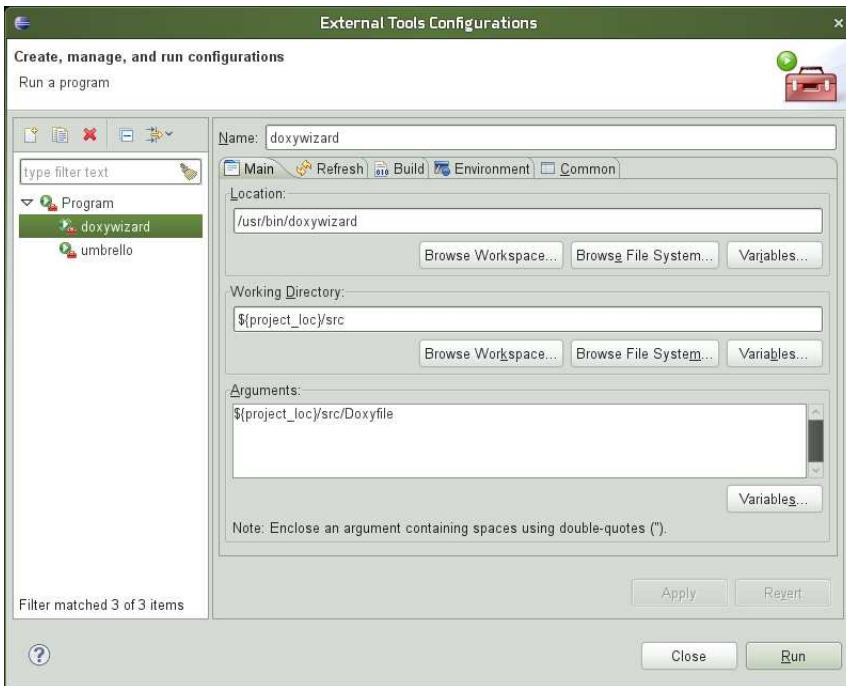
### Doxygen-Integration in eclipse:

Run

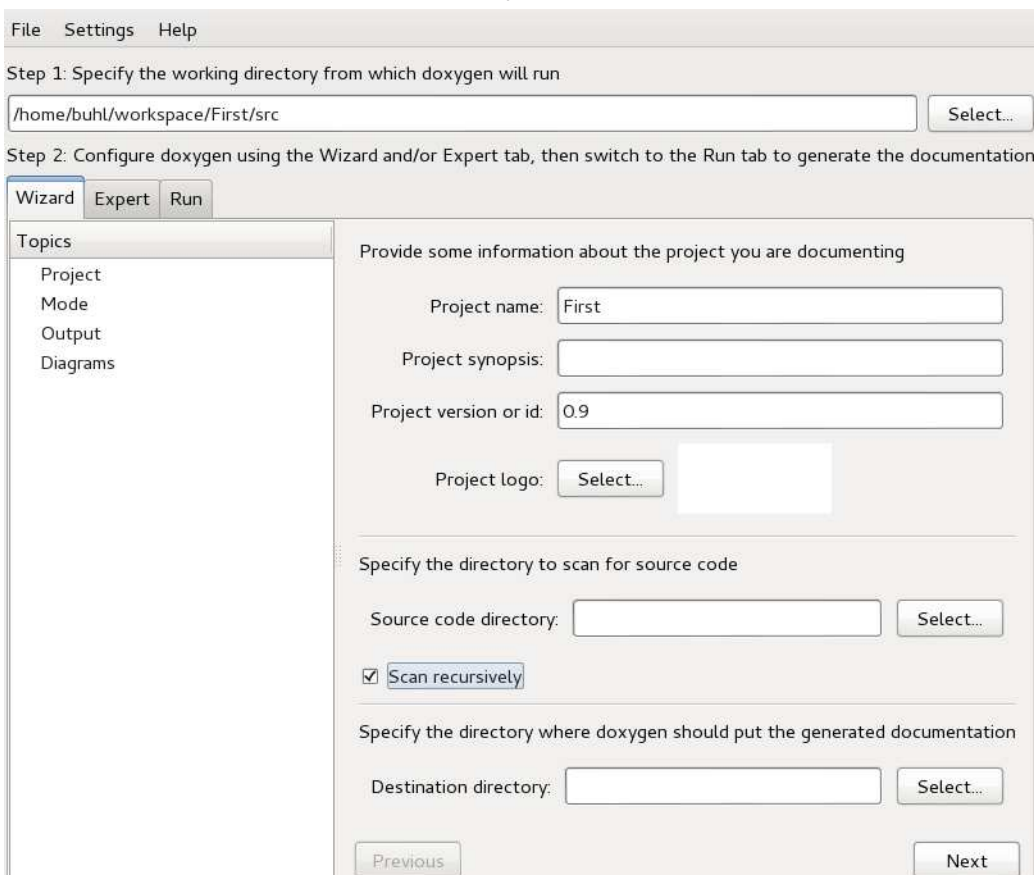
External Tools

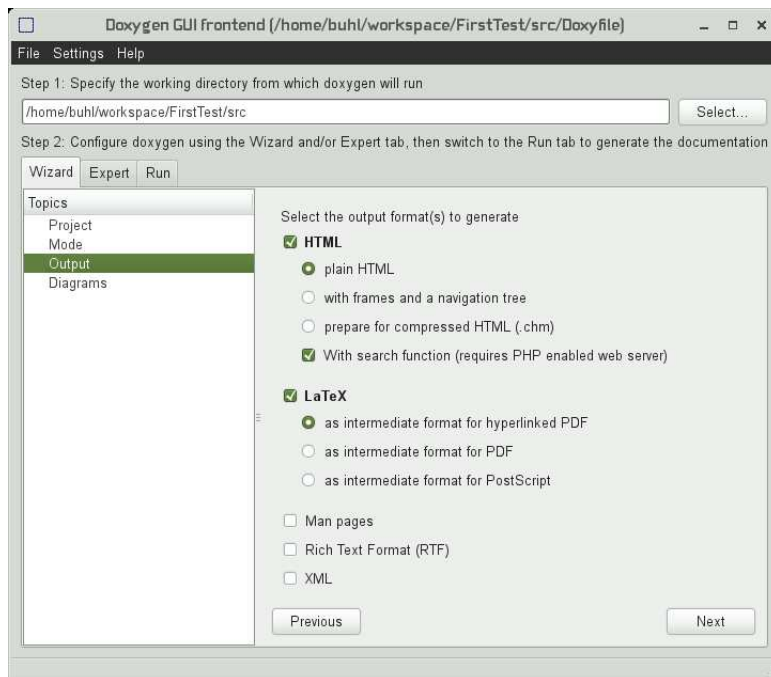
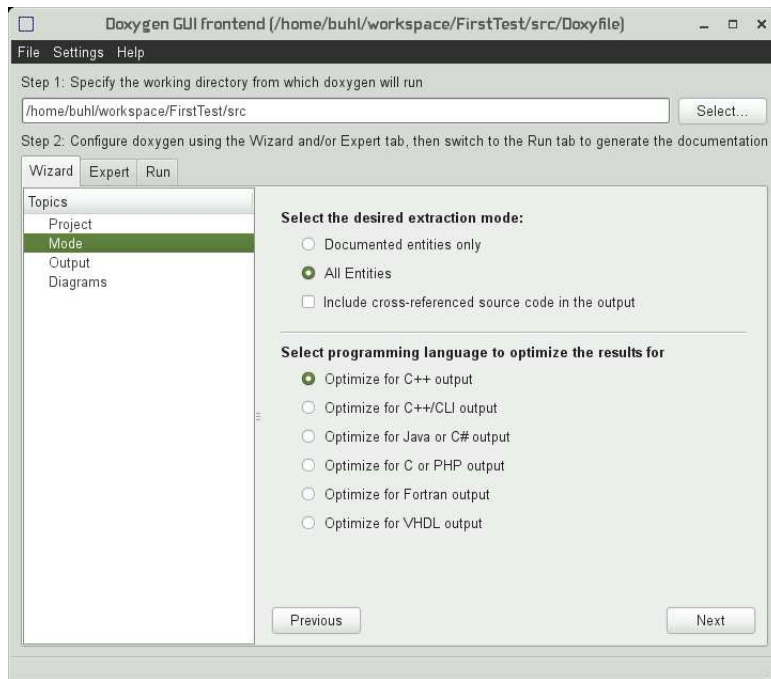
External Tools Configuration

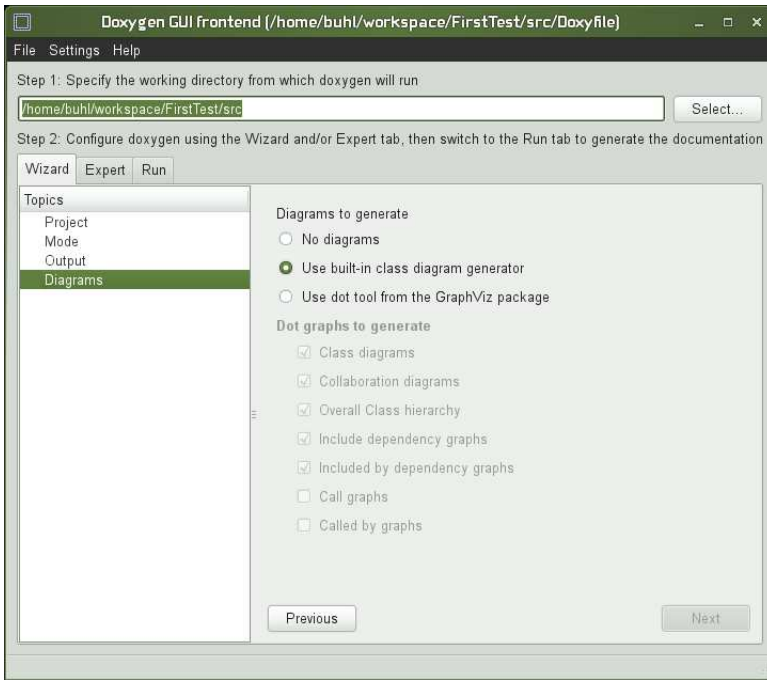
New launch configuration



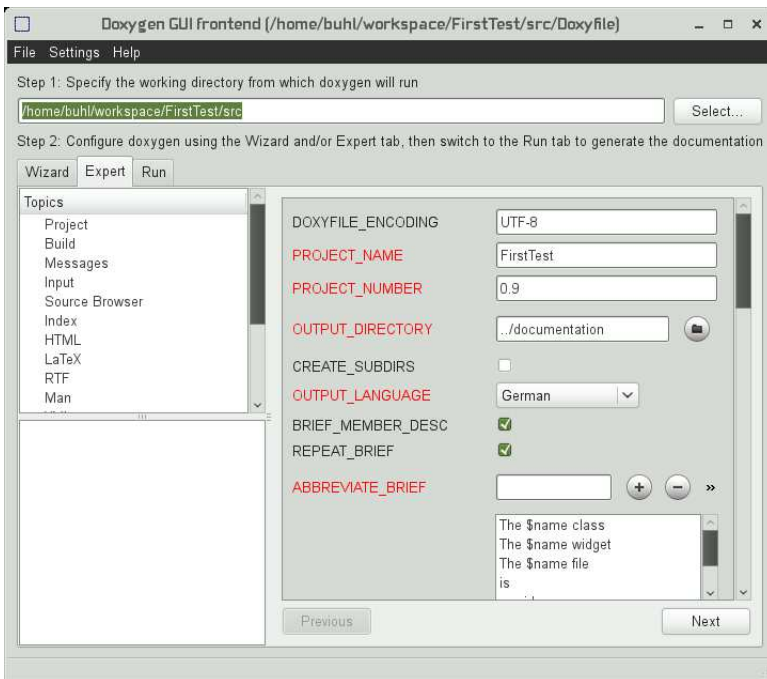
Refresh und Build-Einstellungen analog wie bei umbrello. Nach Betätigung des Run-Knopfes können Sie nun die Doxywizard-Konfiguration anlegen:







Vergessen Sie nicht, über den Expert-Reiter die `OUTPUT_LANGUAGE` auf German umzustellen:



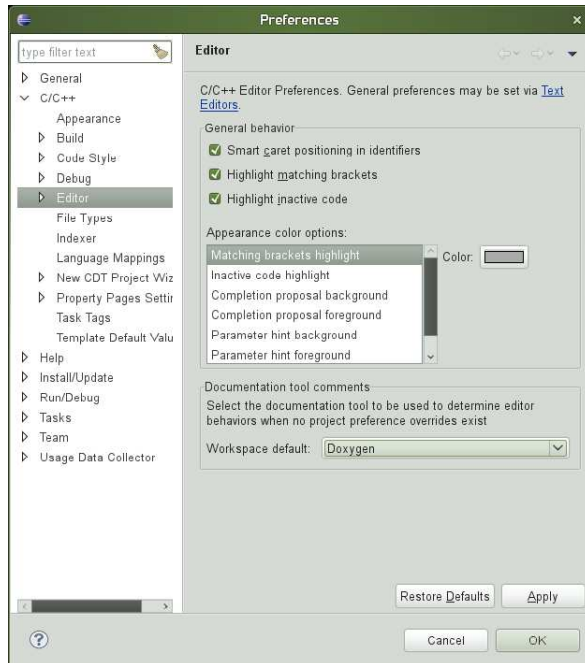
Schließlich müssen Sie noch die eclipse-Editor-Unterstützung für doxygen-Spezialkommentare aktivieren:

Windows

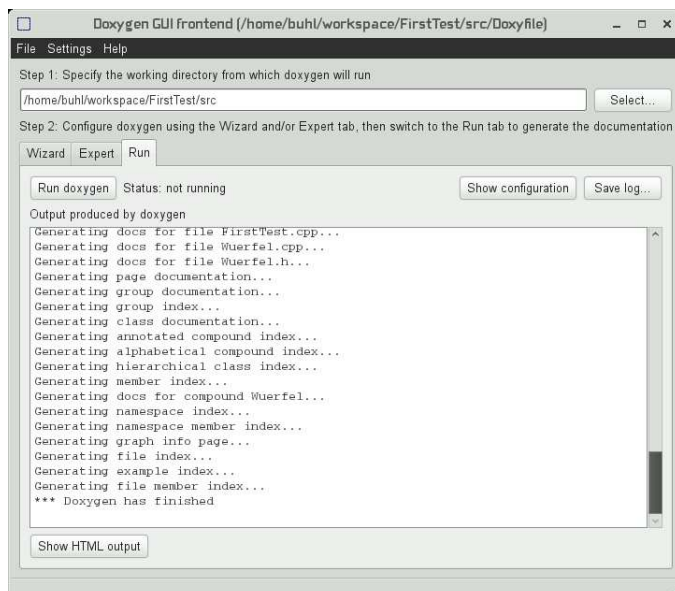
Preferences

C/C++

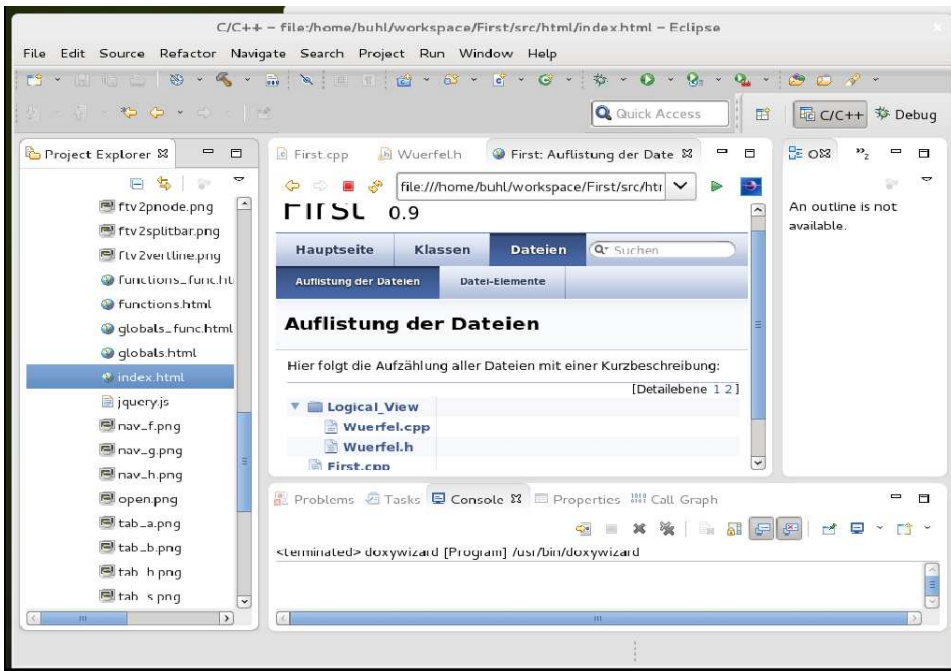
Editor



Jetzt können Sie über das externe Tool doxywizard jederzeit eine aktuelle Dokumentationsversion erzeugen



und mittels des Show HTML output-Knopfes in einem externen Browser betrachten oder Sie doppelklicken die Datei index.html im documentation-Unterordner des eclipse Project Explorers:



doxygen-Spezialkommandos: <http://www.stack.nl/~dimitri/doxygen/docblocks.html>

Beachten Sie insbesondere @mainpage

```
/*!  
 * @mainpage Wuerfel, ...  
 *  
 * @version 0.99  
 * @author HJB  
 * @date 25. AprilJanuar 2011  
 */  
  
/*! Testrahmenprogramm  
 *  
 * erzeugt einige Test-Wuerfel und  
 * druckt deren Attribute  
 *  
 * @param[in] argc Anzahl der Kommandozeilen-Parameter  
 * @param[in] argv Feld mit den Kommandozeilen-Parametern  
 * @return EXIT_SUCCESS  
 * @warning eventuell tritt der Fehler ...  
 */  
  
#include <stdlib.h>  
int main(int argc, char **argv) {  
    //...  
    return EXIT_SUCCESS;  
}
```

sowie die Contract-Dokumentation in doxygen:

`@pre { description of the precondition }`

Starts a paragraph where the precondition of an entity can be described. The paragraph will be indented. The text of the paragraph has no special internal structure. All visual enhancement commands may be used inside the paragraph. Multiple adjacent @pre commands will be joined into a single paragraph. Each precondition will start on a new line. Alternatively, one @pre command may mention several preconditions. The @pre command ends when a blank line or some other sectioning command is encountered.

`@post { description of the postcondition }` Starts a paragraph where the postcondition of an entity can be described. The paragraph will be indented. The text of the paragraph has no special internal structure. All visual enhancement commands may be used inside the paragraph. Multiple adjacent @post commands will be joined into a single paragraph. Each postcondition will start on a new line. Alternatively, one @post command may mention several postconditions. The @post command ends when a blank line or some other sectioning command is encountered.



`@invariant { description of invariant }` Starts a paragraph where the invariant of an entity can be described. The paragraph will be indented. The text of the paragraph has no special internal structure. All visual enhancement commands may be used inside the paragraph. Multiple adjacent `@invariant` commands will be joined into a single paragraph. Each invariant description will start on a new line. Alternatively, one `@invariant` command may mention several invariants. The `@invariant` command ends when a blank line or some other sectioning command is encountered.

Ein Beispiel für die Contract-Befehle in doxygen

```
/*!
 * Konstruktor
 * @param[in] mySeite
 * @pre{ mySeite >= 0 }
 */
...
/*!
 * Oberflaeche
 * @return Oberflaeche
 * @post{ approximatelyEqualTo(result, 6.0 * Seite * Seite, 1.0) }
...
/*!
 * 3D-Wuerfel
 *
 * @invariant{ Seite >= 0.0 }
 */
class Wuerfel{
...
}
*/
```

und die Ergebnisse:

## Wuerfel Klassenreferenz

#include <Wuerfel.h>

[Aufstellung aller Elemente](#)

### Öffentliche Methoden

	<b>Wuerfel</b> (double mySeite=1.0)
virtual	<b>~Wuerfel</b> ()
virtual bool	<b>invariant</b> ()
std::string	<b>toString</b> ()
double	<b>Oberflaeche</b> () = 6.0 * pow( <b>Seite</b> , 2)
double	<b>Volumen</b> () = pow( <b>Seite</b> , 3)
double	<b>Raumdiagonale</b> () = <b>Seite</b> * sqrt(3.0)

### Geschützte Attribute

double	<b>Seite</b>
--------	--------------

### Ausführliche Beschreibung

3D-Wuerfel

#### Invariant:

```
{ Seite >= 0.0 }
```

Definiert in Zeile **33** der Datei [Wuerfel.h](#).

### Beschreibung der Konstruktoren und Destruktoren

<b>Wuerfel::Wuerfel ( double mySeite = 1.0 )</b>
Konstruktor
<b>Parameter:</b> [in] mySeite
<b>Vorbedingung:</b> { mySeite >= 0 }
Definiert in Zeile <b>20</b> der Datei <a href="#">Wuerfel.cpp</a> .

**Tips:** 1) Ctrl-Space öffnet im Eclipse-Quellcodeeditor-Fenster mögliche Ergänzungen der bisher eingegebenen Zeichenfolge.

2) Am Zeilenanfang nach „/\*!“ eingegebenes Return ergänzt den begonnen Doxygen-Kommentar um zur Folgezeile gehörige -Doxygen-Dokumentationskommentare.

### Dokumentation der Funktionen

<b>static double myown_sqrt ( double x ) [static]</b>
Quadratwurzel nach Newton
<b>Parameter:</b> [in] x
<b>Rückgabe:</b> Wurzel
<b>Vorbedingung:</b> { x >= 0.0 }
<b>Nachbedingung:</b> { approximatelyEqualTo(result*result, x, 1.0) }
Definiert in Zeile <b>37</b> der Datei <a href="#">double_math.h</a> .
Benutzt <a href="#">approximatelyEqualTo()</a> , <a href="#">ENSURE()</a> und <a href="#">result</a> .
Wird benutzt von <a href="#">main()</a> .

Beispieldokumentationen:

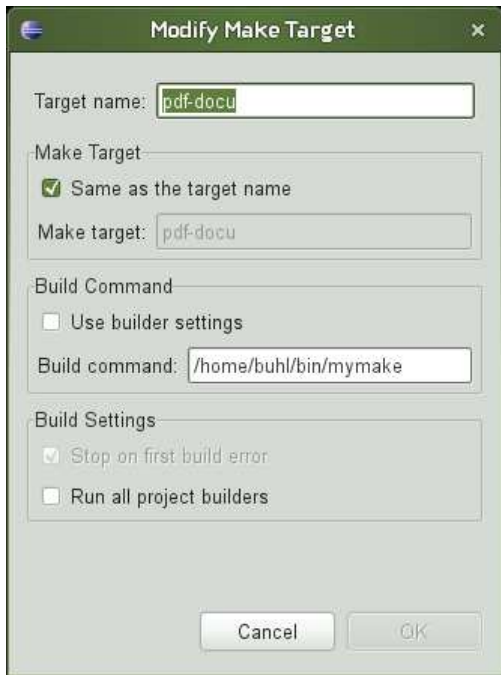
<http://gcc.gnu.org/onlinedocs/libstdc++/latest-doxygen/modules.html> (STL-Docu)

<http://www.stack.nl/~dimitri/doxygen/results.html>

<http://www.stack.nl/~dimitri/doxygen/projects.html>

**Ein Make-Target zur Erzeugung der pdf-doxygen-Dokumentation:**  
Bei angeklickter Projekt-Quelldatei

Project  
  Make Target    >  
    Create ...



Erzeugung der Hilfsdatei /home/username/bin/mymake:

```
> cd $HOME/bin  
> emacs mymake  
#!/bin/sh  
cd ../src/latex  
make
```

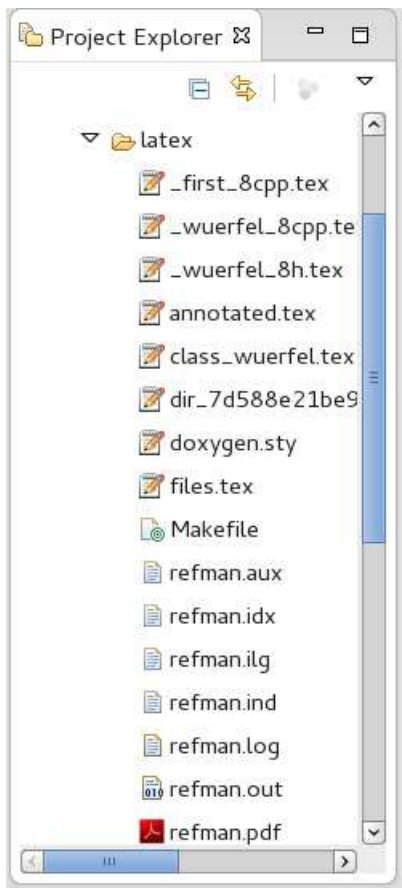
```
>chmod 755 mymake
```

Im Kontext einer Quelldatei kann nun mittels

Project

```
Make Target    >  
Build ...  
pdf-docu      Build
```

die Datei refman.pdf erzeugt



und benutzt werden.

## 1.26. nana-Installation

(Vergleiche Übungsblatt 8)

Laden von nana durch

<http://savannah.gnu.org/projects/nana/>  
die Dateien `nana-2.5.tar.gz` und `nana-2.5.tar.gz.sig`. Besorgen des öffentlichen gpg-Schlüssels von Phil Maker (über <http://savannah.gnu.org/project/memberlist-gpgkeys.php?group=nana> oder einen der gpg-Keyserver mittels seahorse, gpg2, gpa, kpgg, gpgv2, ...) und überprüfen der Authentizität der heruntergeladenen Datei:

```
p> gpg2 --search-keys Phil Maker
gpg: suche nach "Phil Maker" auf hkp-Server wwwkeys.eu.pgp.net
(1)  Phil Maker <pjm@gnu.org>
    1024 bit DSA key AD58EA42, erzeugt: 2005-04-13
(2)  Phil Maker <pjm@gnu.org>
    2048 bit RSA key 7D25F017, erzeugt: 1998-11-03
Keys 1-2 of 2 for "Phil Maker". Eingabe von Nummern, Nächste (N) oder Abbrechen (Q) >
1
gpg: fordere Schlüssel AD58EA42 von hkp-Server wwwkeys.eu.pgp.net an
gpg: Schlüssel AD58EA42: Öffentlicher Schlüssel "Phil Maker <pjm@gnu.org>" importiert
gpg: Anzahl insgesamt bearbeiteter Schlüssel: 1
gpg:                               importiert: 1

> gpg2 nana-2.5.tar.gz.sig
gpg: Unterschrift vom Mi 13 Apr 2005 05:48:25 CEST mittels DSA-Schlüssel ID AD58EA42
gpg: Korrekte Unterschrift von "Phil Maker <pjm@gnu.org>"
gpg: WARNUNG: Dieser Schlüssel trägt keine vertrauenswürdige Signatur!
gpg:      Es gibt keinen Hinweis, daß die Signatur wirklich dem vorgeblichen Besitzer gehört.
Haupt-Fingerabdruck = 1806 4A1E F48D 8372 3BE8 156E A266 78A3 AD58 EA42
```

Entpacken des Archivs:

```
tar xfz nana-2.5.tar.gz
cd nana-2.5
```

Führen Sie ein paar wegen inzwischen in der Programmiersprache C durchgeführter Änderungen nötige Modifikationen durch

Zeile 87 von `src/I.h`:

```
void _I_default_handler(const char *expr, const char *file, int line);
```

Zeile 48 von `src/I.c`:

```
void _I_default_handler(const char *exprn, const char *file, int line) {
```

statt Zeile 84 von `src/nana-config.h.in`:

```
typedef void*(FKTxxy)(unsigned int);
static FKTxxy *_nana_p = malloc; /* this costs us storage space */
```

Zeile 82 von `src/nana-clg.in`:

```
`${NANACC-@CC@} $CPPFLAGS $* -L$HOME/lib -lnana -o a.out &&
```

Zeile 73 von `src/nana.in`:

```
@CPP@ -I$HOME/include -D_NANA_FILTER_ $* 2>/dev/null | `${NANAEXECDIR-@libexecdir@}/nanafilter
```

Zeile 63 in `shortform/nana-sfdir.in`:

```
`${NANABIN-@bindir@}/nana-sfg $f >$TARGET/$f
```

neue Zeile 58 in `shortform/nana-sfdir.in` einfügen:

```
-name \*.cpp -o \
```

```
Zeile 33 von src/nana_error.c:  
#include <stdlib.h>
```

```
Zeile 36 von src/nanafilter.c:  
#include <stdlib.h>
```

```
Zeile 31 von src/I.c:  
#include <stdlib.h>
```

und installieren von nana in Ihrem Home-Verzeichnis:

```
nana-2.5> ./configure --prefix=~  
creating cache ./config.cache  
checking host system type... i686-pc-linux-gnu  
...  
nana-2.5> make  
Making all in src  
make[1]: Entering directory '/home/User/Desktop/nana-2.5/src'  
gcc -DPACKAGE=\"nana\" -DVERSION=\"2.5\" -DHAVE_VSPRINTF=1 -DHAVE_VSNPRINTF=1 -DHAVE_GETTIMEOFDAY=1 -I. -I. -g -I. \\  
-I/home/User/include -g -I. -I/home/User/incllude -c I.c  
...  
nana-2.5> make install  
Making install in src  
make[1]: Entering directory '/home/User/Desktop/nana-2.5/src'  
make[2]: Entering directory '/home/User/Desktop/nana-2.5/src'  
/bin/sh ../mkinstalldirs ~/lib  
...
```

(Lesen Sie bei Bedarf die Datei INSTALL beziehungsweise doc/nana.ps)  
Ändern Sie Ihr Linux-Environment (Datei ~/.bashrc) durch Hinzufügen von:

```
export NANAPKGDIR=$HOME/share/nana  
# export NANARUN_GDB_OPTIONS=  
export NANABIN=~/.bin  
export NANACC=gcc  
  
export CPPFLAGS="-g -I. -I$HOME/include"  
export CFLAGS="-g -I. -I$HOME/include"  
export LDFLAGS="-L $HOME/lib"  
export LDLIBS="-lnana"
```

Veränderte Verzeichnisse Ihres Accounts:

```
~/bin  
~/include  
~/lib  
~/share/nana
```

## 1.27. Vertragsverletzungen zur Laufzeit

... bei Applikationsstart von der Kommandozeile:

```
> ./main
Testrahmenprogramm fuer Wuerfel:
Wuerfel.cpp:17: I(mySeite >= 0) failed; dumping core
Abgebrochen (Speicherabzug geschrieben)

> ls -al core
-rw----- 1 buhl users 409600 16. Jan 11:47 core

> cat Wuerfel.cpp
...
Wuerfel::Wuerfel(double mySeite): Seite(mySeite)
{
    REQUIRE(mySeite >= 0);
    ENSURE(invariant());
}
...

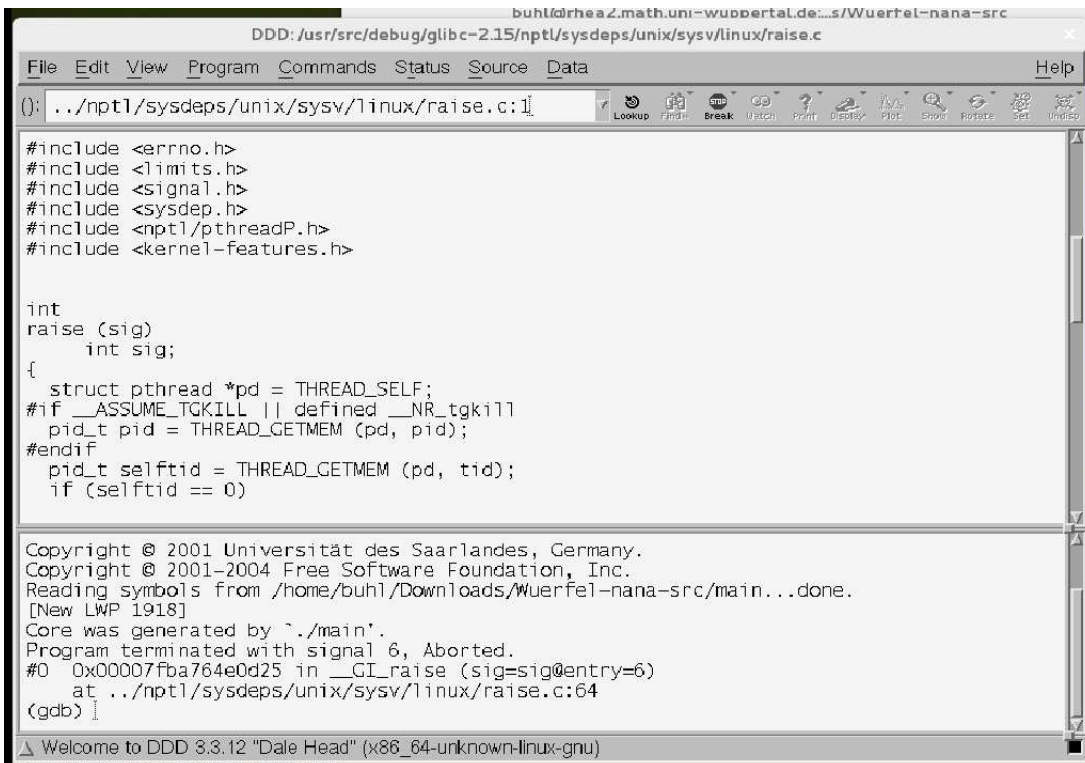
> cat main.cpp
...
int main(int argc, char* argv[]) {
    std::cout << "Testrahmenprogramm fuer Wuerfel:" << std::endl;

    Wuerfel w(-2.5);
}
```

**Bemerkung:** Es kann notwendig sein, in die Userkonfiguration `$HOME/.bashrc` des Programmierers die Zeile `ulimit -S -c unlimited` aufzunehmen (vergleiche [generate a core dump in linux](#)).

**Analyse des Hauptspeicherauszeuges (core-Files) mit ddd:**

```
> ddd main core
```

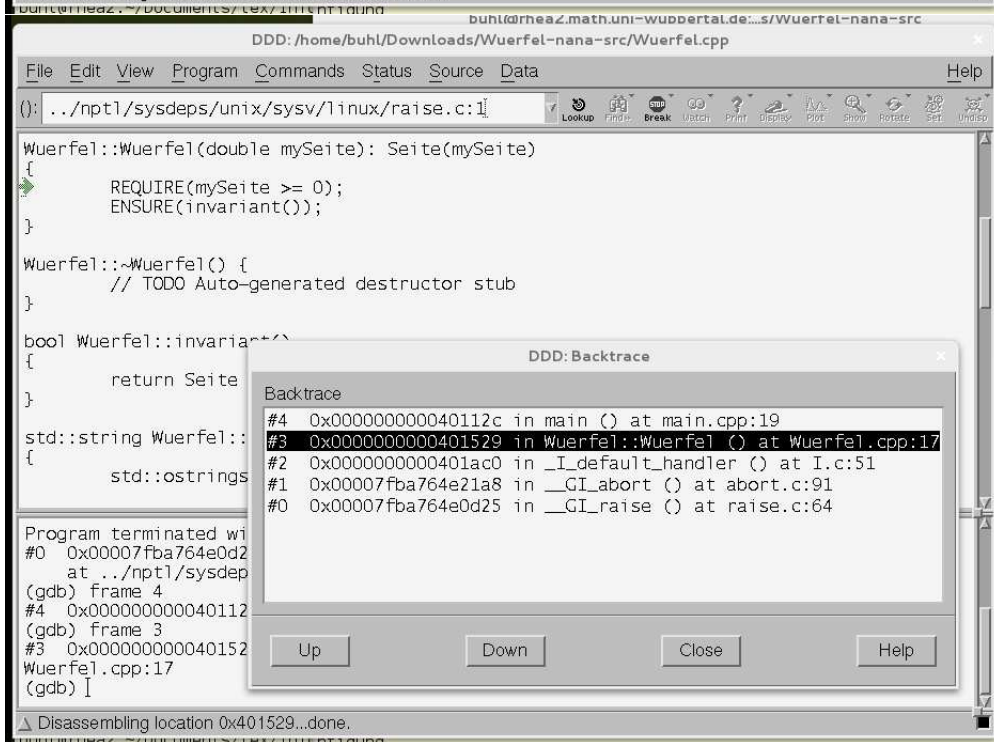
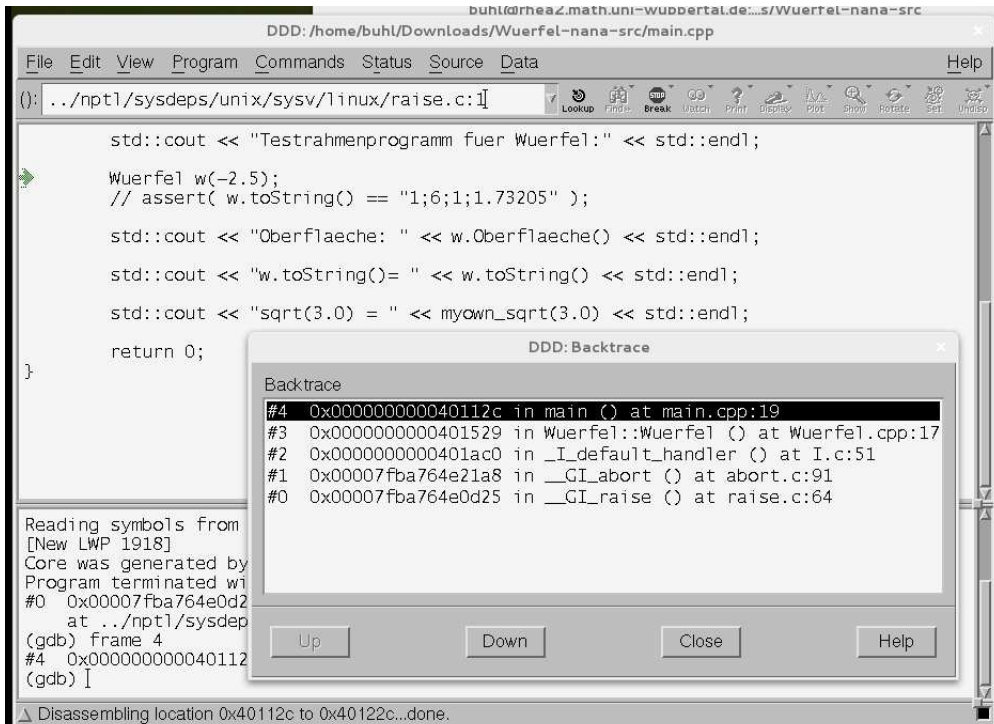


```
DDD: /usr/src/debug/glibc-2.15/npt1/sysdeps/unix/sysv/linux/raise.c
File Edit View Program Commands Status Source Data Help
(): ../npt1/sysdeps/unix/sysv/linux/raise.c:1
#include <errno.h>
#include <limits.h>
#include <signal.h>
#include <sysdep.h>
#include <npt1/theadP.h>
#include <kernel-features.h>

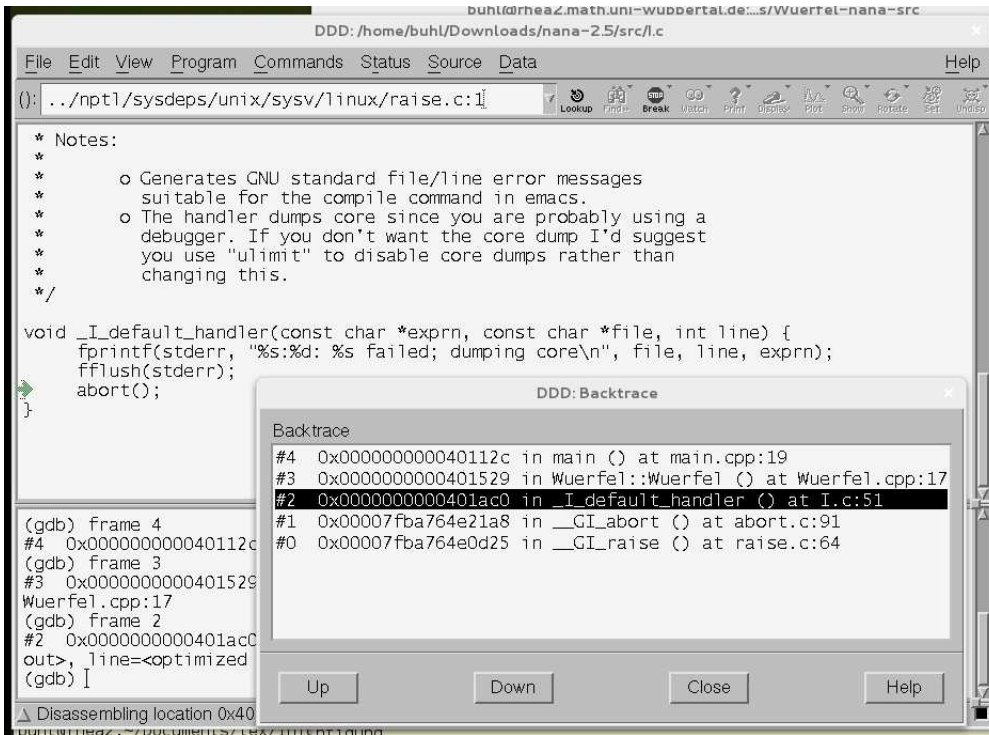
int
raise (sig)
    int sig;
{
    struct pthread *pd = THREAD_SELF;
    #if __ASSUME_TGKILL || defined __NR_tgkill
    pid_t pid = THREAD_GETMEM (pd, pid);
    #endif
    pid_t selftid = THREAD_GETMEM (pd, tid);
    if (selftid == 0)

Copyright © 2001 Universität des Saarlandes, Germany.
Copyright © 2001–2004 Free Software Foundation, Inc.
Reading symbols from /home/buhl/Downloads/Wuerfel-nana-src/main...done.
[New LWP 1918]
Core was generated by './main'.
Program terminated with signal 6, Aborted.
#0  0x00007fba764e0d25 in __GI_raise (sig=sig@entry=6)
    at ../npt1/sysdeps/unix/sysv/linux/raise.c:64
(gdb)

Welcome to DDD 3.3.12 "Dale Head" (x86_64-unknown-linux-gnu)
```



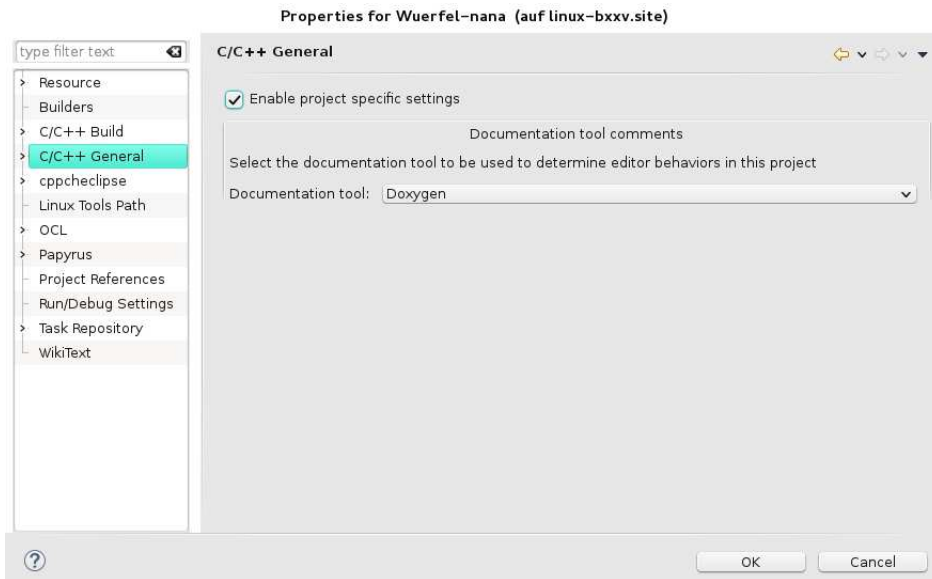




## 1.28. C++-nana-Contracts in Eclipse

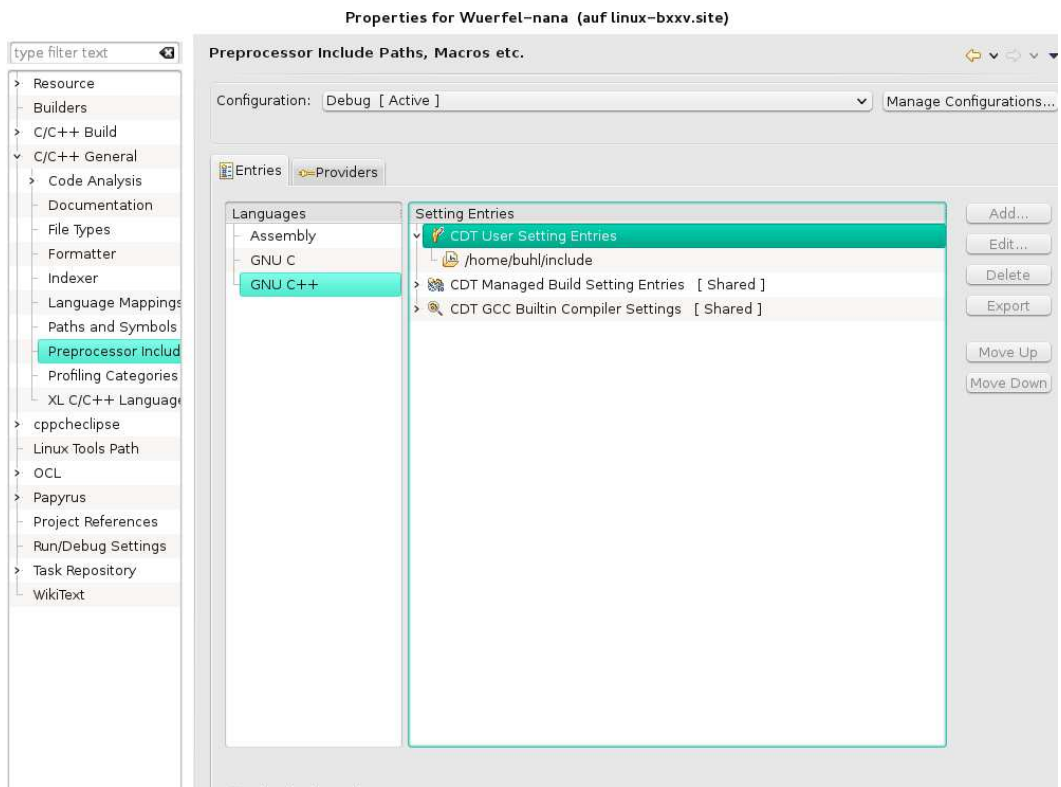
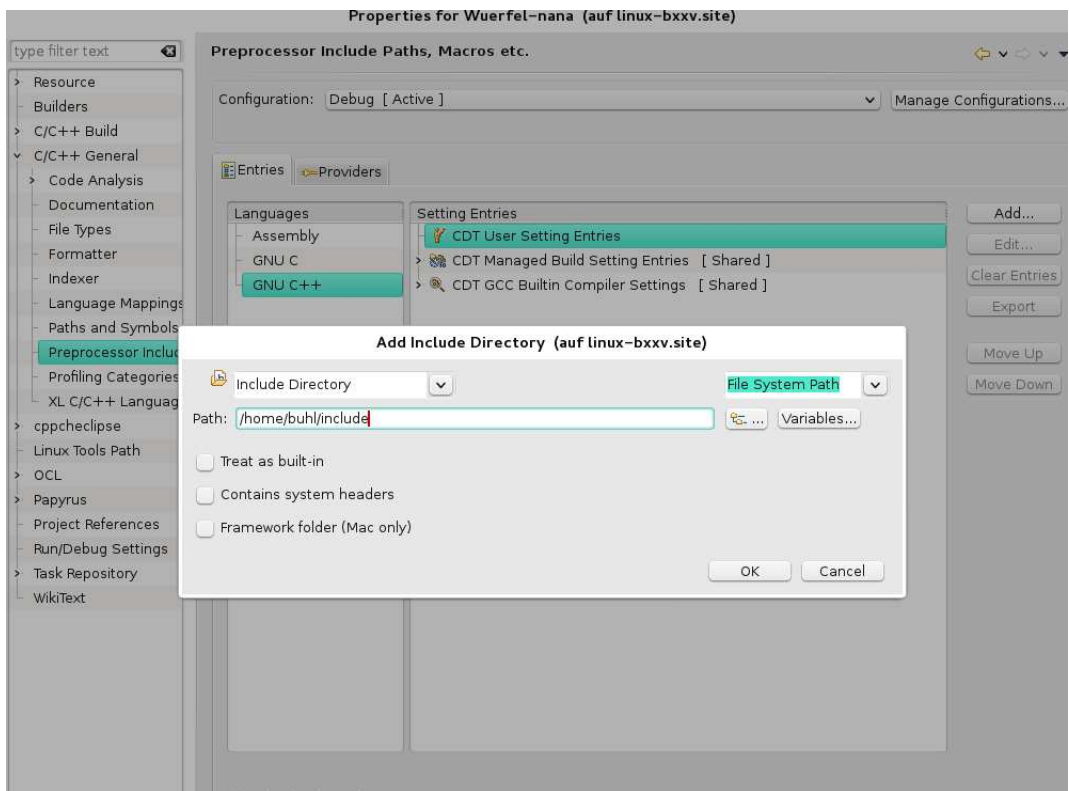
nana  
savannah

(Alternative nur projektweite Konfiguration des Eclipse-Editors für Doxygen-Spezialkommentare:

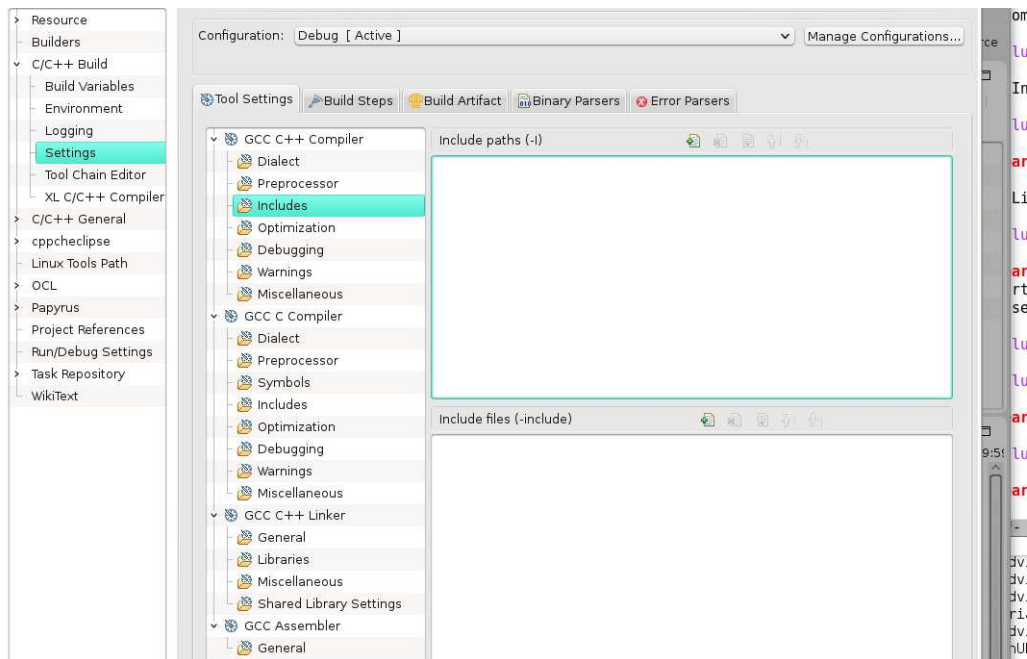


)

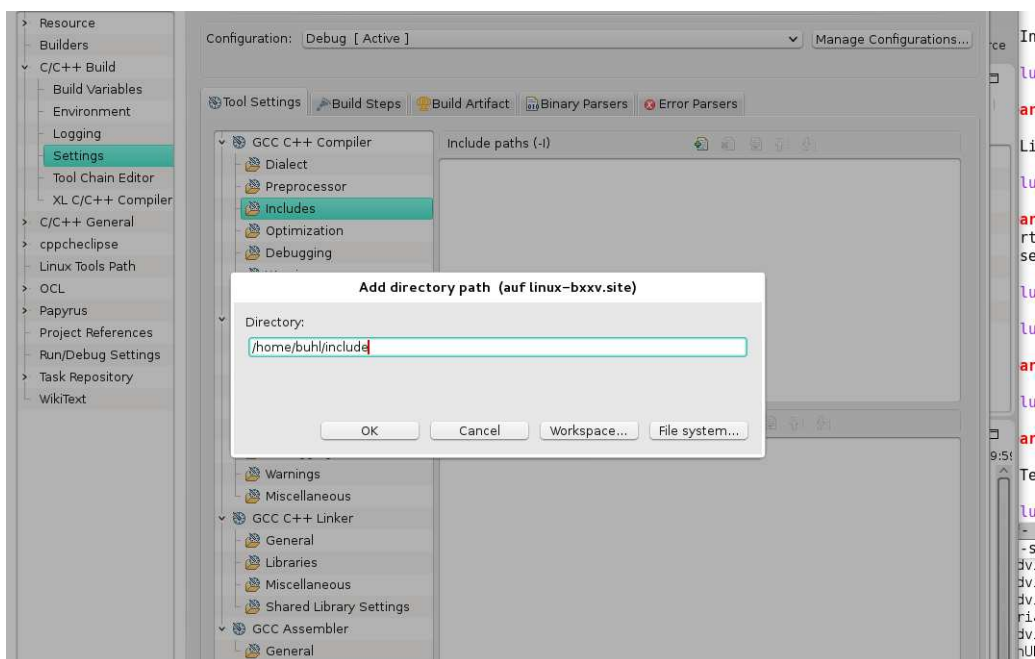
Konfiguration von eclipse zur nana-Nutzung:  
a) Include-Pfad der nana-Makros für den Eclipse-Editor:



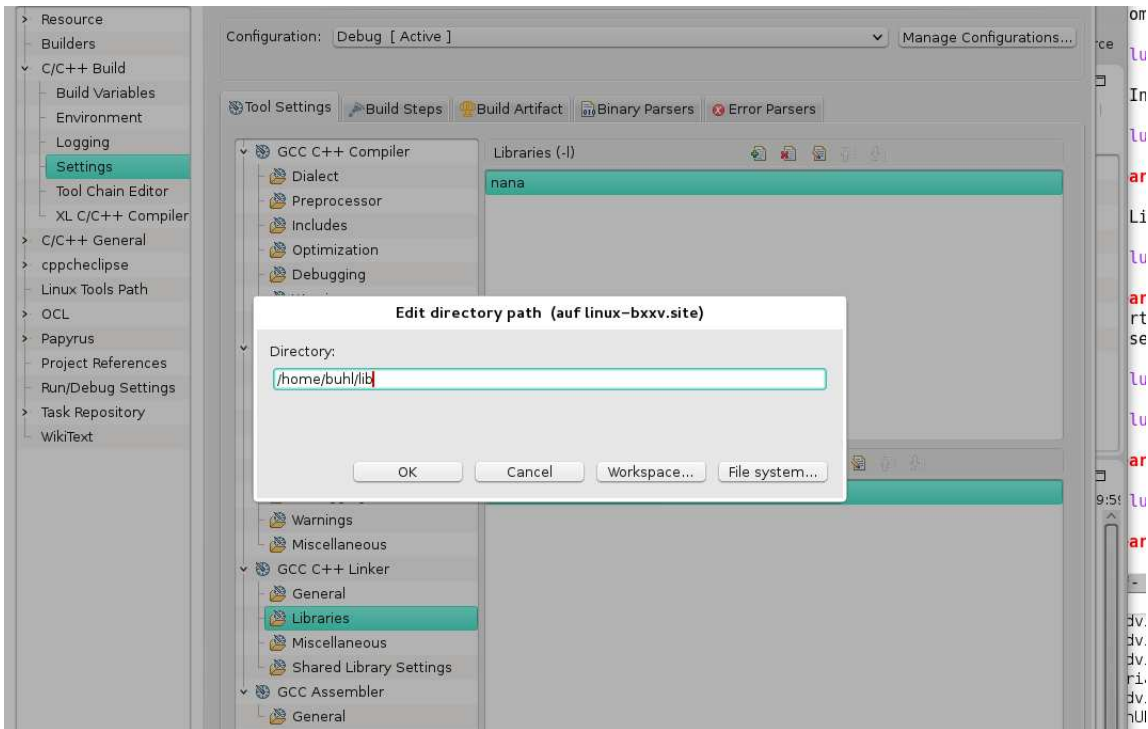
b) Compiler- und Linker-Settings:



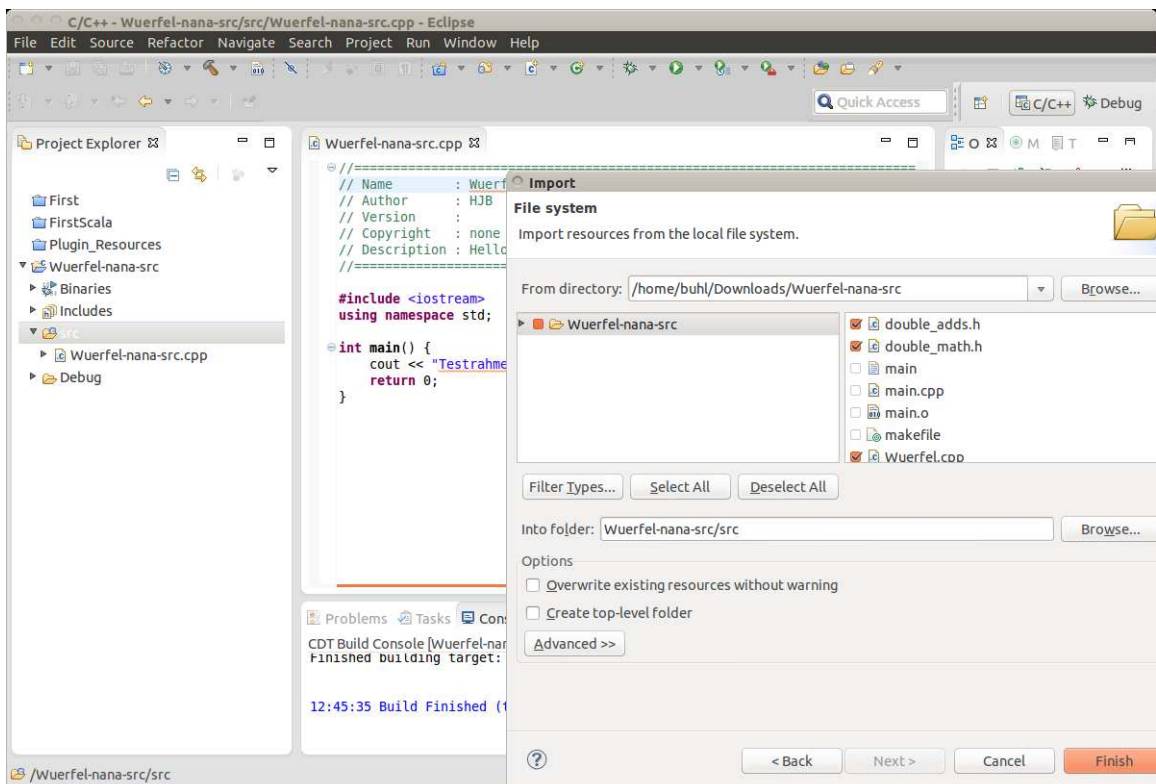
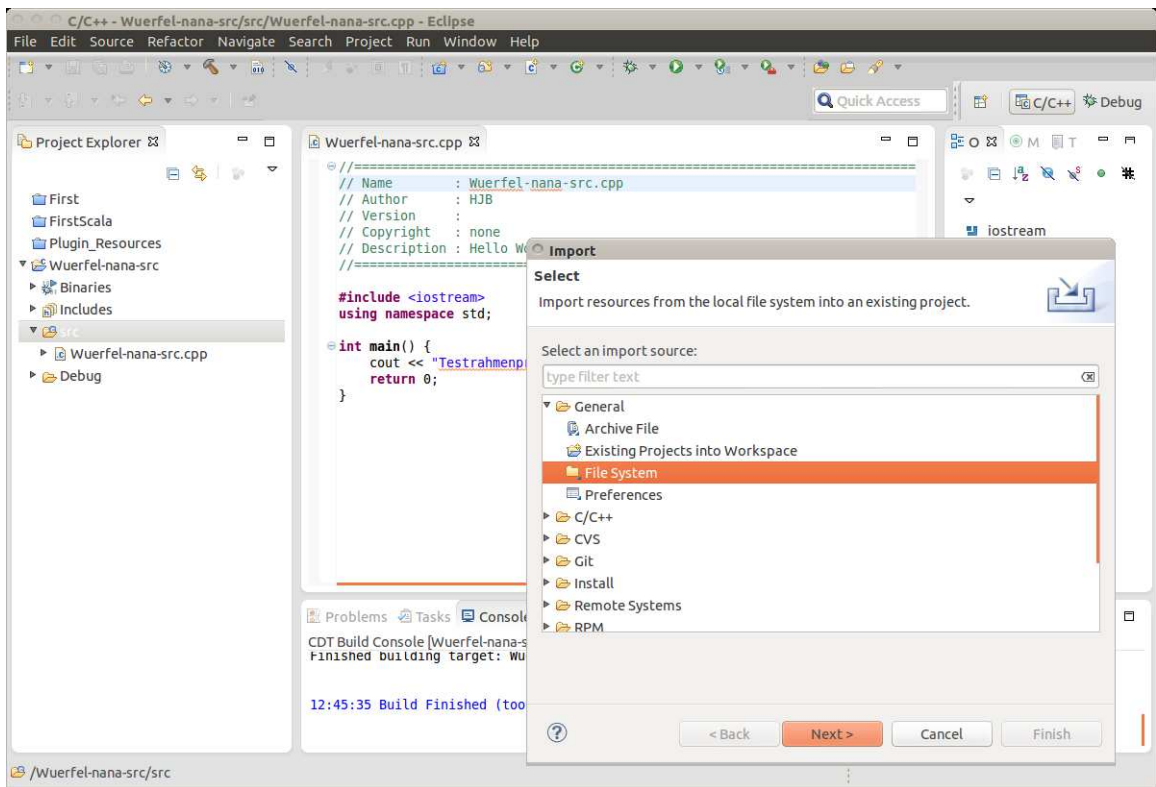
b1) Include-Path -I:



b2) Library-Pfad und -Name:



Import der externen Testklasse Wuerfel (Übungsblatt 8, Aufgabe 3) in ein Eclipse-nana-Projekt:



C/C++ - Wuerfel-nana-src/src/Wuerfel-nana-src.cpp - Eclipse

File Edit Source Refactor Navigate Search Project Run Window Help

Quick Access

C/C++ Debug

Project Explorer

- First
- FirstScala
- Plugin\_Resources
- Wuerfel-nana-src
  - Binaries
  - Includes
  - src
    - double\_adds.h
    - double\_math.h
    - Wuerfel.cpp
    - Wuerfel.h
    - Wuerfel-nana-src.cpp
    - Debug

Wuerfel-nana-src.cpp

```

//
#include <iostream>
#include <stdlib>
#include <string>
#include <cassert>
#include "Wuerfel.h"
#include "double_math.h"

int main(int argc, char* argv[]){
    std::cout << "Testrahmenprogramm fuer Wuerfel:" << std::endl;

    Wuerfel w(2.5);
    // assert( w.toString() == "1;6;1;1.73205" );

    std::cout << "Oberflaeche: " << w.Oberflaeche() << std::endl;

    std::cout << "w.toString()= " << w.toString() << std::endl;

    std::cout << "sqrt(3.0) = " << myown_sqrt(3.0) << std::endl;

    return 0;
}

```

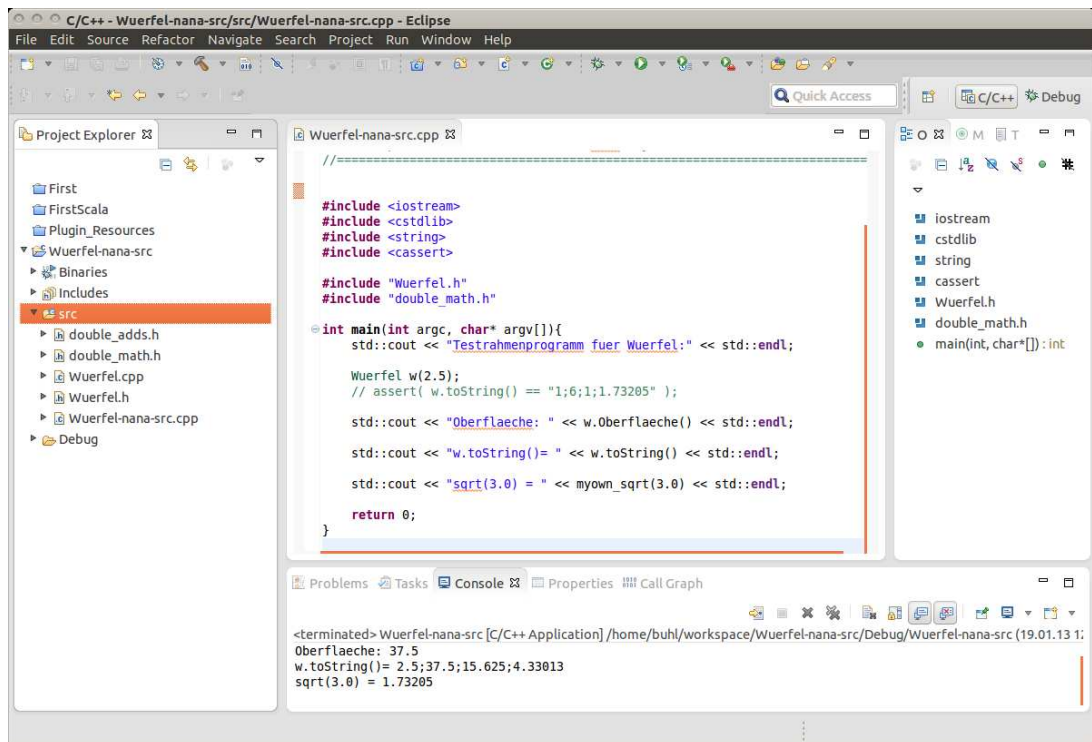
Problems Tasks Console Properties Call Graph

CDT Build Console [Wuerfel-nana-src]  
 Finished building target: Wuerfel-nana-src

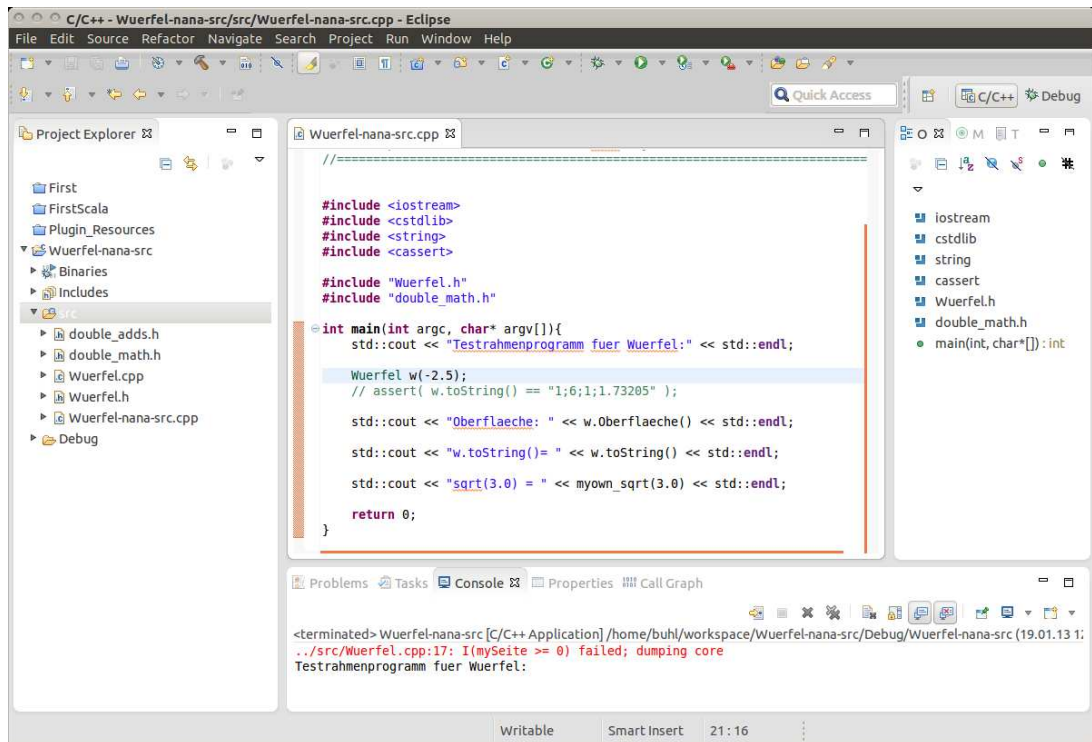
12:45:35 Build Finished (took 314ms)

Writable Smart Insert 32:1

... Testlauf:

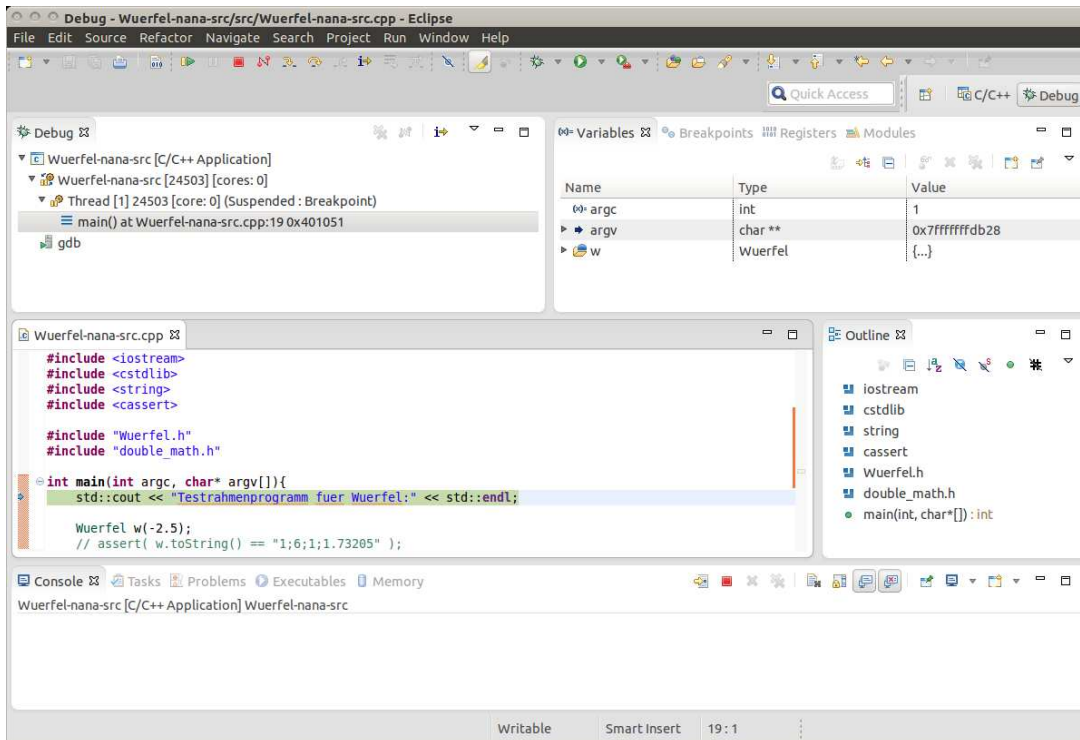


... und erste (provozierte) Vertragsverletzungen: Vorbedingung nicht erfüllt

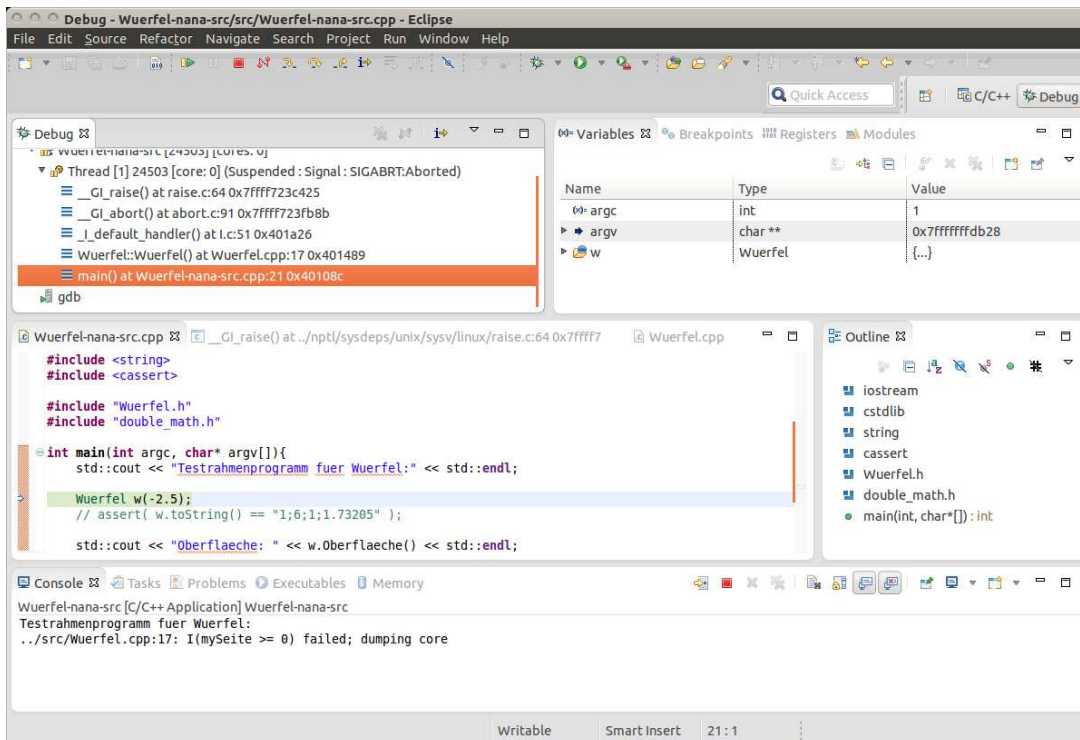




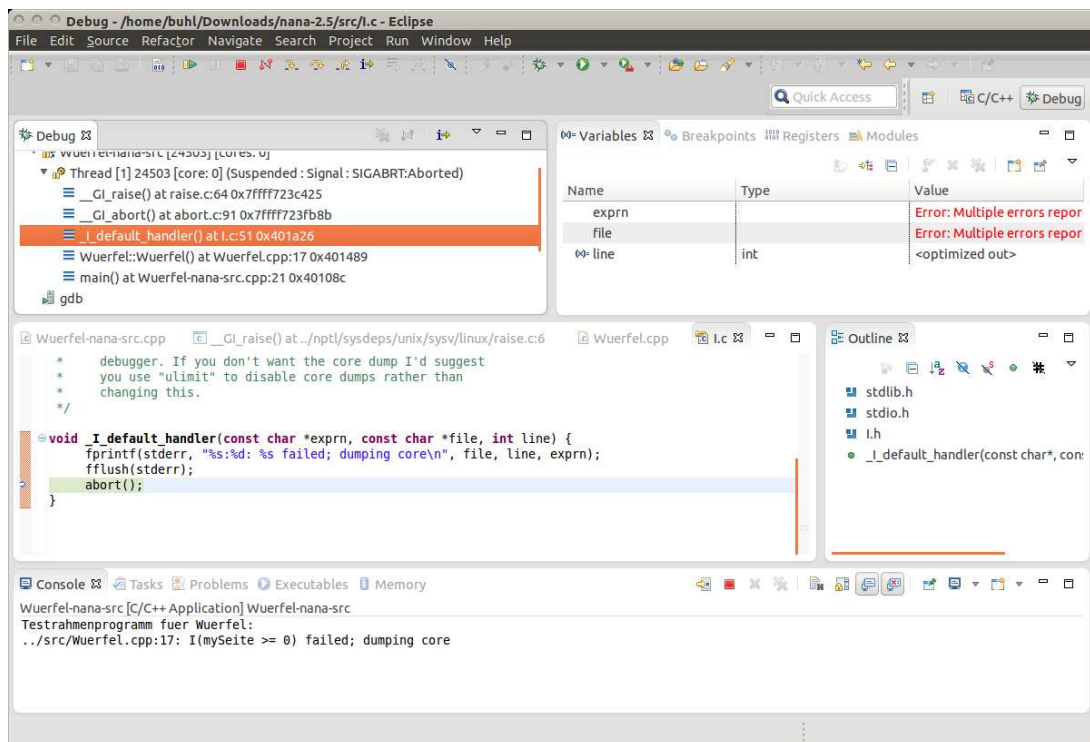
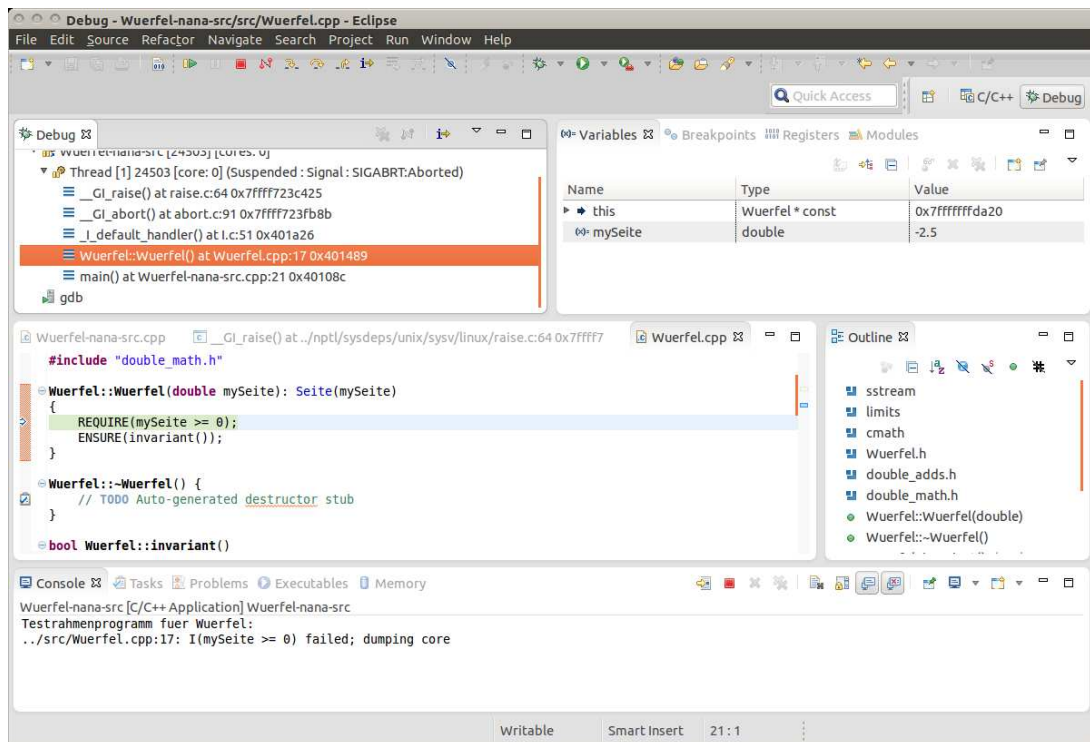
... Debug-Lauf: Start mit Breakpoint in erster Zeile des Testrahmenprogramms



... Programmabbruch nach erster Vertragsverletzung (Vorbedingung nicht erfüllt)



... Analyse der Abbruchstelle (Stacktrace mit Betrachtung der Variableninhalte zum Abbruchzeitpunkt)



step over(F6), step into(F5), Step Return(F7), ...

## 1.29. Nachbedingungen mit Gleitkommawerten: absolute oder relative Abweichung

### 1.29.1. Klasse Wuerfel

```
/*
 * Wuerfel.h
 *
 * Created on: 22.04.2009
 * Author: buhl
 */
#ifndef WUERFEL_H_
#define WUERFEL_H_
#define EIFFEL_DOEND
#ifndef EIFFEL_CHECK
#define EIFFEL_CHECK CHECK_ALL
//
// CHECK_LOOP Makros CHECK() und folgende
// CHECK_INVARIANT Makros INVARIANT() und folgende
// CHECK_ENSURE Methode invariant() und folgende
// CHECK_REQUIRE Nachbedingungen und folgende
// CHECK_NO Vorgedingungen
#endif
#include <eiffel.h>
#include <nana.h>
#include <string>
class Wuerfel{
public:
    Wuerfel(double mySeite = 1.0);
    virtual ~Wuerfel();
    virtual bool invariant();

    std::string toString();

    double Oberflaeche();

    double Volumen();

    double Raumdiagonale();
protected:
    double Seite;
};
#endif /* WUERFEL_H_ */
```

## 1.29.2. Contract der Klasse Wuerfel

```
/*
 * Wuerfel.cpp
 *
 * Created on: 22.04.2009
 * Author: buhl
 */
#include <sstream>
#include <limits>
#include <cmath>
#include "Wuerfel.h"
#include "double_adds.h"
#include "double_math.h"

Wuerfel::Wuerfel(double mySeite): Seite{mySeite}
{
    REQUIRE(mySeite >= 0);
    ENSURE(invariant());
}

Wuerfel::~Wuerfel() {
    // TODO Auto-generated destructor stub
}

bool Wuerfel::invariant()
{
    return Seite >= 0.0;
}

std::string Wuerfel::toString()
{
    std::ostringstream help{};
    help << Seite << ";" << Oberflaeche() << ";" << Volumen() << ";"
        << Raumdiagonale();
    return help.str();
}

double Wuerfel::Oberflaeche()
DO
    double result {6.0 * pow(Seite, 2)};
    ENSURE(result == 6.0 * Seite * Seite);
    ENSURE(invariant());
    return result;
}

double Wuerfel::Volumen()
DO
    double result = pow(Seite,3);
    ENSURE(withinEpsilonOf(result, Seite*Seite*Seite, 1.0E-6));
    ENSURE(invariant());
    return result;
}

double Wuerfel::Raumdiagonale()
DO
    double result {Seite * sqrt(3.0)};
    ENSURE(approximatelyEqualTo(result, Seite * myown_sqrt(3.0), 1.0));
    ENSURE(invariant());
    return(result);
}

```

Naiver Wertevergleich, absolute Abweichung und relative Abweichung:  
Diskutieren Sie den sinnvollen Einsatz!

[http://msdn.microsoft.com/en-us/library/6x7575x3\(VS.80\).aspx](http://msdn.microsoft.com/en-us/library/6x7575x3(VS.80).aspx)  
<http://www.roguewave.com/portals/0/products/legacy-hpp/docs/stdref/numeric-limits.html>  
What Every Computer Scientist Should Know About Floating  
<http://www.cplusplus.com/reference/clibrary/cfloat/>  
<http://realtimecollisiondetection.net/blog/?p=89>

## 1.29.3. double\_adds.h

```
/*
 * double_adds.h
 *
 * Created on: 13.06.2009
 * Author: buhl
 */

#ifndef DOUBLE_ADDS_H_
#define DOUBLE_ADDS_H_

#define EIFFEL_DOEND
#ifndef EIFFEL_CHECK
#define EIFFEL_CHECK CHECK_ALL
//
// CHECK_LOOP Makros CHECK() und folgende
// CHECK_INVARIANT Makros INVARIANT() und folgende
// CHECK_ENSURE Methode invariant() und folgende
// CHECK_REQUIRE Nachbedingungen und folgende
// CHECK_NO Vorgedingungen
#endif
#include <eiffel.h>
#include <nana.h>

#include <limits>
#include <cmath>
#include <algorithm>

[[maybe_unused]]static bool withinEpsilonOf(double left, double right, double delta)
{
    return fabs(left - right) <= delta;
}

static bool approximatelyEqualTo(double left, double right, double factor)
{
    return fabs(left - right) <= std::numeric_limits<double>::epsilon( ) * factor *
        std::max(fabs(left), fabs(right));
}

#endif /* DOUBLE_ADDS_H_ */
```

## 1.29.4. double\_math.h

Alternativ-Implementierungen für Nachbedingungen:

```
/*
 * double_math.h
 *
 * Created on: 13.06.2009
 * Author: buhl
 */

#ifndef DOUBLE_MATH_H_
#define DOUBLE_MATH_H_

#define EIFFEL_DOEND
#ifndef EIFFEL_CHECK
#define EIFFEL_CHECK CHECK_ALL
//
// CHECK_LOOP Makros CHECK() und folgende
// CHECK_INVARIANT Makros INVARIANT() und folgende
// CHECK_ENSURE Methode invariant() und folgende
// CHECK_REQUIRE Nachbedingungen und folgende
// CHECK_NO Vorgedingungen
#endif
#include <eiffel.h>
#include <nana.h>

#include "double_adds.h"

#include <limits>
#include <cmath>
#include <algorithm>

static double myown_sqrt(double x)
{
    double xold{}, xi{}, result{};
    REQUIRE(x >= 0.0);
    if (x == 0.0) {
        result = 0.0;
    } else {
        xi = x / 2.0;
        do{
            xold = xi;
            xi = 0.5 * (xi + x/xi);
        } while (xi != xold);
        result = xi;
    };
    ENSURE(approximatelyEqualTo(result*result, x, 1.0));
    return result;
}

#endif /* DOUBLE_MATH_H_ */
```

## 1.30. Vererbung und Codeverträge

Es gelten folgende Regeln bei der Vererbung (von is-a-Methoden):

- a) Vorbedingungen können in einer Kindklasse abgeschwächt werden.
- b) Nachbedingungen in einer Kindklasse müssen stärker (oder gleich) sein als diejenigen der Elterklasse.
- c) Invarianten in der Kindklasse müssen ebenfalls stärker (oder gleich) als in der Elterklasse sein.

Dann ist ein echtes *Subcontracting* realisiert.

Bemerkung: Es reicht die Kindnachbedingung im Falle des Eintreffens der Eltervorbedingung stärker als die Elternnachbedingung zu realisieren. Im Falle „Kindvorbedingung **and not** Eltervorbedingung“ darf die Kindnachbedingung frei gewählt werden.

Das ist unbedingt zu beachten, will man in Nachkommen eine benutzerfreundlichere Methodenvariante mit allgemeineren Vorbedingungen realisieren, z. B.:

```
class name_list{
...
public:

    //////////// basic queries:

    unsigned int get_count() const; // number of items in stack
    bool has(const string& a_name) const;
    ...
    //////////// (pure) modifiers:

    virtual void put(const string& a_name); // Push a_name into list
}

void name_list::put(const string& a_name) // Push a_name into list
{
    REQUIRE(invariant());
    REQUIRE(/* name not in list */ !has(a_name));
    ID(unsigned int count_old = get_count());
    ...
    ENSURE(has(a_name));
    ENSURE(get_count() == count_old + 1);
    ENSURE(invariant());
}
...
```

```

...
////////// child class relaxed_name_list //////////
////////// (more user friendly) //////////

class relaxed_name_list : public name_list{
    ////////// (pure) modifiers: (redefined)

    virtual void put(const string& a_name); // Push a_name into list
}
...
void relaxed_name_list::put(const string& a_name) // Push a_name into list
{
    REQUIRE(invariant());
    REQUIRE(/* nothing */ true); // usable without conditions
    ID(unsigned int count_old = get_count());
...
    ENSURE(has(a_name));
    ENSURE(get_count() == count_old + 1); // falls die VB des Vaters gilt
    ENSURE(get_count() == count_old); // sonst
    ENSURE(invariant());
}

```

Wie kann die fallweise-Definition der Nachbedingung der benutzerfreundlichen put()-Version realisiert werden?

```

    ID(bool not_in_list = !has(a_name));
...
    ENSURE((!not_in_list) || (get_count() == count_old + 1)); // &&
    ENSURE(not_in_list || (get_count() == count_old));

```

Lesen sollte man

```
(!not_in_list) || (get_count() == count_old + 1)
```

wie

```
(not_in_list) implies (get_count() == count_old + 1)
```

Mit Hilfe der Hilfsfunktion

```

template <class T>
set<T> operator+(const set<T>& s, const T& e){
    set<T> result(s);
    result.insert(e);
    return result;
}

```

(Mengenvereinigung mit einem Element) kann schließlich die Nachbedingung „alle anderen Elemente der Liste bleiben unverändert in ihr enthalten“ (keine Verschlimmbesserung) folgendermaßen realisiert werden:

```

    ID(set<string> contents_old(begin(),end()));
...
    ID(set<string> contents(begin(),end()));
    ENSURE(not_in_list || (contents == contents_old));
    ENSURE((!not_in_list) || (contents == contents_old + a_name));

```



Subcontracting in nana:

```
class name_list{
...
public:

    //////////// basic queries:

    unsigned int get_count() const;    // number of items in stack

    bool has(const string& a_name) const;
...
    //////////// (pure) modifiers:

    virtual void put(const string& a_name); // Push a_name into list
}

void name_list::put(const string& a_name)    // Push a_name into list
DO
    REQUIRE(/* name not in list */    !has(a_name));
    ID(set<string> contents_old(begin(),end()));
    ID(int count_old = get_count());
    ID(bool not_in_list = !has(a_name));
...
    ENSURE(has(a_name));
    ENSURE( (!not_in_list) || (get_count() == count_old + 1));
    ID(set<string> contents(begin(),end()));
    ENSURE( (!not_in_list) || (contents == contents_old + a_name));
END
...
////////// child class relaxed_name_list ////////////
////////// (more user friendly) ////////////

class relaxed_name_list : public name_list{
    //////////// (pure) modifiers: (redefined)

    virtual void put(const string& a_name);    // Push a_name into list
...
}
void relaxed_name_list::put(const string& a_name)    // Push a_name into list
DO
    REQUIRE(/* nothing */ true);    // usable without conditions
    ID(set<string> contents_old(begin(),end()));
    ID(int count_old = get_count());
    ID(bool not_in_list = !has(a_name));
...
    ENSURE(has(a_name));
    ENSURE((!not_in_list) || (get_count() == count_old + 1)); // &&
    ENSURE( not_in_list || (get_count() == count_old));
    ID(set<string> contents(begin(),end()));
    ENSURE( not_in_list || (contents == contents_old));
    ENSURE((!not_in_list) || (contents == contents_old + a_name));
END
```

## Forschungsministerium fördert Standard für IT-Sicherheit

Trotz des flächendeckenden Einsatzes von Computersystemen in sicherheitsrelevanten Bereichen fehlt bislang eine standardisierte Methode, die das fehlerfreie Funktionieren solcher Systeme garantiert. Das Bundesministerium für Bildung und Forschung (BMBF) will nun Arbeiten fördern, bei denen mit Methoden der Verifikation der so genannte geschlossene integrierte Korrektheitsbeweis erbracht werden kann. Damit sollen sich Fehler bereits im Entwurf von autonomen oder integrierten Computersystemen erkennen und korrigieren lassen – eine sorgfältige Spezifikation vorausgesetzt. Alle möglichen Fehlersituationen könnten aber nur dann abgefangen werden, wenn bereits in der Planung die entsprechenden Einsatzszenarien definiert wurden, betonte Projektleiter Prof. Dr. Wolfgang Paul gegenüber heise Security.

Für die erste zweijährige Forschungsphase werde das BMBF 7,2 Millionen Euro zur Verfügung stellen, teilte das Ministerium am heutigen Mittwoch in Berlin mit. An dem Projekt beteiligen sich neben der Universität Saarland unter anderen auch die TUs Darmstadt, Karlsruhe, München sowie Infineon, T-Systems und BMW.

Die Entwicklung eines integrierten Korrektheitsbeweises gilt zurzeit als eine der größten Herausforderungen der Informatik. Er soll die Funktionen bei der Entwicklung von Hard- und Systemsoftware bis zur Netzwerk- und Anwendungsebene laufend überprüfen. Zunächst sollen die mathematischen Grundlagen entwickelt, vollständig formalisiert und für Informatikanwendungen in den Bereichen Embedded Systems, Kommunikation und Anwendungssoftware erschlossen werden. Darauf aufbauend sollen die Projektpartner Demonstratoren entwickeln und mit ihnen Computersysteme für Chipkarten, Telekommunikation und Automobilelektronik von der Hardware bis zur Anwendungssoftware überprüfen. Im Rahmen des Projektes werden auch Softwaretools entwickelt, die den Verifikationsprozess unterstützen. (dab/c't)

aus: <http://www.heise.de/newsticker/data/dab-01.10.03-002/>

Mehr als jedes zehnte Unternehmen in Deutschland hat Probleme mit der Sicherheit seiner Informationssysteme. Das geht aus einer Mitteilung[1] des Statistischen Bundesamtes anlässlich des 5. Nationalen IT-Gipfels[2] am morgigen Dienstag in Dresden hervor. Danach gaben 74 Prozent der betroffenen Unternehmen an, dass bei ihnen 2009 aufgrund von Hardware- oder Softwarefehlern Daten zerstört oder verändert wurden und bestimmte Dienste ihrer Informations- und Kommunikationssysteme nicht verfügbar waren.

28 Prozent der Unternehmen hatten Probleme, weil Schadsoftware oder nicht autorisierte Zugriffe zur Veränderung beziehungsweise Zerstörung von Daten führten. Phishing-Angriffe störten bei 3 Prozent der betroffenen Firmen die Systeme. Stärker fiel der unbedachte Umgang der Belegschaft mit vertraulichen Daten ins Gewicht: In 11 Prozent der Firmen legten Mitarbeiter vertrauliche Daten offen. Deshalb führen laut Erhebung inzwischen 25 Prozent der Unternehmen mit zehn und mehr Beschäftigten obligatorische Schulungen der Mitarbeiter zum Thema IT-Sicherheit durch.

Die Zahlen hat das Statistische Bundesamt im Rahmen seiner jährlichen Erhebung zur Nutzung von Informations- und Kommunikationstechnik in Unternehmen ermittelt.

aus: [Jedes zehnte Unternehmen hat IT-Sicherheitsprobleme 06.12.2010 12:24](#)

## 1.31. beA-Sicherheitsmängel

beA: Schwere Panne beim besonderen elektronischen Anwaltspostfach

34C3: Das besondere Anwaltspostfach beA als besondere Stümperei

Fataler Konstruktionsfehler im besonderen elektronischen Anwaltspostfach

Sie haben keine Post! Elektronisches Anwaltspostfach wird zum Debakel

Besonderes elektronisches Anwaltspostfach: Deutscher Anwaltverein fragt: beA - Wie geht es weiter?

beA: Das Chaos nimmt zu

Atos hält die eigene Lösung für sicher

Zur Sicherheit sollen Rechtsanwälte die beA Client Security deaktivieren

## 1.32. Biometrische Merkmale zum Authentifizieren

iPhone X – Face ID

Mehrere Lücken in Lenovos biometrischem Passwort-Manager

...

## 1.33. Vertragsverletzungen zur Laufzeit (Fortsetzung)

### 1.33.1. Vertragsverletzungen eines Eiffel-Programms in EiffelStudio:

The screenshot shows the EiffelStudio IDE with the following components:

- Editor:** Displays the source code for the `myown_sqrt` function in the `MAIN` class. The code includes a `require` clause for `DOUBLE_ADDS` with the condition `argument_nonnegative: x >= 0.0`. The function body calculates the square root using an iterative Newton-Raphson method. At the end, an `ensure` clause is present: `ensure -- from DOUBLE_ADDS approximatelyequalto (Result * Result, x, 0.5)`.
- Right Panel:** Shows the execution status: `Status = Implicit exception pending` and `Postcondition violated.` Below this is a stack frame table with columns 'In Feature' and 'In Class', showing the current frame is `myown_sqrt` in `MAIN`.
- Objects Panel:** A table showing the state of local variables and the `Result` object at the time of the violation.

Name	Value	Type
Locals		
xi	3,1622776601683791	REAL_64
xold	3,1622776601683791	REAL_64
Result		
Result	3,1622776601683791	REAL_64
- Watch Panel:** A table showing the evaluation of arithmetic expressions.

Expression	Value	Type
Result*Result	9,9999999999999982	REAL_64
(Result*Result-1...	-1,7763568394002506e-16	REAL_64
- Status Bar:** Displays the message `Implicit exception pending: Postcondition violated.` and the current file `cout_test` at `45:25`.

Anzeige der Art der Exception: Postcondition violated

Stackframe zum Zeitpunkt der Vertragsverletzung

Attributwerte zum Zeitpunkt der Vertragsverletzung

Berechnung von arithmetischen Ausdrücken unter Benutzung der Attributwerte möglich im Watch-Fenster

## 1.33.2. Vertragsverletzung bei CLI-Debuglauf: backtrace

Eine Modifikation des Programms `main()` zum Abfangen des Signals `SIGABRT` kann dessen Verhalten bei Vertragsverletzung beeinflussen. Prinzipielles Vorgehen dazu:

```
...
#include <csignal>
...
void ABRT_signalhandler(int signum)
{
    std::cerr << "modifizierter SIGABRT-Handler" << std::endl;
    signal(SIGABRT, SIG_DFL);
    abort();
}

int main(int argc, char* argv[]) {
    signal(SIGABRT, ABRT_signalhandler);
    ...
}
```

Benutzt wird nun `execinfo.h` und die Funktion `backtrace_symbols` (vgl. [http://www.gnu.org/software/libc/manual/html\\_node/Backtraces.html](http://www.gnu.org/software/libc/manual/html_node/Backtraces.html)), um bei Vertragsverletzung automatisch den `backtrace` auf `cout` zu schreiben, ohne einen Debugger bemühen zu müssen:

```
/* main.cpp */
* Created on: 22.04.2009 */

#include <iostream>
#include <cstdlib>
#include <string>
#include <cassert>
#include <csignal>
#include <execinfo.h>

#include "Wuerfel.h"
#include "double_math.h"

class ExceptionTracer
{
public:
    ExceptionTracer()
    {
        void * array[25];
        int nSize = backtrace(array, 25);
        char ** symbols = backtrace_symbols(array, nSize);

        for (int i = 0; i < nSize; i++)
        {
            std::cout << symbols[i] << std::endl;
        }

        free(symbols);
    }
};

void ABRT_signalhandler(int signum)
{
    std::cout << "SIGABRT: backtrace is " << std::endl;
    ExceptionTracer::ExceptionTracer();
    std::cout << "resuming abort " << std::endl << std::endl;
    signal(SIGABRT, SIG_DFL);
    abort();
}

int main(int argc, char* argv[]) {
    signal(SIGABRT, ABRT_signalhandler);

    std::cout << "Testrahmenprogramm fuer Wuerfel:" << std::endl;
    Wuerfel w(2.5);
}
```

```

// assert( w.toString() == "1;6;1;1.73205" );

std::cout << "Oberflaeche: " << w.Oberflaeche() << std::endl;
std::cout << "w.toString()=" << w.toString() << std::endl;
std::cout << "sqrt(3.0) = " << myown_sqrt(3.0) << std::endl;

// ...

return 0;
}

```

Vergleiche zu ExceptionTracer: <http://www.ibm.com/developerworks/linux/library/l-cppexcep.html>

Jetzt wird nach der nana-Fehlermeldung automatisch der backtrace angezeigt:

```

./main
...
double_math.h:44: I(approximatelyEqualTo(result*result, x, 0.5)) failed;
dumping core
SIGABRT: backtrace is
./main [0x401447]
./main [0x401132]
/lib64/libc.so.6 [0x7f3d880256e0]
/lib64/libc.so.6(gsignal+0x35) [0x7f3d88025645]
/lib64/libc.so.6(abort+0x183) [0x7f3d88026c33]
./main [0x401c80]
./main [0x401776]
./main [0x4017f6]
./main [0x401abb]
./main [0x4012fb]
/lib64/libc.so.6(__libc_start_main+0xe6) [0x7f3d88011586]
./main [0x400f19]
resuming abort
Abgebrochen

```

Leider enthält das main()-Binary nur genügend Symbolinformationen, wenn Sie g++ mit der Option `-rdynamic` übersetzt haben:

```

./main
...
double_math.h:44: I(approximatelyEqualTo(result*result, x, 0.5)) failed;
dumping core
SIGABRT: backtrace is
./main(_ZN15ExceptionTracerC1Ev+0x23) [0x401af7]
./main(_Z18ABRT_signalhandleri+0x30) [0x4017e2]
/lib64/libc.so.6 [0x7f4ca97926e0]
/lib64/libc.so.6(gsignal+0x35) [0x7f4ca9792645]
/lib64/libc.so.6(abort+0x183) [0x7f4ca9793c33]
./main(__libc_csu_fini+0) [0x402330]
./main [0x401e26]
./main(_ZN7Wuerfel13RaumdiagonaleEv+0x70) [0x401ea6]
./main(_ZN7Wuerfel8toStringEv+0x39) [0x40216b]
./main(main+0x92) [0x4019ab]
/lib64/libc.so.6(__libc_start_main+0xe6) [0x7f4ca977e586]
./main [0x4015c9]
resuming abort
Abgebrochen

```

Filtern Sie schließlich die Ausgaben von `main()` durch den Demangler `c++filt`, so wird eine akzeptable Fehleranzeige erreicht:

```
./main | c++filt
...
double_math.h:44: I(approximatelyEqualTo(result*result, x, 0.5)) failed;
dumping core
...
./main(ExceptionTracer::ExceptionTracer()+0x23) [0x401af7]
./main(ABRT_signalhandler(int)+0x30) [0x4017e2]
/lib64/libc.so.6 [0x7ff33cbcf6e0]
/lib64/libc.so.6(gsignal+0x35) [0x7ff33cbcf645]
/lib64/libc.so.6(abort+0x183) [0x7ff33cbd0c33]
./main(_libc_csu_fini+0) [0x402330]
./main [0x401e26]
./main(Wuerfel::Raumdiagonale()+0x70) [0x401ea6]
./main(Wuerfel::toString()+0x39) [0x40216b]
./main(main+0x92) [0x4019ab]
/lib64/libc.so.6(_libc_start_main+0xe6) [0x7ff33cbbb586]
./main [0x4015c9]
```

([http://en.wikipedia.org/wiki/Name\\_mangling#Standardised\\_name\\_mangling\\_in\\_C.2B.2B](http://en.wikipedia.org/wiki/Name_mangling#Standardised_name_mangling_in_C.2B.2B))

### 1.33.3. Vertragsverletzungen bei C++-Debug-Lauf in ddd

The screenshot shows the DDD (Data Display Debugger) interface. The main window title is "DDD: /home/buhl/Wuerfel-na". The menu bar includes File, Edit, View, Program, Commands, Status, Source, and Data. The main area is divided into several sections:

- Backtrace Window (Top Right):** Displays a stack trace with the following frames:
  - #7 0x08048e29 in main () at main.cpp:24
  - #6 0x080494d8 in Wuerfel::toString () at Wuerfel.cpp:33
  - #5 0x08049271 in Wuerfel::Raumdiagonale () at Wuerfel.cpp:56
  - #4 0x08049208 in myown\_sqrt () at double\_math.h:44 (highlighted)
  - #3 0x0804969b in \_I\_default\_handler () at I.c:51
  - #2 0xb7d562c8 in abort () from libc.so.6
  - #1 0xb7d54990 in raise () from libc.so.6
  - #0 0xffffe430 in \_\_kernel\_vsyscall ()Buttons for "Up", "Down", "Close", and "Help" are visible below the list.
- Source Code Window (Middle):** Shows C++ code with a green arrow pointing to the current execution point:

```
        result = xi;
    };
    ENSURE(approximatelyEqualTo(result*result, x, 0.5));
    return result;
}

#endif /* DOUBLE_MATH_H_ */
```
- Output Window (Bottom):** Contains the following text:

```
Copyright © 1999–2001 Universität Passau, Germany.
Copyright © 2001 Universität des Saarlandes, Germany.
Copyright © 2001–2004 Free Software Foundation, Inc.
(gdb) run
Testrahmenprogramm fuer Wuerfel:
Oberflaeche: 37.5
double_math.h:44: I(approximatelyEqualTo(result*result, x, 0.5)) failed; dumping core

Program received signal SIGABRT, Aborted.
0xffffe430 in __kernel_vsyscall ()
(gdb) frame 4
#4 0x08049208 in myown_sqrt (x=3) at double_math.h:44
(gdb) |

^ #4 0x08049208 in myown_sqrt (x=3) at double_math.h:44
```



## 1.33.4. Automatischer Start von ddd beim CLI-Debug-Lauf und Vertragsverletzung

Um bei einer Vertragsverletzung auch ohne core-Dump automatisch direkt in den Debugger ddd zu wechseln, um dort die Abbruchstelle zu untersuchen (Variableninhalte, Ausdrücke in Variableninhalten zum Abbruchzeitpunkt, ...) modifiziert man den ABRT-Signalhandler folgendermaßen:

```
#include <csignal>
#include <sys/types.h>
#include <unistd.h>
#include <sstream>
...
void exec_ddd();
void ABRT_signalhandler(int signum)
{
    std::cerr << "SIGABRT: starting ddd... " << std::endl;
    exec_ddd();
    std::cerr << "resuming abort " << std::endl << std::endl;
    signal(SIGABRT, SIG_DFL);
    abort();
}

static int ddd_process_pid = 0;
void exec_ddd()
{
    // Create child for running ddd
    int pid = fork();

    if (pid < 0) /* error */
    {
        abort();
    }
    else if (pid) /* parent */
    {
        // C++ nana-application
        ddd_process_pid = pid; // save debugger pid
        sleep(10); // give ddd time to attach
        // Continue C++ nana-application
    }
    else /* child */
    {
        // ddd process
        std::stringstream args;
        args << "--pid=" << getpid();
        execl("/usr/bin/ddd",
            "ddd", "--debugger", "gdb",
            args.str().c_str(), (char *) 0);

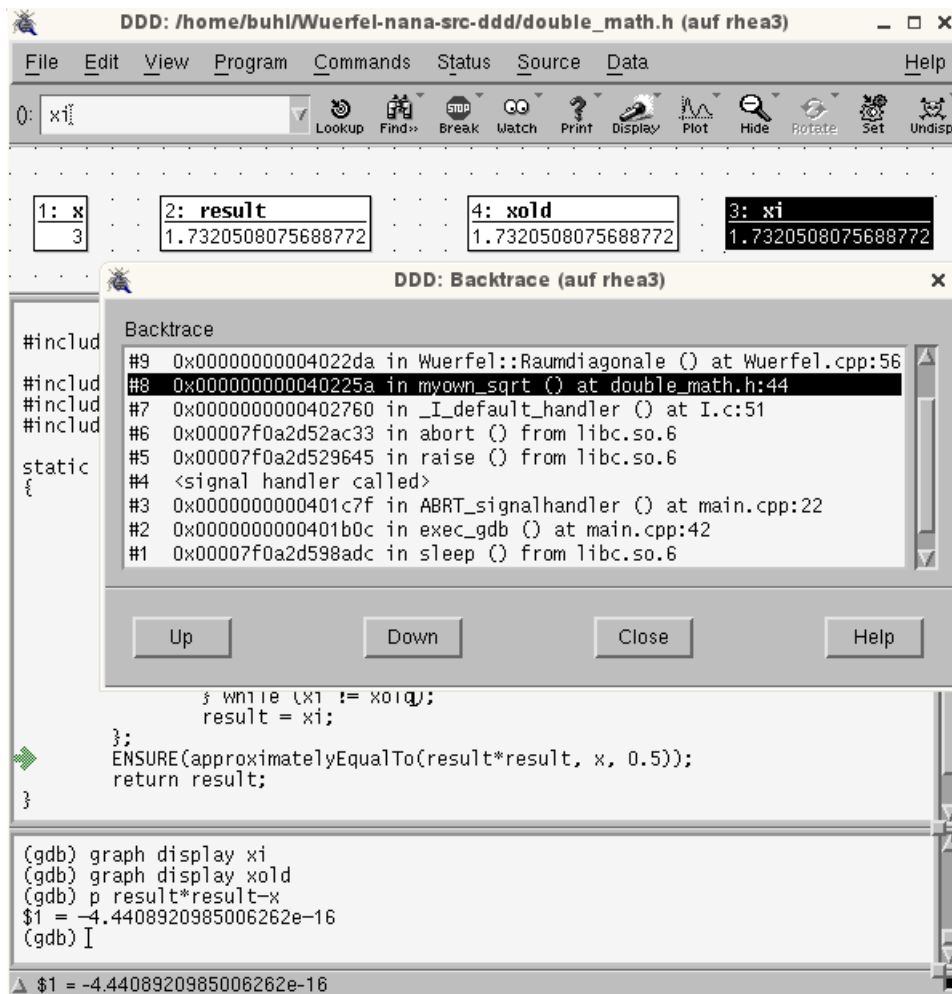
        std::cerr << "\nFailed to exec ddd\n" << std::endl;
    }
}

int main(int argc, char* argv[]){
    signal(SIGABRT, ABRT_signalhandler);

    std::cout << "Testrahmenprogramm fuer Wuerfel:" << std::endl;
    ...
}

```

Vergleiche: [http://www.codeproject.com/KB/debug/java\\_cpp\\_debugging.aspx?display=Print](http://www.codeproject.com/KB/debug/java_cpp_debugging.aspx?display=Print)



Bemerkung: Diese Methode benötigt nicht das Schreiben einer core-Datei auf die Festplatte, funktioniert also auch ohne Änderung der Datei `/etc/security/limits.conf`.

### Alternativer Debugger statt ddd

ab der Luna-M7-Version wird der Eclipse-CDT-StandaloneDebugger geeigneter als der oben benutzte ddd:

The screenshot displays the Eclipse Stand-alone Debugger interface. The main window is titled "Eclipse Stand-alone Debugger" and contains several panes:

- Debug Console:** Shows the current thread as "main() at hello.c:4 0x400440" and the debugger as "gdb".
- Variables:** A table showing the current state of variables:

Name	Type	Value
argc	int	2
argv	char **	0x7fffffff2b8
- Source Code:** The source file "hello.c" is open, showing the following code:

```
#include <stdio.h>
#include "somehdr.h"

int main(int argc, char **argv) {
    int k = SOME_VALUE; /* comment */
    int j = x(k);
    int i;
    for (i = 0; i < argc; ++i) {
        printf("argv %d is %s\n", i, argv[i]);
    }
    printf ("hello world %d\n", j);
    return 0;
}
```
- Outline:** Shows the project structure with files like "stdio.h", "somehdr.h", and the current function "main(int, char\*\*) : int".
- Executables:** A table listing the executables and their source files:

Executable Name	Project	Location	Source File Name	Location
a.out	Executables	/home/cygnus/johnstn/sour	hello.c	/home/cygnus/johnstn/sources2/sample/hell
			libio.h	/usr/include/libio.h
			somehdr.h	/home/cygnus/johnstn/somehdr.h

Verfügbarmachen des Eclipse Standalone-Debuggers:

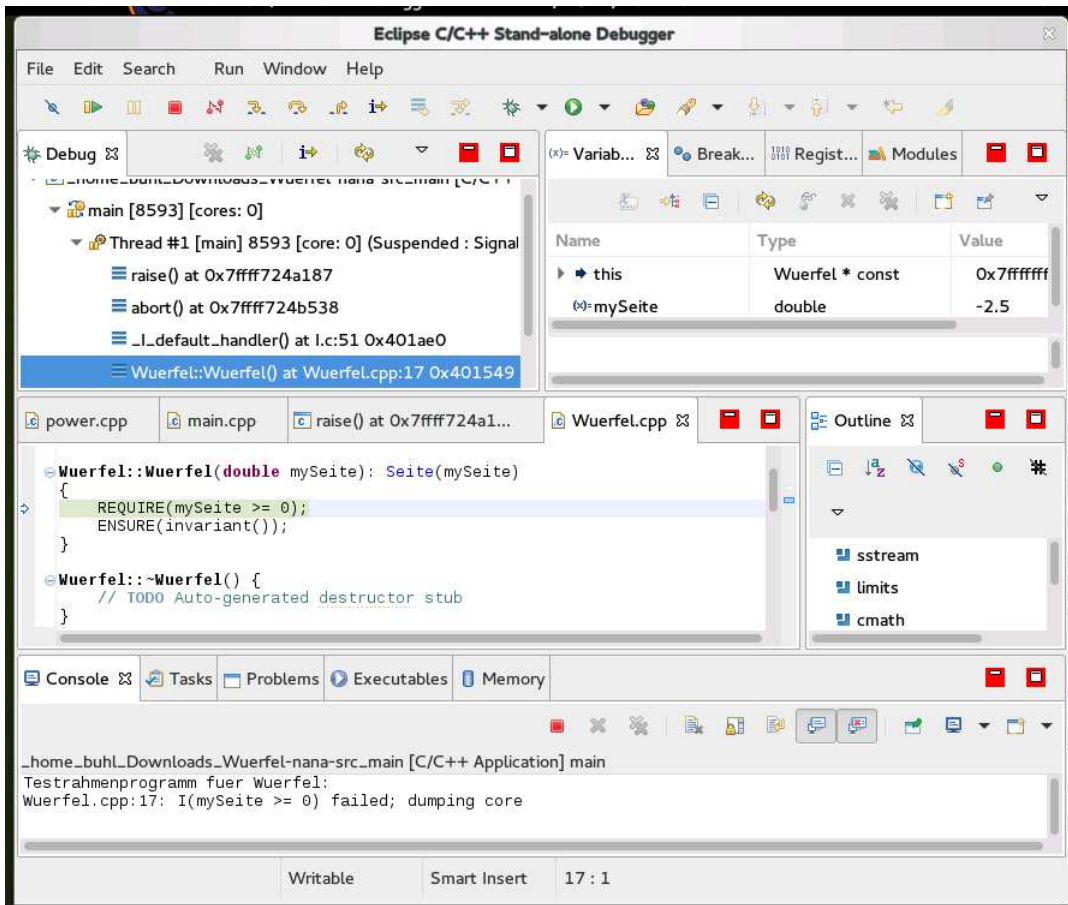
```
username@xxx> cd eclipse-modeling-neon-1a-linux-gtk-x86_64/plugins/org.eclipse.cdt.  
debug.application-1.1.0.201*/scripts  
username@xxx> ls -al  
insgesamt 36  
drwxr-xr-x 2 buhl buhl 4096 Jun 28 11:45 .  
drwxr-xr-x 6 buhl buhl 4096 Jun 28 11:45 ..  
-rw-r--r-- 1 buhl buhl 4281 Jun 7 05:01 cdtdebug.sh  
-rw-r--r-- 1 buhl buhl 4112 Jun 7 05:01 config.ini  
-rw-r--r-- 1 buhl buhl 49 Jun 7 05:01 dev.properties  
-rw-r--r-- 1 buhl buhl 2028 Jun 7 05:01 install.sh  
-rw-r--r-- 1 buhl buhl 3788 Jun 7 05:01 README  
username@xxx> sh ./install.sh  
Installation complete
```

Dabei wird in \$HOME der Arbeitsordner cdtdebugger angelegt; der einfacheren Benutzbarkeit halber ein neues Debugger-Startscript cdtdebug in \$HOME/bin anlegen:

```
#!/bin/sh  
#$HOME/cdtdebugger/cdtdebug.sh -e $*
```

Nun kann statt des ddd der standalone cdtdebugger etwa wie folgt benutzt werden:

```
username@xxx> cd ../Wuerfel-nana-src  
username@xxx> cdtdebug main
```



Oder gemäß

<https://wiki.eclipse.org/CDT/StandaloneDebugger>,

wenn neben dem Binary main ein Absturz-Speicherauszug core existiert:

```
username@xxx> cd .../Wuerfel-nana-src
username@xxx> ~/cdtdebugger/cdtdebug.sh -c core -e main
```

(Beachte auch die Option -a zur Verbindung mit einem laufenden Prozess.)

### Zusätzliche Komponenten:

- Komponenten/Unit-Tests:
  - `cute`  
siehe Abschnitt 1.2
  - `cpptest`
  - `cxctest`oder
- statische Codeanalyse
  - `cppcheck` mit eclipse ccppcheclipse-Plugin

## CLI und automatischer cdtdebug.sh-Start bei Vertragsverletzung

```
...
#include <csignal>
#include <sys/types.h>
#include <unistd.h>
#include <sstream>

#include "Wuerfel.h"
#include "double_math.h"

void exec_debugger ();
void ABRT_signalhandler (int signum)
{
    std::cerr << "SIGABRT: starting cdtdebug.sh ... " << std::endl;
    exec_debugger ();
    std::cerr << "resuming abort " << std::endl << std::endl;
    signal(SIGABRT, SIG_DFL);
    abort ();
}

static int deugger_process_pid = 0;
void exec_debugger ()
{
    // Create child for running debugger
    int pid = fork ();
    if (pid < 0) /* error */
    {
        abort ();
    }
    else if (pid) /* parent */
    {
        // C++ nana-application
        debugger_process_pid = pid; // save debugger pid
        sleep (60); // give ddd time to attach
        // Continue C++ nana-application
    }
}
```

```

else /* child */
{
    // debugger process
    std::stringstream args;

    std::string homedir = getenv("HOME");
    std::string cdtdebugger = homedir + "/cdtdebugger/cdtdebug.sh";

    args << "-a " << getpid();

    execl(cdtdebugger.c_str(),
          "cdtdebug.sh",
          args.str().c_str(), (char *) 0);

    std::cerr << "\nFailed to exec cdtdebug.sh\n" << std::endl;
}
}

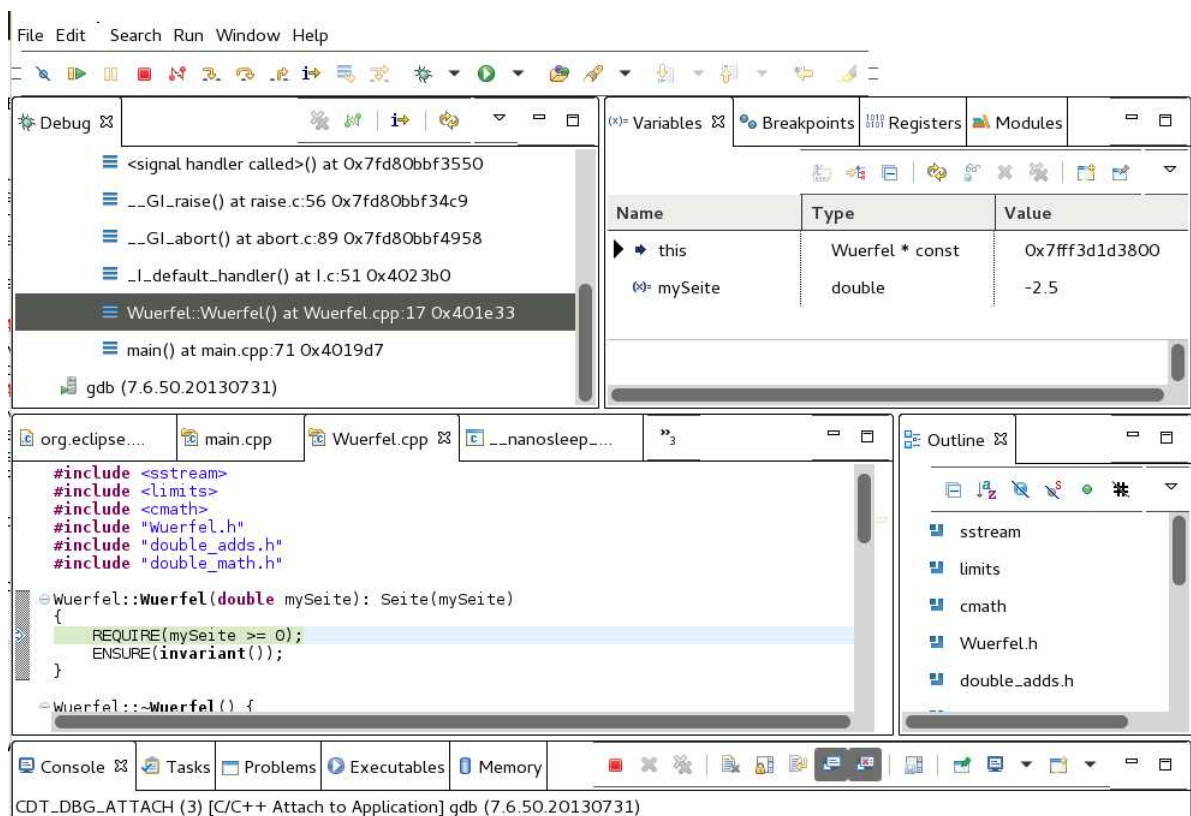
int main(int argc, char* argv[]) {

    signal(SIGABRT, ABRT_signalhandler);

    std::cout << "Testrahmenprogramm fuer Wuerfel:" << std::endl;

    Wuerfel w(-2.5);
    // assert( w.toString() == "1;6;1;1.73205" );
    ...
}

```



## 1.34. Nichtänderungs-Verträge für Attribute: Framebedingungen

```
/*
 * Wuerfel.cpp
 */
#include <sstream>
#include <limits>
#include <cmath>

#include "Wuerfel.h"
#include "double_adds.h"
#include "double_math.h"

Wuerfel::Wuerfel(double mySeite): Seite(mySeite)
{
    REQUIRE(mySeite >= 0);
    ENSURE(invariant());
}
Wuerfel::~~Wuerfel() {
    // TODO Auto-generated destructor stub
}
bool Wuerfel::invariant()
{
    return Seite >= 0.0;
}
std::string Wuerfel::toString()
{
    std::ostringstream help;
    help << Seite << ";" << Oberflaeche() << ";" << Volumen
        () << ";"
        << Raumdiagonale();
    return help.str();
}
double Wuerfel::Oberflaeche()
DO
    ID(double Seite_old = Seite);
    double result = 6.0 * pow(Seite, 2);
    ENSURE(approximatelyEqualTo(result, 6.0 * Seite * Seite
        , 1.0));
    ENSURE(Seite == Seite_old);
    ENSURE(invariant());
    return result;
```

```

}
double Wuerfel::Volumen()
DO
    ID(double Seite_old = Seite);
    double result = pow(Seite,3);
    ENSURE(approximatelyEqualTo(result , Seite*Seite*Seite ,
        1.0));
    ENSURE(Seite == Seite_old);
    ENSURE(invariant());
    return result;
}
double Wuerfel::Raumdiagonale()
DO
    ID(double Seite_old = Seite);
    double result = Seite * sqrt(3.0);
    ENSURE(approximatelyEqualTo(result , Seite * myown_sqrt
        (3.0) , 1.0));
    ENSURE(Seite == Seite_old);
    ENSURE(invariant());
    return(result);
}

```



## 1.35. Ultimative Nichtänderungsverträge: const-Methoden

```
/*
 * Wuerfel.h
 *
 * Created on: 22.04.2009
 * Author: buhl
 */
#ifndef WUERFEL_H
#define WUERFEL_H
#define EIFFEL_DOEND
#ifndef EIFFEL_CHECK
#define EIFFEL_CHECK CHECK_ALL
// Makros CHECK()
// und folgende
// CHECK_LOOP Makros INVARIANT
// () und folgende
// CHECK_INVARIANT Methode invariant
// () und folgende
// CHECK_ENSURE Nachbedingungen
// und folgende
// CHECK_REQUIRE Vorgedingungen
// CHECK_NO
#endif
#include <eiffel.h>
#include <nana.h>
#include <string>

class Wuerfel{
public:
    Wuerfel(double mySeite = 1.0);
    virtual ~Wuerfel();

    virtual bool invariant() const;

    std::string toString() const;

    double Oberflaeche() const;

    double Volumen() const;

    double Raumdiagonale() const;
```

```
protected:
    double Seite;
};
#endif /* WUERFEL_H */
```

```

/*
 * Wuerfel.cpp
 *
 * Created on: 22.04.2009
 * Author: buhl
 */

#include <sstream>
#include <limits>
#include <cmath>
#include "Wuerfel.h"
#include "double_adds.h"
#include "double_math.h"

Wuerfel::Wuerfel(double mySeite): Seite(mySeite)
{
    REQUIRE(mySeite >= 0);
    ENSURE(invariant());
}

Wuerfel::~~Wuerfel() {
    // TODO Auto-generated destructor stub
}

bool Wuerfel::invariant() const
{
    return Seite >= 0.0;
}

std::string Wuerfel::toString() const
DO
    std::ostringstream help;
    help << Seite << ";" << Oberflaeche() << ";" << Volumen
        () << ";"
        << Raumdiagonale();
    return help.str();
}

double Wuerfel::Oberflaeche() const
DO
    double result = 6.0 * pow(Seite, 2);
    ENSURE(approximatelyEqualTo(result, 6.0 * Seite * Seite
        , 1.0));
    return result;

```

```

}

double Wuerfel::Volumen() const
DO
    double result = pow(Seite,3);
    ENSURE(approximatelyEqualTo(result, Seite*Seite*Seite,
        1.0));
    return result;
}

double Wuerfel::Raumdiagonale() const
DO
    double result = Seite * sqrt(3.0);
    ENSURE(approximatelyEqualTo(result, Seite * myown_sqrt
        (3.0), 1.0));
    return(result);
}

```

<http://www.linuxjournal.com/article/7629>

[http://www.cprogramming.com/tutorial/const\\_correctness.html](http://www.cprogramming.com/tutorial/const_correctness.html)

[http://en.wikipedia.org/wiki/Const\\_correctness](http://en.wikipedia.org/wiki/Const_correctness)

Bei const-Methoden ist es in der Regel nicht notwendig, die Klassen-Invariante bei Methodenende zu überprüfen, wohl aber bei Methodenbeginn (eine private-Methode könnte vor Methodenaktivierung die Invariante ungültig gemacht haben!).

## 2. Programming by Contract

### 2.1. Spezifikation durch Verträge

(SdV, *Design by Contract*<sup>1</sup>, *Programming by Contract*) ist eine Methode zur Spezifikation der dynamischen Semantik von Softwarekomponenten mit Hilfe von Verträgen aus erweiterten booleschen Ausdrücken. SdV basiert auf der Theorie der abstrakten Datentypen und formalen Spezifikationsmethoden. Spezifizierte Komponenten können Module, Klassen oder Komponenten im Sinne von Komponententechnologien (wie Microsofts COM, .NET oder Suns EJB) sein. Verträge ergänzen das Kunden-Lieferanten-Modell:

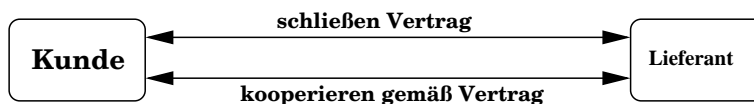


Abbildung 2.1.: Kunden-Lieferanten-Modell

Grundlegend für die Vertragsmethode ist das **Prinzip der Trennung von Diensten in Abfragen und Aktionen** (*command-query separation*):

- **Abfragen** geben Auskunft über den Zustand einer Komponente, verändern ihn aber nicht. Sie liefern als Ergebnis einen Wert. Die Abfragen einer Komponente beschreiben ihren abstrakten Zustand.
- **Aktionen** verändern den Zustand einer Komponente, liefern aber kein Ergebnis. Die Aktionen einer Komponente bewirken ihre Zustandsveränderungen.

Diesem Prinzip folgend sind seiteneffektbehaftete Funktionen als Dienste zu vermeiden<sup>2</sup>.

---

<sup>1</sup>„Design by Contract“ ist ein Warenzeichen von Interactive Software Engineering.

<sup>2</sup>In bestimmten Fällen, z.B. bei Fabrikfunktionen, können Seiteneffekte sinnvoll sein. Solche Funktionen sind nicht als Spezifikatoren verwendbar und sollten entsprechend gekennzeichnet sein.

Ein Grund dafür ist, dass Abfragen als **Spezifikatoren** dienen, d.h. als Elemente von Verträgen. **Verträge** setzen sich aus Bedingungen folgender Art zusammen:

- **Invarianten** einer Komponente sind allgemeine unveränderliche Konsistenzbedingungen an den Zustand einer Komponente, die vor und nach jedem Aufruf eines Dienstes gelten. Formal sind Invarianten boolsche Ausdrücke über den Abfragen der Komponente; inhaltlich können sie z.B. Geschäftsregeln (business rules) ausdrücken.
- **Vorbedingungen** (preconditions) eines Dienstes sind Bedingungen, die vor dem Aufruf eines Dienstes erfüllt sein müssen, damit er ausführbar ist. Invarianten sind boolsche Ausdrücke über den Abfragen der Komponente und den Parametern des Dienstes.
- **Nachbedingungen** (postconditions) eines Dienstes sind Bedingungen, die nach dem Aufruf eines Dienstes erfüllt sind; sie beschreiben, welches Ergebnis ein Dienstaufruf liefert oder welchen Effekt er erzielt. Nachbedingungen sind boolsche Ausdrücke über den Abfragen der Komponente und den Parametern des Dienstes, erweitert um ein Gedächtniskonstrukt, das die Werte von Ausdrücken vor dem Dienstaufruf liefert.

Verträge legen Pflichten und Nutzen für Kunden und Lieferanten fest. Die Verantwortlichkeiten sind klar verteilt:

Der Lieferant garantiert die Nachbedingung jedes Dienstes, den der Kunde aufruft, falls der Kunde die Vorbedingung erfüllt. Eine verletzte Vorbedingung ist ein Fehler des Kunden, eine verletzte Nachbedingung oder Invariante (bei erfüllter Vorbedingung) ist ein Fehler des Lieferanten.

	KUNDE	LIEFERANT
PFLICHT	Die Vorbedingung einhalten.	Anweisungen ausführen, die die Nachbedingungen herstellen und die Invarianten erhalten
NUTZEN	Ergebnisse/Wirkungen nicht prüfen, da sie durch die Nachbedingungen garantiert sind.	Aufrufe, die die Vorbedingung verletzen, ignorieren. (Die Vorbedingungen nicht prüfen.)

Tabelle 2.1.: Pflichten - Nutzen von Kunden und Lieferanten

Schwache Vorbedingungen erleichtern den Kunden die Arbeit, starke Vorbedingungen dem Lieferanten. Je schwächer die Nachbedingungen sind, umso freier ist der Lieferant und umso ungewisser sind die Kunden über das Ergebnis/den Effekt. Je stärker die Nachbedingungen sind, umso mehr muß der Lieferant leisten.

Siehe auch:

[Spezifikation durch Vertrag — eine Basistechnologie für eBusiness](#)

Einige Beispielverträge für eine Klasse `vektor`:

- friend-Funktion `Norm()` (abgeleitete Abfrage)

```

double Norm(const vektor& v)
{
    REQUIRE(v.invariant());
    ...
    ENSURE(approximatelyEqualTo(qsum, S(int k=v.lo(), k<=v.hi
        ( ),k++,
                                v(k)*v(k)), 2.0));
    ENSURE(approximatelyEqualTo(result*result, qsum, 2.0));
    ...
}

```

- Methode `normalize()` (Modifikator)

```

void vektor::normalize()
DO
    REQUIRE(Norm(*this)!=0.0);
    ID(vektor value_old(*this));
    ...
    // double n = ...
    ENSURE(approximatelyEqualVekTo(result*n, value_old,
        2.0));
    ENSURE(approximatelyEqualTo(Norm(result), 1.0, 2.0)
        );
    ...
END

```

- i-ter Einheitsvektor (statische Klassenmethode)

```

vektor vektor::ei(int n, int i)
{
    REQUIRE((n>=1) && (1<=i) && (i<=n));
    ...
    ENSURE(result.lo()==1);
    ENSURE(result.hi()==n);
    ENSURE(E1(int k=result.lo(), k<=result.hi(), k++,
        result(k)!=0.0));
    ENSURE(result(i)==1.0);
    ENSURE(result.invariant());
    ...
}

```

- Konstruktor

```
vektor::vektor(const double x[], int n) : low(1), high(n)
{
    REQUIRE((n>=1) && (x!=0));
    REQUIRE("x[] hat mindestens n Komponenten");
    ...
    ENSURE(lo()==1 && hi()==n);
    ENSURE(A(int k=lo(), k<=hi(), k++, (*this)(k)==x[k-lo()]
    ));
END
```

- Modifikator

```
void vektor::changeValueAt(int i, double x)
DO
    REQUIRE((lo()<=i) && (i <=hi()));
    ...
    ENSURE((*this)(i)==x);
    ENSURE("alle anderen Komponenten von *this ungeändert");
END
```

Überlegen Sie sich einen expliziten Nichtänderungsvertrag für „alle anderen Komponenten“ von `*this` (Frame-Bedingung).

- `operator!=` (abgeleitete Abfrage)

```
bool vektor::operator!=(const vektor& w) const
DO
    REQUIRE(w.invariant());
    ...
    ENSURE(result == ((hi()-lo())!=(w.hi()-w.lo())) ||
    E(int k=lo(), k<=hi(), k++, (*this)(k)!=w(k-lo()+w.lo
    ()))));
    ...
}
```

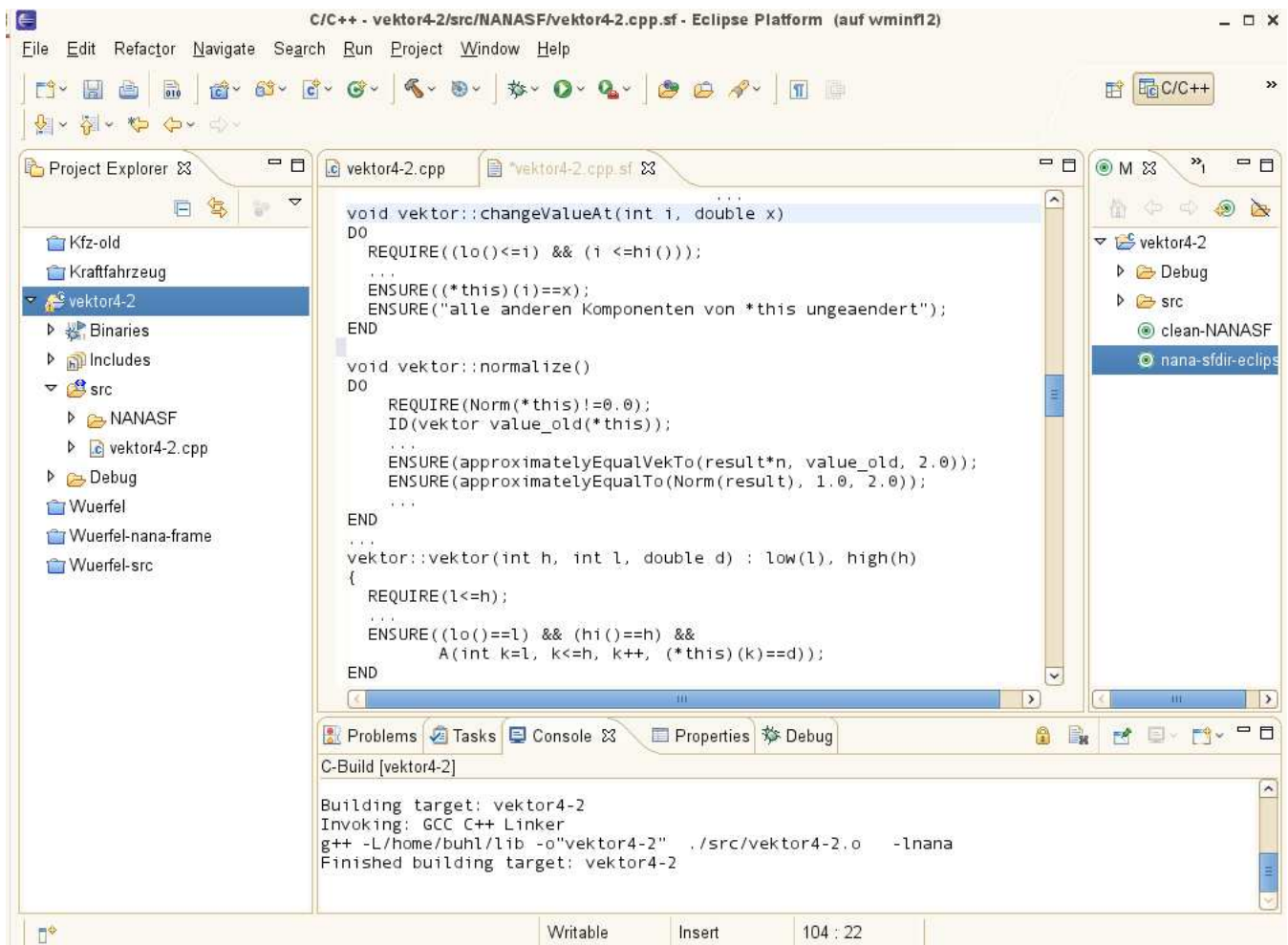
**Bemerkung:** C++11 enthält alternativ zu den `nana`-Quantoren nutzbare STL-Prädikate:

`none_of`, ...

Siehe: [C++11, Seite 863f.](#)



## 2.2. Alle Verträge der Klasse Klasse vektor



### 2.2.1. Klassendeklaration/ Interface

```
#define EIFFEL_DOEND
#define EIFFEL_CHECK CHECK_ALL

#include <iostream>
#include <limits>
#include <cmath>
#include <algorithm>

#include <eiffel.h>
#include "nana.h"
```

```

class vektor
{
private:
    int low;                // v(low..high)
    int high;

    double* vp;            // Startadresse fuer dyn. verwaltetes
                          // Exemplar

public:
    virtual bool invariant() const;

    // Grundlegende Abfragen:
    int lo() const;
    int hi() const;

    double operator()(int i) const;

    // Modifikatoren:
    double& operator()(int i);
    void changeValueAt(int i, double x);
    void normalize();

    // Konstruktoren:
    vektor(int h, int l = 1, double d = 0.0);
    // v(l..h) = d

    vektor(int h, double d);
    // v(1..h) = d

    vektor(const double x[], int n);
    // v(1..n) = x[0..n-1]

    // Kopierkonstruktor, Destruktor, Wertzuweisung
    vektor(const vektor& w);
    virtual ~vektor();
    vektor& operator=(const vektor& w);

    // abgeleitete Abfragen:
    bool operator==(const vektor& w) const;
    bool operator!=(const vektor& w) const;

    friend ostream& operator<<(ostream& os, const vektor& v);

```

```

// Operationen:
vektor operator+(const vektor& w) const;
friend vektor operator+(const vektor& w, double a);
friend vektor operator+(double a, const vektor& w);
friend vektor operator*(const vektor& w, double a);
friend vektor operator*(double a, const vektor& w);

friend double Skalarprodukt(const vektor& v, const vektor& w)
    ;
friend double Norm(const vektor& v);
friend bool approximatelyEqualVekTo(const vektor& left ,
                                     const vektor& right ,
                                     double factor);

static vektor ei(int n, int i);
};

```

Die `private`-Deklarationen sollten eigentlich dem Klassenbenutzer nicht kenntlich sein, also von `nana-sfg` unterdrückt werden!

### 2.2.1.1. Grundlegende Abfragen

```
int vektor::lo() const
DO
    ...
}
int vektor::hi() const
DO
    ...
}
double vektor::operator()(int i) const
DO
    REQUIRE((lo()<=i) && (i<=hi()));
    ...
    ENSURE(result == vp[i-low]); // Nachbedingung nur fuer
                                // das Implementierungsteam
                                // damit automatische
                                // Ueberpruefung
                                // stattfinden kann
}

```

### 2.2.1.2. Klassen-Invariante

```
bool vektor::invariant() const // Vorsicht! Kein DO
{
    ... // nicht lo(), hi(), operator
    () // benutzen, da sonst
        Endlosrekursion
}

```

Die Klasseninvariante ist primär für die automatische Überprüfung der Gültigkeit von `*this`, also als Unterstützung des Implementierungsteams konzipiert. Der Klassenbenutzer braucht nicht direkt zu wissen, was sie überprüft, da er durch `public`-Methoden und Konstruktoren nur gültige Klassenexemplare erzeugen kann.

### 2.2.1.3. Konstruktoren

```
vektor::vektor(int h, int l, double d) : low(l), high(h)
{
    REQUIRE(l<=h);
    ...
    ENSURE((lo()==l) && (hi()==h) &&
           A(int k=l, k<=h, k++, (*this)(k)==d));
}
END
...

```

```

vektor::vektor(int h, double d) : low(1), high(h)
{
    REQUIRE(h>=1);
    ...
    ENSURE((lo()==1) && (hi()==h) &&
            A(int k=lo(), k<=hi(), k++, (*this)(k)==d));
END

...
vektor::vektor(const double x[], int n) : low(1), high(n)
{
    REQUIRE((n>=1) && (x!=0));
    REQUIRE("x[] hat mindestens n Komponenten");
    ...
    ENSURE(lo()==1 && hi()==n);
    ENSURE(A(int k=lo(), k<=hi(), k++, (*this)(k)==x[k-lo()]));
END

...
vektor::vektor(const vektor& w) : low(w.low), high(w.high)
{
    REQUIRE(w.invariant());
    ...
    ENSURE((hi()-lo()==(w.hi()-w.lo()) &&
            A(int k=lo(), k<=hi(), k++, (*this)(k)==w(k-lo()+w.lo
            ()))));
END

```

#### 2.2.1.4. Destruktor

```
vektor::~~vektor()
DO
    ...
}
```

#### 2.2.1.5. abgeleitete Abfragen/Operationen auf vektor-Exemplaren

```
ostream& operator<<(ostream& os, const vektor& w)
{
    ...
    ENSURE("Drucke alle vektor-Komponenten auf Stream os");
    ...
};
```

```
bool vektor::operator!=(const vektor& w) const
DO
    REQUIRE(w.invariant());
    ...
    ENSURE(result == ((hi()-lo())!=(w.hi()-w.lo())) ||
            E(int k=lo(), k<=hi(), k++, (*this)(k)!=w(k-
            lo()+w.lo())));
    ...
}
```

```
bool vektor::operator==(const vektor& w) const
DO
    REQUIRE(w.invariant());
    ...
    ENSURE(result == !((*this) != w));
    ...
}
```

```
double Skalarprodukt(const vektor& v, const vektor& w)
{
    REQUIRE((v.hi()-v.lo()) == (w.hi()-w.lo()));
    REQUIRE(v.invariant());
    REQUIRE(w.invariant());
    ...
    ENSURE(approximatelyEqualTo(result, S(int k=v.lo(), k<=v.hi()
    ,k++,
    v(k)*w(k-v.lo()+w.lo())
    ), 2.0));
    ...
}
```

```

}

double Norm(const vektor& v)
{
    REQUIRE(v.invariant());
    ...
    ENSURE(approximatelyEqualTo(qsum, S(int k=v.lo(), k<=v.hi(),k
        ++,
                                v(k)*v(k)),
                                2.0));
    ENSURE(approximatelyEqualTo(result*result, qsum, 2.0));
    ...
}

bool approximatelyEqualVekTo(const vektor& left, const vektor&
    right,
                             double factor)
{
    ...
    ENSURE("relativer Abstand zwischen vektor left und
        right ist klein");
    ...
}

```

### 2.2.1.6. Modifikatoren

```
double& vektor::operator()(int i) // ermoeeglicht externe
    Modifikationen
    // deshalb besser durch
    // folgende
    // Methode ersetzen!
DO
    REQUIRE((lo()<=i) && (i<=hi()));
    ...
    ENSURE(invariant());
    ...
}
```

```
void vektor::changeValueAt(int i, double x)
DO
    REQUIRE((lo()<=i) && (i <=hi()));
    ...
    ENSURE((*this)(i)==x);
    ENSURE("alle anderen Komponenten von *this ungeaendert");
END
```

```
vektor& vektor::operator=(const vektor& w)
DO
    ENSURE(w.invariant());
    ...
    ENSURE(*this == w);
    ...
}
```

```
void vektor::normalize()
DO
    REQUIRE(Norm(*this)!=0.0);
    ID(vektor value_old(*this));
    ...
    ENSURE(approximatelyEqualVekTo(result*n, value_old, 2.0));
    ENSURE(approximatelyEqualTo(Norm(result), 1.0, 2.0));
    ...
END
```

### 2.2.1.7. Operationen, die vektor-Exemplare erzeugen

```
vektor vektor::operator+(const vektor& w) const
DO
    REQUIRE(w.invariant());
```



```

    REQUIRE(( hi ()-lo ())==(w. hi ()-w. lo ()));
    ...
    ENSURE(result.lo()==lo());
    ENSURE(result.hi()==hi());
    ENSURE(A(int k=lo(),k<=hi(),k++,
              approximatelyEqualTo(result(k),
                                   (*this)(k)+w(k-lo()+w.lo
                                   ()),2.0)));
    ENSURE(result.invariant());
}

```

```

vektor operator+(const vektor& w, double a)
{
    REQUIRE(w.invariant());
    ...
    ENSURE(result.lo()==w.lo());
    ENSURE(result.hi()==w.hi());
    ENSURE(A(int k=result.lo(),k<=result.hi(),k++,
              approximatelyEqualTo(result(k),w(k)+a,2.0)));
    ENSURE(result.invariant());
    ...
}

```

```

vektor operator+(double a, const vektor& w)
{
    REQUIRE(w.invariant());
    ...
    ENSURE(result.lo()==w.lo());
    ENSURE(result.hi()==w.hi());
    ENSURE(A(int k=result.lo(),k<=result.hi(),k++,
              approximatelyEqualTo(result(k),a+w(k),2.0)));
    ENSURE(result.invariant());
    ...
}

```

```

vektor operator*(const vektor& w, double a)
{
    REQUIRE(w.invariant());
    ...
    ENSURE(result.lo()==w.lo());
    ENSURE(result.hi()==w.hi());
    ENSURE(A(int k=result.lo(),k<=result.hi(),k++,
              approximatelyEqualTo(result(k),w(k)*a,2.0)));
    ENSURE(result.invariant());
}

```

```

    ...
}

vektor operator*(double a, const vektor& w)
{
    REQUIRE(w.invariant());
    ...
    ENSURE(result.lo() == w.lo());
    ENSURE(result.hi() == w.hi());
    ENSURE(A(int k=result.lo(), k<=result.hi(), k++,
              approximatelyEqualTo(result(k), a*w(k), 2.0)));
    ENSURE(result.invariant());
    ...
}

vektor vektor::ei(int n, int i)                // static
    Klassenmethode
{
    REQUIRE((n>=1) && (1<=i) && (i<=n));
    ...
    ENSURE(result.lo() == 1);
    ENSURE(result.hi() == n);
    ENSURE(E1(int k=result.lo(), k<=result.hi(), k++, result(k)
              != 0.0));
    ENSURE(result(i) == 1.0);
    ENSURE(result.invariant());
    ...
}

```

### 2.2.1.8. benutzte klassenexterne Hilfsfunktionen/-Methoden

```

bool approximatelyEqualTo(double left, double right, double
    factor)
{
    ENSURE("relativer Abstand zwischen left und right ist klein")
    ;
    ...
}

```

### 2.2.2. Quellcode

<http://www.math.uni-wuppertal.de/~buhl/teach/exercises/PbC09/vektor4-2.cpp>

## 2.3. Ein Vertrag zur Klasse rationalNumber

Die Klasse

```
#define EIFFEL_DOEND
#define EIFFEL_CHECK CHECK_ALL
#include <eiffel.h>
#include "nana.h"
#include <iostream>
#include <climits>
#include <cstdlib>
...
class rationalNumber
{
    long Z, Nenner;
public:
    void kuerze(); // sollte privat sein
public:
    rationalNumber(long z = 0, long n = 1.0);
    long getZ() const;
    long getN() const;
    virtual bool invariant() const;
};

// Definitionen:

bool MulOk( long a, long b );
bool DivOk( long a, long b );
bool AddOk( long a, long b );

long ggT(long a, long b);
long kgV(long a, long b);

rationalNumber operator+ (const rationalNumber &r1, const
    rationalNumber &r2);
rationalNumber operator- (const rationalNumber &r1, const
    rationalNumber &r2);
rationalNumber operator* (const rationalNumber &r1, const
    rationalNumber &r2);
```

```
rationalNumber operator/ ( const rationalNumber &r1, const  
    rationalNumber &r2);
```

```
bool operator==( const rationalNumber left, const rationalNumber  
    right );
```

```
ostream &operator<< (ostream &os, const rationalNumber &r);
```

und ihre Verträge:

```

// grundlegende Abfragen:

inline long rationalNumber::getZ() const { return Z; };
inline long rationalNumber::getN() const { return Nenner; };

// abgeleitete Abfragen:

bool operator==(const rationalNumber left, const rationalNumber
    right)
{
    REQUIRE(left.invariant());
    REQUIRE(right.invariant());
    ...
    ENSURE("left repraesentiert denselben Bruch wie right");
    return (left.getN()==right.getN()) &&
        (left.getZ()==right.getZ());
}

// Die Klassen-Invariante:

bool rationalNumber::invariant() const{
    return (getN() > 0) &&
        (ggT(getN(), getZ())==1) &&
        ((getZ()!=0) || (getN()==1));
}

// Konstruktor:

inline rationalNumber::rationalNumber(long z, long n) : Z(z),
    Nenner(n)
{
    REQUIRE(n != 0);
    ...
    ENSURE((getZ()==0) || (z%getZ() == 0));
    ENSURE((getN()==0) || (n%getN() == 0));
    ENSURE(z * getN() == n * getZ()); // (z / getZ()) == (
        n / getN())
}
END;

// Hilfsfunktionen zur Vermeidung von Integer-Overflows:
bool MulOk( long a, long b )
{
    ...
}

```

```

}
bool DivOk( long a, long b )
{
  ...
}
bool AddOk( long a, long b )
{
  ...
}

// Funktionen zum Kuerzen, um unnoetige Overflows
// solange wie moeglich aufzuschieben UND eindeutige
// Repraesentanten zu ermoeglichen
long ggT(long a, long b)
{
  ID(long a_old(a));
  ID(long b_old(b));
  ...
  ENSURE(result >= 0);
  ENSURE((result == 0) || labs(a_old) % result == 0);
  ENSURE((result == 0) || labs(b_old) % result == 0);
  ENSURE("result ist groesster solcher Teiler von a_old und
          b_old, ausser wenn:");
  ENSURE(!((a_old==0)&&(b_old==0)) || (result==0));
  ...
}
long kgV(long a, long b)
{
  ...
  ENSURE(result % a == 0);
  ENSURE(result % b == 0);
  ENSURE("result ist kleinstes solches gemeinsames Vielfaches
          von a und b");
  // ENSURE(ggT(a,b) == labs(a*b)/result); // Konsistenz zu
  ggT()
  ...
}
...
void rationalNumber::kuerze ()
{
  // Invariante nicht erfuehlt (!)
  ,
  ...
  ID(long Z_old(getZ()));
  ID(long N_old(getN()));
}

```

```

...
ENSURE(ggT(getZ(),getN())==1);
ID(long f = N_old/getN());
ENSURE(f * getZ()==Z_old);
END

// Operationen mit rationalNumber's:

rationalNumber operator+ (const rationalNumber &r1, const
    rationalNumber &r2)
{
    REQUIRE(r1.invariant());
    REQUIRE(r2.invariant());
    ...
    ENSURE(r.invariant());
    ENSURE("r ist Summe von r1 und r2");
}
rationalNumber operator- (const rationalNumber &r1, const
    rationalNumber &r2)
{
    REQUIRE(r1.invariant());
    REQUIRE(r2.invariant());
    ...
    ENSURE(r.invariant());
    ENSURE(r+r2 == r1); // Konsistenz zu operator+
}
rationalNumber operator* (const rationalNumber &r1, const
    rationalNumber &r2)
{
    REQUIRE(r1.invariant());
    REQUIRE(r2.invariant());
    ...
    ENSURE(r.invariant());
    ENSURE("r ist Produkt von r1 und r2");
}
rationalNumber operator/ (const rationalNumber &r1, const
    rationalNumber &r2)
{
    REQUIRE(r1.invariant());
    REQUIRE(r2.invariant());
    ...
    ENSURE(r.invariant());
    ENSURE(r*r2==r1); // Konsistenz zu operator*
}

```

```
ostream &operator<< (ostream &os, const rationalNumber &r)
{
    ...
    ENSURE("Druckt rationalNumber in der Form (./.) auf os");
    ...
}
```



## 2.4. Ein Vertrag mit Queries, Invariants und Actions

Das folgende Beispiel (in der **Contract Specification Language CLEO** formuliert) modelliert eine (mathematische) Menge als **Set**:

```
INTERFACE Set[Element]

  QUERIES
    Count:INTEGER
      - - Number of elements in the set.

    Has(IN x:Element):BOOLEAN
      - - Does the set contain x?
  POST
    result IMPLIES(Count>0)

    IsEmpty:BOOLEAN
      - - Does the set contain no element?

  INVARIANTS
    Count>=0
    IsEmpty=(Count=0)
  ACTIONS
    Put(IN x:Element)
      - - include x into the set.
    POST
      Has(x)
      OLD (Has(x)) IMPLIES (Count=OLD(Count))
      NOT OLD (Has(x)) IMPLIES (Count=OLD(Count)+1)

    Remove(IN x:Element)
      - - Exclude x from the set.
    POST
      NOT Has(x)
      OLD (Has(x)) IMPLIES (Count=OLD(Count)-1)
      NOT OLD (Has(x)) IMPLIES (Count=OLD(Count))

    WipeOut
      - - Exclude all elements from the set.
    POST
      Count=0

END Set
```

Um das Verhalten der Aktionen **Put** und **Remove**, des Hinzufügens eines Elementes zur Menge und des Entfernens eines Elementes aus der Menge, zu spezifizieren,

ist der Zustand der Menge vor einem Aktionsaufruf mit ihrem Zustand nach dem Aktionsaufruf zu vergleichen. Mit OLD (...) geklammerte Ausdrücke liefern den Wert des geklammerten Ausdrucks vor einem Dienstaufufr. OLD-Ausdrücke dürfen nur in Nachbedingungen auftreten. der Ausdruck

$$\text{Count}=\text{OLD}(\text{Count})+1$$

bedeutet, das sich der Wert von Count durch die Ausführung des Dienstes um 1 erhöht. Dabei muss es sich um eine Aktion handeln, denn da eine Abfrage q den Zustand ihrer Komponente unverändert lässt, gelten für sie Nachbedingungen der Art

$$q=\text{OLD}(q)$$

mit beliebigem Attribut q.

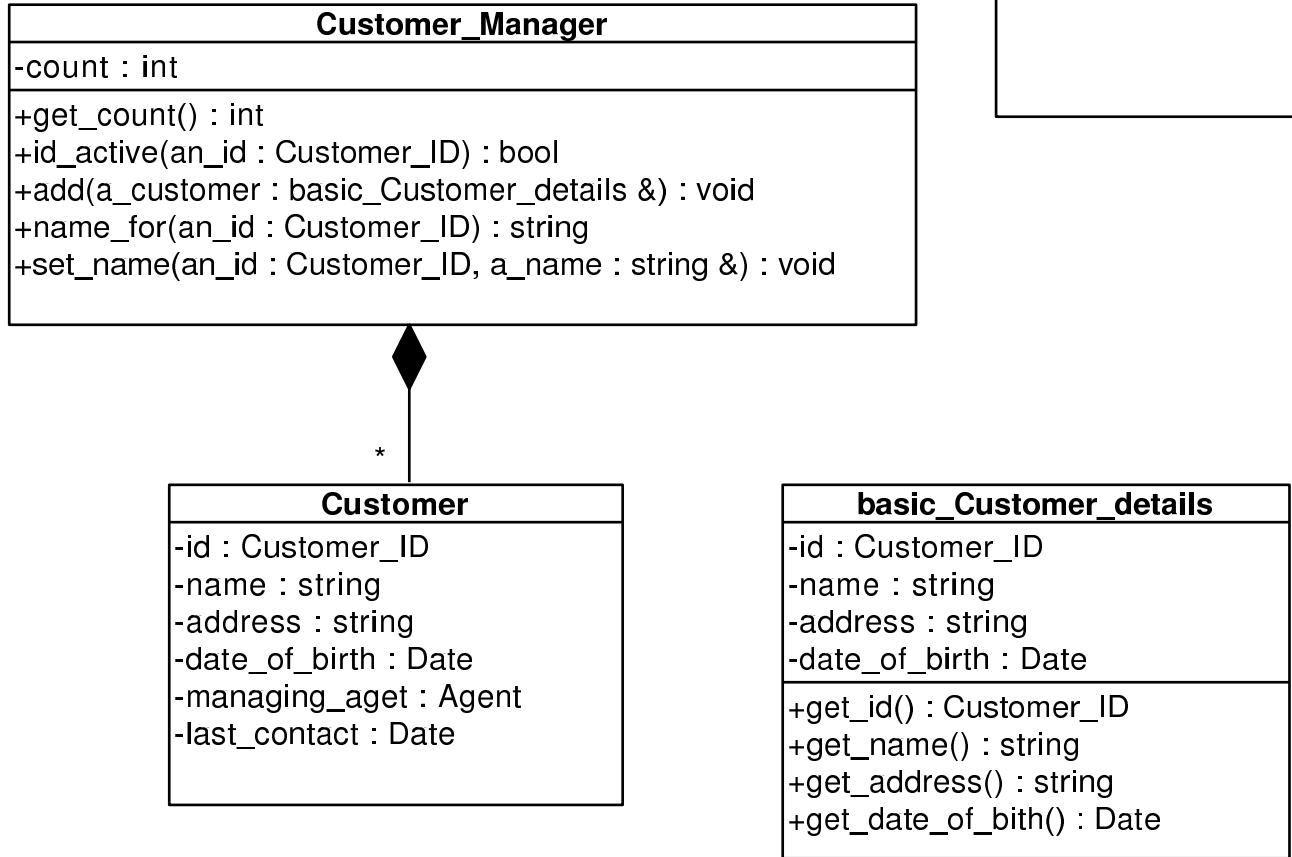
Vergleiche auch: <http://informatik.karlheinz-hug.de/artikel/ForumWI01%20SdV.pdf>

## **A. Design by Contract, by Example, in C++ with nana**

# A.1. A first Taste of Design by Contract

Visual Paradigm for UML Standard Edition(University of Wuppertal)

Abschnitt 1.2



// Kapitel 7.1.2

```

//
// c++ -c DbCrev.cc -I$HOME/include -L$HOME/lib -lnana
//
// eventuell mit
//             -DWITHOUT_NANA
// und/oder
//             -DNDEBUG
//
// und/oder
//             -DEIFELL_CHECK=CHECK_....
//
// -----
// evtl. myexcept.o mit angeben

#define EIFFEL_DOEND

#ifndef EIFFEL_CHECK
#define EIFFEL_CHECK CHECK_ALL
//
//             CHECK_LOOP             Makros CHECK() und folgende
//             CHECK_INVARIANT        Makros INVARIANT() und folgende
//             CHECK_ENSURE           Methode invariant() und folgende
//             CHECK_REQUIRE          Nachbedingungen und folgende
//             CHECK_NO               Vorgedingungen
#endif
#include "eiffel.h"
#include "nana.h"

#include <string>
#include <vector>
#include <exception>

using namespace std;

// -----

class Date
{
private: int day;
private: int month;
private: int year;

```

```

public:
    // ...
};

class Customer_ID
{
    // ...
};

class Agent
{
    // ...
};

class Customer_Manager;

// -----

    class Customer
    {
        private: Customer_ID id;
        private: string name;
        private: string address;
        private: Date date_of_birth;
        private: Agent managing_aget;
        private: Date last_contact;

        private: Customer_Manager* customer_manager;
    };

// -----

class basic_Customer_details
{
    private: Customer_ID id;
    private: string name;
    private: string address;
    private: Date date_of_birth;

    public: Customer_ID get_id() const;
    public: string get_name() const;
    public: string get_address() const;
    public: Date get_date_of_bith() const;
};

```

```

};

// -----

class Customer_Manager
{
    private: int count;
    private: virtual bool invariant() const;

    private: vector<Customer*> customer;

    public: int get_count() const;
    public: bool id_active(Customer_ID an_id) const;
    public: void add(const basic_Customer_details& a_customer);
    public: string name_for(Customer_ID an_id) const;
    public: void set_name(Customer_ID an_id, const string& a_name);
};

bool Customer_Manager::invariant() const
{
    return get_count() >= 0;
};

// Anzahl der Kunden, die vom Custom_Manager verwaltet werden
//
int Customer_Manager::get_count() const
DO
    throw "Not yet implemented";
END;

// Existiert ein Kunde mit der Kennung an_id?
//
bool Customer_Manager::id_active(Customer_ID an_id) const
DO
    throw "Not yet implemented";
END;

// Fuege a_customer der Customer_Manager-Datenbasis hinzu
//
void Customer_Manager::add(const basic_Customer_details& a_customer)
DO
    REQUIRE(!id_active(a_customer.get_id()));
    ID(int old_count = get_count());

```

```

        throw "Not yet implemented";
    ENSURE(get_count() == old_count + 1 );
    ENSURE(id_active(a_customer.get_id()));
END;

// was ist der Name des mit an_id gekennzeichneten Kunden?
//
string Customer_Manager::name_for(Customer_ID an_id) const
DO
    REQUIRE(id_active(an_id));
    throw "Not yet implemented";
END;

// Setze/Ändere den Namen des mit an_id gekennzeichneten Kunden auf a_name
//
void Customer_Manager::set_name(Customer_ID an_id, const string& a_name)
DO
    REQUIRE(id_active(an_id));
    throw "Not yet implemented";
    ENSURE(name_for(an_id) == a_name);
END;

// -----

int main(){

    exit (0);
}

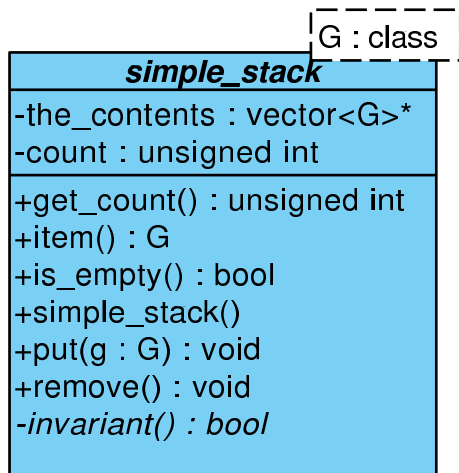
```



## A.2. Elementary Principles of Design by Contract

### A.2.1. First Trial

Visual Paradigm for UML Standard Edition(University



```
//
// simple_stack0.cc
//
// g++ -g -o simple_stack0 simple_stack0.cc -I$HOME/include -L$HOME/lib -lnana
//
//
// eventuell mit
//             -DWITHOUT_NANA
// und/oder
//             -DNDEBUG
//
// und/oder
//             -DEIFFEL_CHECK=CHECK_....
//
// -----
// evtl. myexcept.o mit angeben
// -----
//
// oder: nana-c++lg simple_stack0.cc
//

#define EIFFEL_DOEND
#ifndef EIFFEL_CHECK
#define EIFFEL_CHECK CHECK_ALL
//
//                                     Makros CHECK() und folgende
```

```

//          CHECK_LOOP          Makros INVARIANT() und folgende
//          CHECK_INVARIANT      Methode invariant() und folgende
//          CHECK_ENSURE         Nachbedingungen und folgende
//          CHECK_REQUIRE        Vorgedingungen
//          CHECK_NO
#endif

```

```

#include <iostream>
#include <vector>

```

```

#include <eiffel.h>
#include <nana.h>

```

```

using namespace std;

```

```

////////// class declaration //////////

```

```

template<class G>
class simple_stack{

```

```

private:

```

```

    vector<G>* the_contents;
    unsigned int count;

```

```

public:

```

```

    ////////// basic queries:

```

```

    unsigned int get_count() const;    // number of items in stack

    G item() const;                  // get top item

```

```

    ////////// class invariant:

```

```

private:
    virtual bool invariant() const;
public:

```

```

    ////////// derived queries:

```

```

bool is_empty() const;

////////// constructors:

simple_stack();

////////// (pure) modifiers:

void put(G g);                // Push 'g' onto the stack

void remove();               // delete the top item

};

////////// class definition //////////

////////// basic queries:

template<class G>
unsigned int simple_stack<G>::get_count() const{
    return count;
};

template<class G>
G simple_stack<G>::item() const{           // the item on the top of stack
    REQUIRE( /* stack not empty */ get_count() > 0 );
    return the_contents->at(count-1);
};

////////// class invariant:

template<class G>
bool simple_stack<G>::invariant() const {
    return (count >= 0) && (the_contents != 0);
};

////////// derived queries:

template<class G>
bool simple_stack<G>::is_empty() const{ // does the stack contain no items?
    bool Result = (get_count() == 0);
    ENSURE( /* consistend with count */      Result == (get_count() == 0) );
    return Result;
};

```

```

};

////////// constructors:

template<class G>
simple_stack<G>::simple_stack(){
    count = 0;
    the_contents = new vector<G>(100);

    ENSURE(/* stack is empty */      (get_count() == 0));
    ENSURE(invariant());
};

////////// (pure) modifiers:

template<class G>
void simple_stack<G>::put(G g)      // Push 'g' onto the stack
DO
    ID(int count_old = get_count());
    count++;
    the_contents->at(count-1) = g;
    ENSURE(/* count incremented */   get_count() == count_old + 1 );
    ENSURE(/* g on top */           item() == g );
END;

template<class G>
void simple_stack<G>::remove()      // delete the top item;
DO
    ID(int count_old = get_count());
    REQUIRE(/* stack not empty */   get_count() > 0 );
    count--;
    ENSURE(/* count decremented */   get_count() == count_old - 1 );
END;

////////// class test main() program //////////

int main(){

    cout << "\nTest der Klasse simple_stack: -----" << endl;

    simple_stack<long> s;

    s.put(10);

```

```

cout << s.item() << endl;
s.put(20);
cout << s.item() << endl;
s.put(30);
std::cout << s.item() << endl;

std::cout << endl;

// Exercise 'remove'
cout << s.item() << endl;
s.remove();
cout << s.item() << endl;
s.remove();
cout << s.item() << endl;

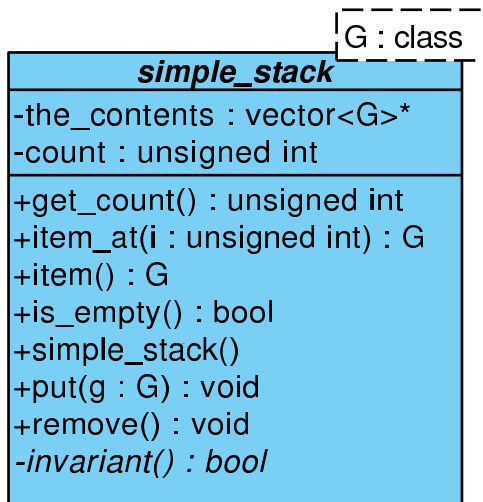
// Exercise contract violation
s.remove();
cout << s.item() << endl;

cout << "----- Testende -----" << endl << endl;
}

```

## A.2.2. Redesign

Visual Paradigm for UML Standard Edition(University of



```

//
// simple_stack.cc
//

```

```

// g++ -g -o simple_stack simple_stack.cc -I$HOME/include -L$HOME/lib -lnana
//
//
// eventuell mit
//             -DWITHOUT_NANA
// und/oder
//             -DNDEBUG
//
// und/oder
//             -DEIFFEL_CHECK=CHECK_....
//
// -----
// evtl. myexcept.o mit angeben
// -----
//
// oder: nana-c++lg simple_stack0.cc
//

#define EIFFEL_DOEND
#ifndef EIFFEL_CHECK
#define EIFFEL_CHECK CHECK_ALL
//
//             CHECK_LOOP           Makros CHECK() und folgende
//             CHECK_INVARIANT      Makros INVARIANT() und folgende
//             CHECK_ENSURE         Methode invariant() und folgende
//             CHECK_REQUIRE        Nachbedingungen und folgende
//             CHECK_NO             Vorgedingungen
#endif

#include <iostream>
#include <vector>

#include <eiffel.h>
#include <nana.h>

using namespace std;

////////// class declaration //////////

template<class G>
class simple_stack{

```

```

private:

    vector<G>* the_contents;
    unsigned int count;

public:

    //////////// basic queries:

    unsigned int get_count() const;    // number of items in stack

    G item_at(unsigned int i) const;

    //////////// class invariant:

private:
    virtual bool invariant() const;

public:

    //////////// derived queries:

    G item() const;                    // the item on the top of stack

    bool is_empty() const;            // does the stack contain no items?

    //////////// constructors:

    simple_stack();

    //////////// (pure) modifiers:

    void put(G g);                      // Push 'g' onto the stack

    void remove();                      // delete the top item;

};

////////// class definition //////////

////////// basic queries:

template<class G>
G simple_stack<G>::item_at(unsigned int i) const {

```

```

    REQUIRE( /* big enough */ i >= 1);
    REQUIRE( /* small enough */ i <= get_count() );

    return the_contents->at(i-1);
};

template<class G>
unsigned int simple_stack<G>::get_count() const{
    return count;
};

////////// class invariant:

template<class G>
bool simple_stack<G>::invariant() const {
    return (get_count() >= 0) && (the_contents != 0);
};
////////// derived queries:

template<class G>
G simple_stack<G>::item() const{ // the item on the top of stack
    REQUIRE( /* stack not empty */ get_count() > 0 );

    G result = the_contents->at(count-1);
    ENSURE( /* consistend with item_at() */ result == item_at(get_count()) );
    return result;
};

template<class G>
bool simple_stack<G>::is_empty() const{ // does the stack contain no items?

    bool result = (count == 0);
    ENSURE( /* consistend with count */ result == (get_count() == 0) );
    return result;
};

////////// constructors:

template<class G>
simple_stack<G>::simple_stack(){

    count = 0;
    the_contents = new vector<G>(100);

```



```

    ENSURE(/* stack is empty */      (get_count() == 0));
    ENSURE(invariant());
};

////////// (pure) modifiers:

template<class G>
void simple_stack<G>::put(G g)      // Push 'g' onto the stack
DO
    ID(unsigned int count_old = get_count());
    count++;
    the_contents->at(count-1) = g;
    ENSURE(/* count incremented */   get_count() == count_old + 1 );
    ENSURE(/* g on top */            item_at(get_count()) == g );
END;

template<class G>
void simple_stack<G>::remove()      // delete the top item;
DO
    ID(unsigned int count_old = get_count());
    REQUIRE(/* stack not empty */    get_count() > 0 );
    count--;
    ENSURE( /* count decremented */   get_count() == count_old - 1 );
END;

////////// class test main() program //////////

int main(){

    cout << "\nTest der Klasse simple_stack: -----" << endl;

    simple_stack<long> s;

    s.put(10);
    cout << s.item() << endl;
    s.put(20);
    cout << s.item() << endl;
    s.put(30);
    cout << s.item() << endl;

    cout << endl;

    // Exercise 'remove'

```

```

    cout << s.item() << endl;
    s.remove();
    cout << s.item() << endl;
    s.remove();
    cout << s.item() << endl;

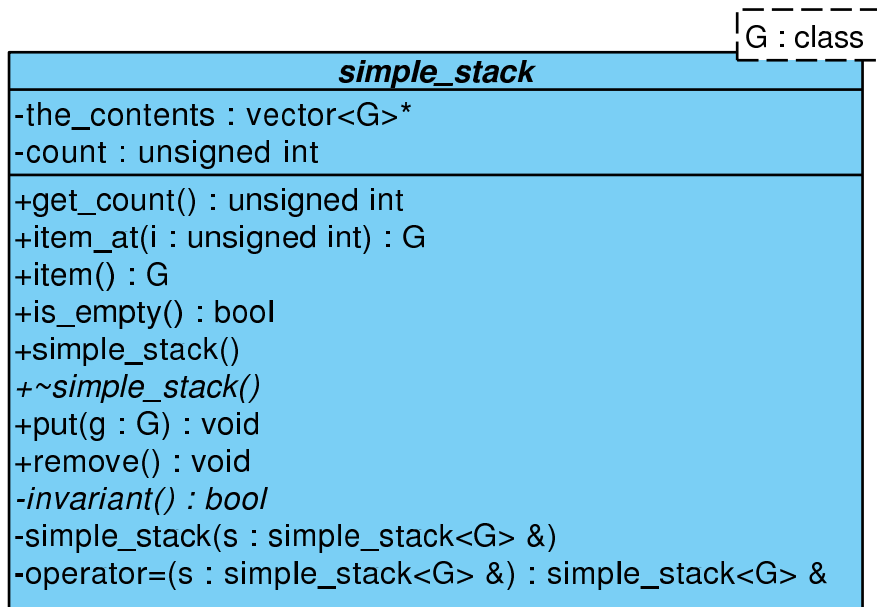
    // Exercise contract violation
    s.remove();
    cout << s.item() << endl;

    cout << "----- Testende -----" << endl << endl;
}

```

### A.2.3. Destruktor, Kopierkonstruktor und Wertzuweisung

Visual Paradigm for UML Standard Edition(University of Wuppertal)



```

...
#define EIFFEL_DOEND
#define EIFFEL_CHECK CHECK_ALL
...
#include <iostream>
#include <vector>
#include <eiffel.h>
#include <nana.h>
...
class simple_stack{

```

```

private:

    vector<G>* the_contents;
    unsigned int count;

public:

    //////////// basic queries:

    unsigned int get_count() const; // number of items in stack

    G item_at(unsigned int i) const;

    //////////// class invariant:

private:
    virtual bool invariant() const;
public:

    //////////// derived queries:

    G item() const; // the item on the top of stack

    bool is_empty() const; // does the stack contain no items?

    //////////// constructors:

    simple_stack();

    virtual ~simple_stack(); // notwendig wegen new in Konstruktor

private: // default copy-Konstruktor
        // und default operator=
        // fragwuerdig

    simple_stack<G>(const simple_stack<G>& s);
    simple_stack<G>& operator=(const simple_stack<G>& s);

public:

    //////////// (pure) modifiers:

    void put(G g); // Push 'g' onto the stack

```

```

void remove();                // delete the top item;

};
...
template<class G>
G simple_stack<G>::item_at(unsigned int i) const {
    REQUIRE( /* big enough */    i >= 1);
    REQUIRE( /* small enough */  i <= get_count() );
    ...
};
...
template<class G>
unsigned int simple_stack<G>::get_count() const{
    ...
};
...
template<class G>
bool simple_stack<G>::invariant() const {
    ...
};
...
template<class G>
G simple_stack<G>::item() const{           // the item on the top of stack
    REQUIRE( /* stack not empty */ get_count() > 0 );
    ...
    ENSURE( /* consistend with item_at() */  result == item_at(get_count()) );
    ...
};
...
template<class G>
bool simple_stack<G>::is_empty() const{    // does the stack contain no items?
    ...
    ENSURE( /* consistend with count */      result == (get_count() == 0) );
    ...
};
...
template<class G>
simple_stack<G>::simple_stack(){
    ...
    ENSURE( /* stack is empty */           (get_count() == 0));
    ENSURE(invariant());
};
...

```

```

template<class G>
simple_stack<G>::~~simple_stack<G>(){
    REQUIRE (/* pointer not null */ the_contents != 0);
    ...
};
...
template<class G>
void simple_stack<G>::put(G g)    // Push 'g' onto the stack
DO
    ID(int count_old = get_count());
    ID(vector<G> old(*the_contents));
    ...
    ENSURE(/* count incremented */      get_count() == count_old + 1 );
    ENSURE(/* g on top */                item_at(get_count()) == g );
    ENSURE(/* old contents unchanged */
            A(int k=1, k<get_count(), k++, item_at(k)== old.at(k-1) ));
    ...
END;
...
template<class G>
void simple_stack<G>::remove()    // delete the top item;
DO
    REQUIRE(/* stack not empty */      get_count() > 0 );
    ID(int count_old = get_count());
    ID(vector<G> old(*the_contents));
    ...
    ENSURE( /* count decremented */      get_count() == count_old - 1 );
    ENSURE( /* consistency with item() */ item_at(get_count()) == old.at(count_old-2) );
    ENSURE( /* old contents unchanged */
            A(int k=1, k<get_count(), k++, item_at(k)== old.at(k-1) ));
END;
...
int main(){
    ...
    s.put(10);
    cout << s.item() << endl;
    s.put(20);
    cout << s.item() << endl;
    s.put(30);
    cout << s.item() << endl;

    cout << endl;

    // Exercise 'remove'

```

```
cout << s.item() << endl;
s.remove();
cout << s.item() << endl;
s.remove();
cout << s.item() << endl;

// Test copy-Konstruktor ...
//
// simple_stack<long> s2(s);
simple_stack<long> s3;
// s3 = s;

cout << "----- Testende -----" << endl << endl;

}
```

Voller Code: [simple\\_stack4.cc](#)

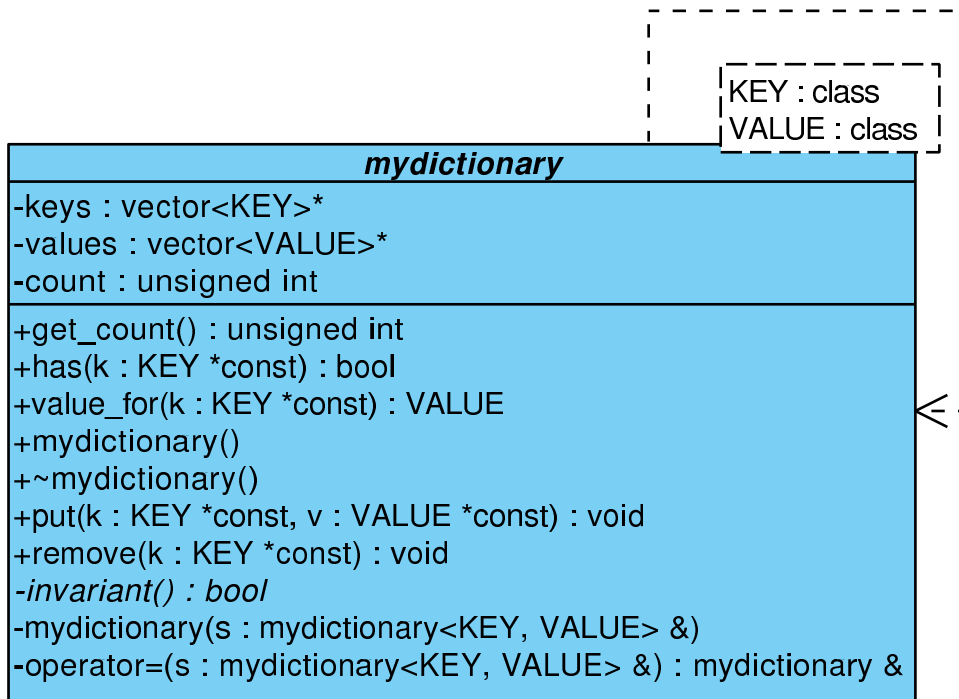
## A.3. Applying the Six Principles

PbC-Regeln:

- Vermeide statusändernde Methoden, die einen Wert liefern! (Observer **oder** Modifikator)
- Unterscheide grundlegende von abgeleiteten (redundanten) Observatoren.
- Schreibe für jeden abgeleiteten Observer eine Nachbedingung mit Hilfe der (aller) grundlegenden Observatoren.
- Schreibe für jeden Modifikator Nachbedingungen, die mit Hilfe der (aller) grundlegenden Observatoren den Inhalt des Klassenexemplars nach Methodenende in seiner Relation zum Exemplarinhalt bei Methodenbeginn **exakt** beschreiben. Nutze implizite (als Kommentar) oder explizite Frame-Bedingungen.
- Schreibe für alle Methoden Vorbedingungen (an Parameter bzw. Exemplarinhalt).
- Schreibe und benutze Invarianten, die gültige von ungültigen Exemplaren trennen.

### A.3.1. Design und Contracts

Visual Paradigm for UML Standard Edition(University of Wuppertal)



```
...
#define EIFFEL_DOEND
#define EIFFEL_CHECK CHECK_ALL
```

```

...
#include <iostream>
#include <vector>
#include <eiffel.h>
#include <nana.h>
...
class mydictionary{

    vector<KEY>* keys;
    vector<VALUE>* values;
    unsigned int count;

public:
    //////////////// basic queries:

    unsigned int get_count() const;        // number of key/value-pairs in dict.

    bool has(const KEY *const k) const;    // key in dictionary?

    VALUE value_for(const KEY *const k) const; // lookup value for key

    //////////////// class invariant:
private:
    virtual bool invariant() const;
public:
    //////////////// derived queries:

    // not yet necessary

    //////////////// constructors and destructors:

    mydictionary();

    ~mydictionary();

    //////////////// deactivate default copy constructor and operator=

private:
    mydictionary(const mydictionary<KEY, VALUE>& s);
    mydictionary& operator=(const mydictionary<KEY, VALUE>& s);
public:

    //////////////// (pure) modifiers

```



```

void put(const KEY *const k, const VALUE *const v);
                                         // put key/value-pair in dict.

void remove(const KEY *const k);          // remove key/value-pair

};
...
template<class KEY, class VALUE>
unsigned int mydictionary<KEY, VALUE>::get_count() const{
    ...
};
...
template<class KEY, class VALUE>
bool mydictionary<KEY, VALUE>::has(const KEY *const k) const {
    REQUIRE( /* key exists */      k != 0);
    ...
    ENSURE( /* consistent with count */ (get_count() != 0) || ! result);
    ...
};
...
template<class KEY, class VALUE>
VALUE mydictionary<KEY, VALUE>::value_for(const KEY *const k) const{
    REQUIRE(/* key exists */      k != 0);
    REQUIRE(/* key in dict. */    has(k));
    ...
};
...
template <class KEY, class VALUE>
bool mydictionary<KEY, VALUE>::invariant() const{
    ...
};
...
template<class KEY, class VALUE>
mydictionary<KEY, VALUE>::mydictionary(){
    ...
    ENSURE(invariant());
};
...
template<class KEY, class VALUE>
mydictionary<KEY, VALUE>::~~mydictionary(){
    REQUIRE(/* keys exist */      keys != 0);
    REQUIRE(/* values exist */    values != 0);
    ...
};

```

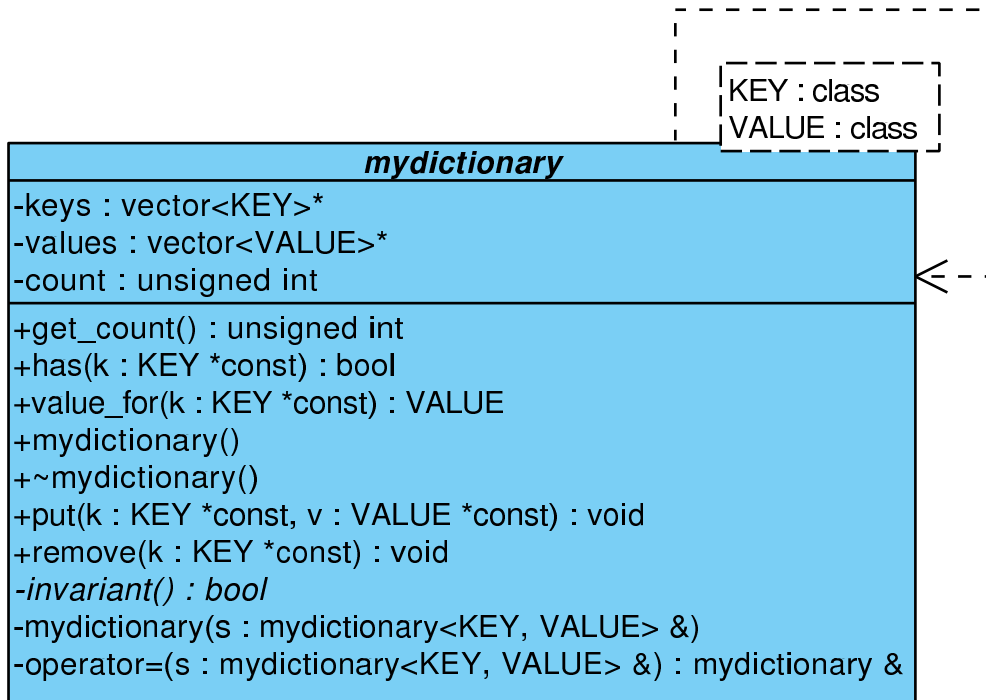
```

...
template<class KEY, class VALUE>
void mydictionary<KEY, VALUE>::put(const KEY *const k, const VALUE *const v)
DO
    ID(unsigned int count_old=get_count());
    REQUIRE(/* key exists */      k != 0);
    REQUIRE(/* key not in dict. */ ! has(k));
    ...
    ENSURE(/* count incremented */ get_count() == count_old + 1);
    ENSURE(/* key in dict. */      has(k) );
    ENSURE(/* correct value */     value_for(k) == *v);
END;
...
template<class KEY, class VALUE>
void mydictionary<KEY, VALUE>::remove(const KEY *const k)
DO
    ID(unsigned int count_old = get_count());
    REQUIRE(/* key exists */      k != 0);
    REQUIRE(/* key in dict. */    has(k));
    ...
    ENSURE(/* count decremented */ get_count() == count_old - 1);
    ENSURE(/* key not in dict. */ ! has(k));
    ...
END;
...
int main(){
    ...
}

```

## A.3.2. Implementierung und Tests

Visual Paradigm for UML Standard Edition(University of Wuppertal)



```
// dictionary2.cc
//
// g++ -g -o dictionary2 dictionary2.cc -I$HOME/include -L$HOME/lib -lnana
//
//
// eventuell mit
//                -DWITHOUT_NANA
// und/oder
//                -DNDEBUG
//
// und/oder
//                -DEIFFEL_CHECK=CHECK_....
//
// -----
// evtl. myexcept.o mit angeben
// -----
//
// oder: nana-c++lg simple_stack4.cc
//
```

```
#define EIFFEL_DOEND
```

```

#ifndef EIFFEL_CHECK
#define EIFFEL_CHECK CHECK_ALL
//
// CHECK_LOOP Makros CHECK() und folgende
// CHECK_INVARIANT Makros INVARIANT() und folgende
// CHECK_ENSURE Methode invariant() und folgende
// CHECK_REQUIRE Nachbedingungen und folgende
// CHECK_NO Vorgedingungen
#endif

#include <iostream>
#include <vector>

#include <eiffel.h>
#include <nana.h>

using namespace std;

////////// class declaration //////////

template<class KEY, class VALUE>
class mydictionary{

    vector<KEY>* keys;
    vector<VALUE>* values;
    unsigned int count;

public:
    //////////// basic queries:

    unsigned int get_count() const; // number of key/value-pairs in dict.

    bool has(const KEY *const k) const; // key in dictionary?

    VALUE value_for(const KEY *const k) const; // lookup value for key

    //////////// class invariant:
private:
    virtual bool invariant() const;
public:
    //////////// derived queries:

    // not yet necessary

```

```

////////// constructors and destructors:

mydictionary();

~mydictionary();

////////// deactivate default copy constructor and operator=

private:
    mydictionary(const mydictionary<KEY, VALUE>& s);
    mydictionary& operator=(const mydictionary<KEY, VALUE>& s);
public:

    ////////// (pure) modifiers

    void put(const KEY *const k, const VALUE *const v);
                                                // put key/value-pair in dict.

    void remove(const KEY *const k);          // remove key/value-pair

};

////////// class definition //////////

////////// basic queries:

template<class KEY, class VALUE>
unsigned int mydictionary<KEY, VALUE>::get_count() const{
    return count;
};

template<class KEY, class VALUE>
bool mydictionary<KEY, VALUE>::has(const KEY *const k) const {

    REQUIRE( /* key exists */      k != 0);

    unsigned int i = 0;
    do {
        i++;
    }while((i <= count) && (keys->at(i-1) != *k) );

    bool result = (i <= count) && (keys->at(i-1) == *k);
    ENSURE( /* consistent with count */ (get_count() != 0) || ! result);
}

```

```

    return result;
};

template<class KEY, class VALUE>
VALUE mydictionary<KEY, VALUE>::value_for(const KEY *const k) const{
    REQUIRE(/* key exists */      k != 0);
    REQUIRE(/* key in dict. */    has(k));

    unsigned int i = 0;
    do {
        i++;
    }while(keys->at(i-1) != *k);
    return values->at(i-1);
};

////////// class invariant:

template <class KEY, class VALUE>
bool mydictionary<KEY, VALUE>::invariant() const{

    return (get_count() >= 0) && (keys != 0) && (values != 0);
};

////////// constructors and destructors:

template<class KEY, class VALUE>
mydictionary<KEY, VALUE>::mydictionary(){

    count = 0;
    keys = new vector<KEY>(100);
    values = new vector<VALUE>(100);
    ENSURE(invariant());
};

template<class KEY, class VALUE>
mydictionary<KEY, VALUE>::~~mydictionary(){

    REQUIRE(/* keys exist */      keys != 0);
    REQUIRE(/* values exist */    values != 0);
    delete keys;
    delete values;
};

```

```
////////// (pure) modifiers
```

```
template<class KEY, class VALUE>
void mydictionary<KEY, VALUE>::put(const KEY *const k, const VALUE *const v)
DO
  ID( unsigned int count_old=get_count());
  REQUIRE(/* key exists */      k != 0);
  REQUIRE(/* key not in dict. */ ! has(k));

  count++;
  keys->at(count-1) = *k;
  values->at(count-1) = *v;

  ENSURE(/* count incremented */  get_count() == count_old + 1);
  ENSURE(/* key in dict. */      has(k) );
  ENSURE(/* correct value */     value_for(k) == *v);
END;
```

```
template<class KEY, class VALUE>
void mydictionary<KEY, VALUE>::remove(const KEY *const k)
DO
  ID(unsigned int count_old = get_count());
  REQUIRE(/* key exists */      k != 0);
  REQUIRE(/* key in dict. */    has(k));

  unsigned int i = 0;
  do {
    i++;
  }while(keys->at(i-1) != *k);
  CHECK(i <= count);
  if (i < count){
    keys->at(i-1) = keys->at(count-1);
    values->at(i-1) = keys->at(count-1);
  };
  count--;

  ENSURE(/* count decremented */  get_count() == count_old - 1);
  ENSURE(/* key not in dict. */ ! has(k));
  // REQUIRE precondition for value_for() is false for k
END;
```

```
////////// class test main() program //////////
```

```

int main(){

    cout << "Test der Klasse dictionary: -----" << endl;

    mydictionary<string, string> d;

    string k("Denver"); string v("Colorado");
    d.put(&k, &v);
    k = "London"; v = "Ontario";
    d.put(&k, &v);
    k = "Austin"; v = "Texas";
    d.put(&k, &v);
    k = "Boston"; v = "Massachusetts";
    d.put(&k, &v);
    k = "Mobile"; v = "Alabama";
    d.put(&k, &v);

    cout << endl << "List of 5 pairs:" << endl;
    {
        string k1("Denver");
        if (d.has(&k1)) cout << k1 << " " << d.value_for(&k1) << endl;
        string k2("London");
        if (d.has(&k2)) cout << k2 << " " << d.value_for(&k2) << endl;
        string k3("Austin");
        if (d.has(&k3)) cout << k3 << " " << d.value_for(&k3) << endl;
        string k4("Boston");
        if (d.has(&k4)) cout << k4 << " " << d.value_for(&k4) << endl;
        string k5("Mobile");
        if (d.has(&k5)) cout << k5 << " " << d.value_for(&k5) << endl;
    };

    string l2("Mobile");
    d.remove(&l2);
    cout << endl << "List of " << d.get_count() << " pairs, last removed:" << endl;
    {
        string k1("Denver");
        if (d.has(&k1)) cout << k1 << " " << d.value_for(&k1) << endl;
        string k2("London");
        if (d.has(&k2)) cout << k2 << " " << d.value_for(&k2) << endl;
        string k3("Austin");
        if (d.has(&k3)) cout << k3 << " " << d.value_for(&k3) << endl;
        string k4("Boston");
        if (d.has(&k4)) cout << k4 << " " << d.value_for(&k4) << endl;
        string k5("Mobile");
    }
}

```



```

    if (d.has(&k5)) cout << k5 << " " << d.value_for(&k5) << endl;
};

string l1("Denver");
d.remove(&l1);
cout << endl << "List of " << d.get_count() << " pairs, 1st removed:" << endl;
{
    string k1("Denver");
    if (d.has(&k1)) cout << k1 << " " << d.value_for(&k1) << endl;
    string k2("London");
    if (d.has(&k2)) cout << k2 << " " << d.value_for(&k2) << endl;
    string k3("Austin");
    if (d.has(&k3)) cout << k3 << " " << d.value_for(&k3) << endl;
    string k4("Boston");
    if (d.has(&k4)) cout << k4 << " " << d.value_for(&k4) << endl;
    string k5("Mobile");
    if (d.has(&k5)) cout << k5 << " " << d.value_for(&k5) << endl;
};

string l3("Austin");
d.remove(&l3);
cout << endl << "List of " << d.get_count() << " pairs, middle removed:" << endl;
{
    string k1("Denver");
    if (d.has(&k1)) cout << k1 << " " << d.value_for(&k1) << endl;
    string k2("London");
    if (d.has(&k2)) cout << k2 << " " << d.value_for(&k2) << endl;
    string k3("Austin");
    if (d.has(&k3)) cout << k3 << " " << d.value_for(&k3) << endl;
    string k4("Boston");
    if (d.has(&k4)) cout << k4 << " " << d.value_for(&k4) << endl;
    string k5("Mobile");
    if (d.has(&k5)) cout << k5 << " " << d.value_for(&k5) << endl;
};

// d.remove(&string("Austin"));

d.put(&k, &v);
cout << endl << "List of " << d.get_count() << " pairs after put:" << endl;
{
    string k1("Denver");
    if (d.has(&k1)) cout << k1 << " " << d.value_for(&k1) << endl;
    string k2("London");
    if (d.has(&k2)) cout << k2 << " " << d.value_for(&k2) << endl;
};

```

```

string k3("Austin");
if (d.has(&k3)) cout << k3 << " " << d.value_for(&k3) << endl;
string k4("Boston");
if (d.has(&k4)) cout << k4 << " " << d.value_for(&k4) << endl;
string k5("Mobile");
if (d.has(&k5)) cout << k5 << " " << d.value_for(&k5) << endl;
};

// d.put(&k, &v);

cout << "----- Testende -----" << endl;

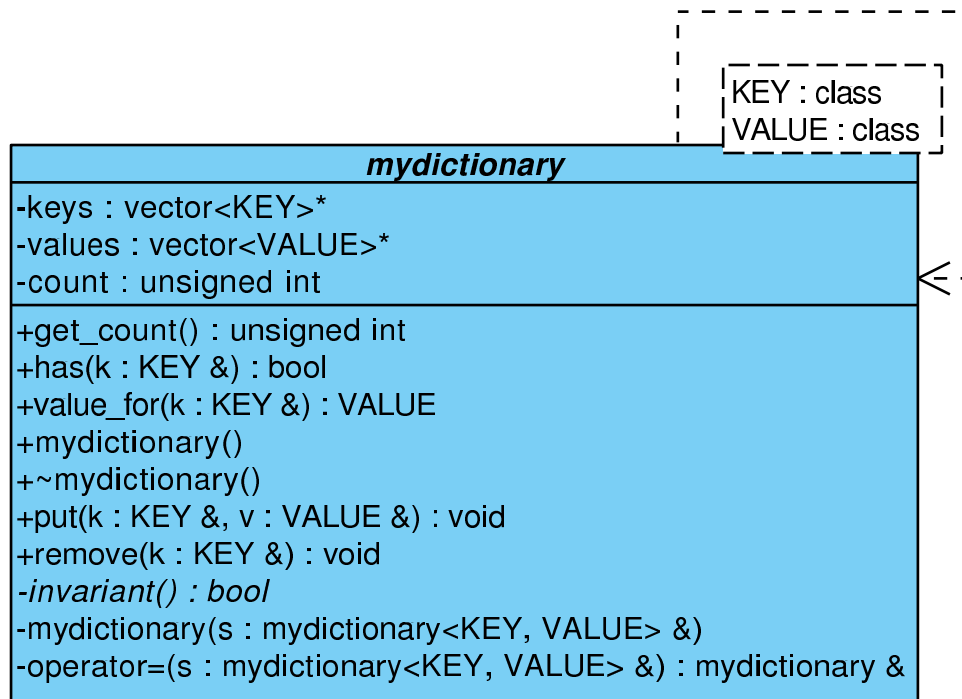
}

```

Zum Download: [dictionary2.cc](http://dictionary2.cc)

### A.3.3. old-Wert durch Kopie in Form eines geeigneten STL-Container-Exemplars

Visual Paradigm for UML Standard Edition(University of Wuppertal)



```

...
#define EIFFEL_DOEND
#define EIFFEL_CHECK CHECK_ALL
...
#include <iostream>

```

```

#include <vector>
#include <set>
#include <eiffel.h>
#include <nana.h>
...
// private Spezifikations-Hilfsmethoden: fuer STL-Container
template <class T>
set<T> operator+(const set<T>& s, const T& e){
    ...
};
...
template <class T>
set<T> operator-(const set<T>& s, const T& e){
    ...
};
...
class mydictionary{

    vector<KEY>* keys;
    vector<VALUE>* values;
    unsigned int count;

public:
    //////////////// basic queries:

    unsigned int get_count() const;        // number of key/value-pairs in dict.

    bool has(const KEY& k) const;          // key in dictionary?

    VALUE value_for(const KEY& k) const;   // lookup value for key

    //////////////// class invariant:
private:
    virtual bool invariant() const;
public:

    //////////////// derived queries:
    // not yet necessary

    //////////////// constructors and destructors:

    mydictionary();

    ~mydictionary();

```

```

////////// deactivate default copy constructor and operator=

private:
    mydictionary(const mydictionary<KEY, VALUE>& s);
    mydictionary& operator=(const mydictionary<KEY, VALUE>& s);
public:

    //////////// (pure) modifiers

    void put(const KEY& k, const VALUE& v);
                                                    // put key/value-pair in dict.

    void remove(const KEY& k);
                                                    // remove key/value-pair

};
...

/// basic queries:

template<class KEY, class VALUE>
unsigned int mydictionary<KEY, VALUE>::get_count() const{
    ...
};
...
template<class KEY, class VALUE>
bool mydictionary<KEY, VALUE>::has(const KEY& k) const {
    ...
    ENSURE( /* consistent with count */ (get_count() != 0) || ! result);
    ...
};
...
template<class KEY, class VALUE>
VALUE mydictionary<KEY, VALUE>::value_for(const KEY& k) const{
    REQUIRE(/* key in dict. */ has(k));
    ...
};
...

/// Invariante:

template <class KEY, class VALUE>
bool mydictionary<KEY, VALUE>::invariant() const{
    ...

```

```

};
...
///< Konstruktoren/Destruktoren:

template<class KEY, class VALUE>
mydictionary<KEY, VALUE>::mydictionary(){
    ...
    ENSURE(invariant());
    ENSURE(count == 0);
};
...
template<class KEY, class VALUE>
mydictionary<KEY, VALUE>::~~mydictionary(){
    REQUIRE(/* keys exist */      keys != 0);
    REQUIRE(/* values exist */    values != 0);
    ...
};
...
///< Modifikatoren:

template<class KEY, class VALUE>
void mydictionary<KEY, VALUE>::put(const KEY& k, const VALUE& v)
DO
    REQUIRE(/* key not in dict. */  ! has(k));
    ID(unsigned int count_old=get_count());
    ID(set<KEY> old_keys(keys->begin(), keys->begin()+count_old));
    ...
    ENSURE(/* count incremented */  get_count() == count_old + 1);
    ENSURE(/* key in dict. */       has(k) );
    ENSURE(/* correct value */      value_for(k) == v);
    ID(set<KEY> new_keys(keys->begin(), keys->begin()+count));
    ENSURE(old_keys + k == new_keys);
    ...
END;
...
template<class KEY, class VALUE>
void mydictionary<KEY, VALUE>::remove(const KEY& k)
DO
    REQUIRE(/* key in dict. */      has(k));
    ID(unsigned int count_old = get_count());
    ID(set<KEY> old_keys(keys->begin(), keys->begin()+count_old));
    ...
    ENSURE(/* count decremented */  get_count() == count_old - 1);
    ENSURE(/* key not in dict. */    ! has(k));

```

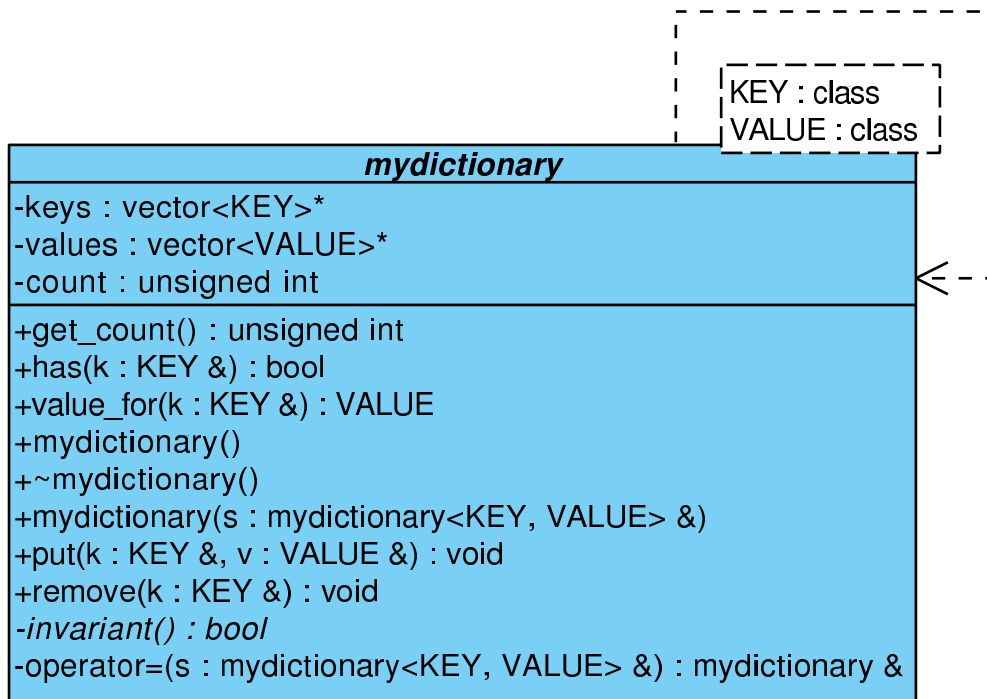
```

...
ID(set<KEY> new_keys(keys->begin(),keys->begin()+count));
ENSURE(old_keys - k == new_keys);
...
END;
...
int main(){
    ...
}

```

### A.3.4. old-Wert durch den Kopierkonstruktor

Visual Paradigm for UML Standard Edition(University of Wuppertal)



Mit Hilfe des Kopierkonstruktors

```

...
template<class KEY, class VALUE>
mydictionary<KEY, VALUE>::mydictionary(const mydictionary<KEY, VALUE>& s){
    count = s.count;
    keys = new vector<KEY>(100);
    values = new vector<VALUE>(100);
    for (int i=1; i<=count; i++){
        keys->at(i-1) = s.keys->at(i-1);
        values->at(i-1) = s.values->at(i-1);
    };
    ENSURE(count == s.count);
}

```

```
    ENSURE(A(int i=1, i<=count, i++, keys->at(i-1) == s.keys->at(i-1)));
    ENSURE(A(int i=1, i<=count, i++, values->at(i-1) == s.values->at(i-1)));
    ENSURE(invariant());
};
...
```

können Sie den alten Exemplarwert vollständig in einem Stück als konstante Variable memorieren und in den Nachbedingungen aller Modifikatoren benutzen.

AUFGABE: Ändern Sie `dictionary4.cc` so ab, dass `put()` und `remove()` so spezifiziert wird!

Es ist noch unschön, dass die Nachbedingungen des Kopierkonstruktors auf die Implementierungsdetails Bezug nehmen. Deshalb:

### A.3.5. Redesign

Ein nächster Schritt sollte die Einführung eines neuen grundlegenden Observators `set<KEY> keys()` sein. Führen Sie die notwendigen Änderungen durch!

Alternativ könnte man einen Iterator in der neuen Containerklasse `mydictionary` implementieren (siehe folgender Abschnitt).

## **A.4. Immutable Lists**

... entfällt in C++ wegen der Existenz der STL

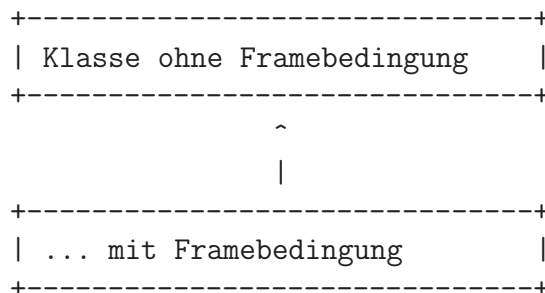
## **A.5. Using Immutable Lists**

... entfällt in C++ wegen der Existenz der STL



## Leitlinien:

- Nutze technische Einschränkungen, wo immer erforderlich: z.B. Zeiger `!= 0`, nicht-leere Container, nichtidentische Exemplare, ...
- In Vorbedingungen genutzte Observatoren sollten effizient berechnet werden. Notfalls führe neue effiziente abgeleitete Observatoren ein und benutze sie in den Vorbedingungen. Die neuen effizienten Observatoren benötigen Nachbedingungen, die ihre Konsistenz zu den grundlegenden Observatoren sicherstellen.
- Attribute haben keine Nachbedingungen. Benutze deshalb die Klasseninvariante für Contracts (oder Nachbedingungen von get-Methoden).
- Nachbedingungen von virtuellen Methoden sollten die Form `ENSURE (!Vorbedingung || Nachbedingung)` haben; Invarianten sollten `protected` als `virtual bool invariant() const` deklariert werden.
- Nutze die Vererbung:



um dem wiederverwendenden Nutzer die Wahl zwischen der Verwendung der effizienten oberen oder sicheren unteren Klasse zu überlassen.

- Nutze die Vererbung analog z.B. für:

Klasse ohne Framebedingungen

Klasse mit Framebedingungen

... mit unzugänglichen (`private`) Observatoren,  
die weiter oben lediglich für die Contracts  
genutzt werden

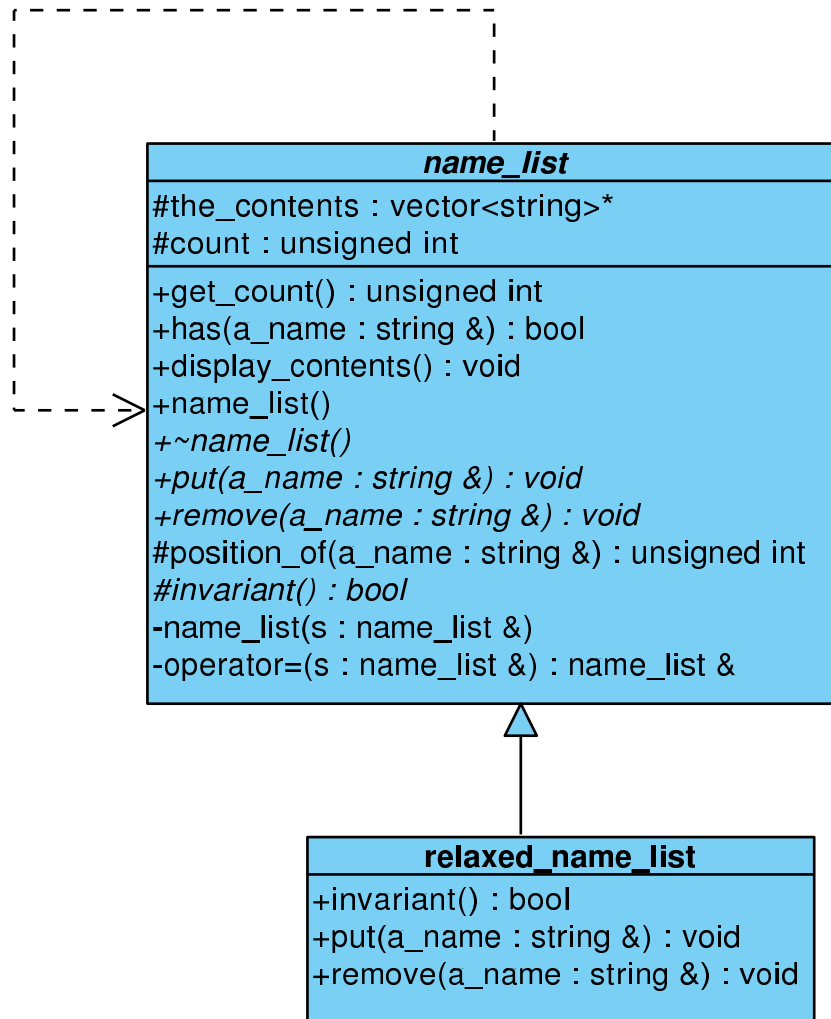
... mit benutzerfreundlichen Methodenvarianten  
ohne Vorbedingungen

...

## A.6. Subcontracting in Design by Contract in Nana

### A.6.1. name\_list-Design (Subcontracting)

Visual Paradigm for UML Standard Edition(University of Wuppertal)



```
...
#define EIFFEL_DOEND
#define EIFFEL_CHECK CHECK_ALL
...
#include <iostream>
#include <vector>
```

```

#include <eiffel.h>
#include <nana.h>
...
class name_list{

protected:

    vector<string>* the_contents;
    unsigned int count;

    // private support function

    unsigned int position_of(const string& a_name) const;

public:

    //////////// basic queries:

    unsigned int get_count() const;           // number of items in stack

    bool has(const string& a_name) const;

    //////////// class invariant:

protected:
    virtual bool invariant() const;
public:

    //////////// derived queries:

    void display_contents() const;           // print contents of list to cout

    //////////// constructors:

    name_list();

    virtual ~name_list();                    // notwendig wegen new in Konstruktor

private:                                     // disable default methods

    name_list(const name_list& s);
    name_list& operator=(const name_list& s);

public:

```

```

////////// (pure) modifiers:

    virtual void put(const string& a_name);        // Push a_name into list

    virtual void remove(const string& a_name);    // delete a_name in list

};
...
unsigned int name_list::position_of(const string& a_name) const{
    ...
    ENSURE(/* a_name in list */
           ((1<=result)&&(result<=get_count())&&
            (the_contents->at(result-1)==a_name)) ||
           /* otherwise: */
           (0 == result)
           );
    ...
};
...
unsigned int name_list::get_count() const{        // number of items in stack
    ...
    ENSURE(result == count);
    ...
};
bool name_list::has(const string& a_name) const{
    ...
    ENSURE((get_count()>0) || !result);
    ...
};
...
bool name_list::invariant() const{
    ...
};
...
void name_list::display_contents() const{        // print contents of list to cout
    ...
};
...
name_list::name_list(): count(0), the_contents(new vector<string>(100)){
    ENSURE(invariant());
};
name_list::~name_list(){                          // notwendig wegen new in Konstruktor
    ...
};

```

```

...
void name_list::put(const string& a_name)    // Push a_name into list
DO
    ID(bool pre = !has(a_name));
    ID( unsigned int count_old = get_count());
    REQUIRE(/* name not in list */    pre );
    ...
    ENSURE(has(a_name));
    ENSURE((!pre) || (get_count() == count_old + 1));
    ...
END;
void name_list::remove(const string& a_name) // delete a_name in list
DO
    ID(bool pre(has(a_name)));
    ID( unsigned int count_old = get_count());
    REQUIRE(/* name in list */    has(a_name));
    ...
    ENSURE(! has(a_name));
    ENSURE((!pre) || (get_count() == count_old -1));
    ...
END;
...
class relaxed_name_list : public name_list{

public:

    //////////// invariant:

    virtual bool invariant() const;

    //////////// (pure) modifiers: (redefined)

    virtual void put(const string& a_name);    // Push a_name into list

    virtual void remove(const string& a_name);    // delete a_name in list

};
...
bool relaxed_name_list::invariant() const{
    ...
};
...
void relaxed_name_list::put(const string& a_name)    // Push a_name into list
DO

```

```

ID(bool pre_parent(!has(a_name)));
ID(unsigned int count_old = get_count());
REQUIRE(/* nothing */ true);          // pre_parent || has(a_name)
...
ENSURE(has(a_name));
ENSURE(!pre_parent || (get_count() == count_old + 1)); // &&
ENSURE( pre_parent  || (get_count() == count_old));
...
END;
void relaxed_name_list::remove(const string& a_name) // delete a_name in list
DO
    ID(bool pre_parent(has(a_name)));
    ID(unsigned int count_old = get_count());
    REQUIRE(/* nothing */ true);          // pre_parent || !has(a_name)
    ...
    ENSURE(! has(a_name));
    ENSURE(!pre_parent || (get_count() == count_old -1)); // &&
    ENSURE( pre_parent  || (get_count() == count_old));
    ...
END;
...
int main(){
    ...
}

```

Zum Download: [name.list.cc](http://name.list.cc)

## A.6.2. Implementierung und Tests

... der Contracts und der Prototypinstallation:

```

...
unsigned int name_list::position_of(const string& a_name) const{

    unsigned int index(1);
    unsigned int result;
    for(; (index<=count) && (the_contents->at(index-1)!=a_name); index++);
    if (index <= count)
        result = index;
    else
        result = 0;

    ENSURE(/* a_name in list */)

```

```

        ((1<=result)&&(result<=get_count())&&
         (the_contents->at(result-1)==a_name)) ||
        /* otherwise: */
        (0 == result)
    );
    return result;
};
...
////////// basic queries:

unsigned int name_list::get_count() const{    // number of items in stack

    unsigned int result = count;
    return result;
};

bool name_list::has(const string& a_name) const{

    bool result = (position_of(a_name) > 0);
    ENSURE((get_count()>0) || !result);    // consistency
    return result;
};
...

////////// class invariant:

bool name_list::invariant() const{
    return (count >= 0) && (the_contents != 0);
};

...
////////// derived queries:

void name_list::display_contents() const{    // print contents of list to cout

    cout << endl << "Anzahl der Listenelemente: " << count << endl;
    for (unsigned int i=1; i<=count; i++)
        cout << " " << the_contents->at(i-1);
    cout << endl << endl;
    // ENSURE('Drucke alle Namen in Name_list');
};

////////// constructors:

name_list::name_list(): count(0), the_contents(new vector<string>(100)){

```

```

        ENSURE(invariant());
};

name_list::~name_list() {                               // notwendig wegen new in Konstruktor
    delete the_contents;
};

////////// (pure) modifiers:

void name_list::put(const string& a_name)    // Push a_name into list
DO
    ID(bool pre(!has(a_name)));
    ID(unsigned int count_old = get_count());

    REQUIRE(/* name not in list */    pre);

    count++;
    the_contents->at(count-1) = a_name;

    ENSURE(has(a_name));
    ENSURE( (!pre) || (get_count() == count_old + 1));
    // ENSURE: Fuer alle s in list_old: has(s)
    // ENSURE: Fuer alle s in list:      (s == a_name) || s in list_old
END;
...
////////// class test main() program //////////

int main(){

    cout << "\nTest der Klasse name_list: -----" << endl;

    { name_list s;
      s.display_contents();

      s.put("Richard"); s.display_contents();
      //s.put("Richard"); s.display_contents(); // Test fuer pre-Verletzung
      s.put("Helen"); s.display_contents();
      s.put("Yu"); s.display_contents();
      s.put("Jim"); s.display_contents();
      s.put("Chen"); s.display_contents();

      ...
    }
}

```

Zum Download: [name\\_list2.cc](#)



### A.6.3. Mit Frameregeln

```
unsigned int name_list::position_of(const string& a_name) const{

    unsigned int index(1);
    unsigned int result;
    for(; (index<=count) && (the_contents->at(index-1)!=a_name); index++);
    if (index <= count)
        result = index;
    else
        result = 0;
    ENSURE(!(result < 0));
    ENSURE(!(result >get_count()));
    ID(set<string> values(the_contents->begin(),the_contents->begin()+get_count()));
    ENSURE(/* a_name in list */
           ((1<=result)&&(result<=get_count())&&
            (the_contents->at(result-1)==a_name)) ||
           /* otherwise: */
           ( ((0 == result) && (values.find(a_name) == values.end()))))
    );
    return result;
};
...
unsigned int name_list::get_count() const{ // number of items in stack

    unsigned int result = count;
    CHECK(result == count);
    return result;
};

bool name_list::has(const string& a_name) const{

    bool result = (position_of(a_name) > 0);
    ENSURE((get_count()>0) || !result);
    ID(set<string> values(the_contents->begin(),
                        the_contents->begin()+get_count()));
    CHECK(result == (values.find(a_name) != values.end( )));
    return result;
};
...
void name_list::put(const string& a_name) // Push a_name into list
DO
    ID(bool pre);
    IS(pre = !has(a_name));
```

```

// DS($pre_false = has(a_name)); // funktioniert nicht
ID(set<string> values_old(the_contents->begin(),
                        the_contents->begin()+get_count()));
REQUIRE(/* name not in list */ pre);
DS($count_old = get_count());

count++;
the_contents->at(count-1) = a_name;

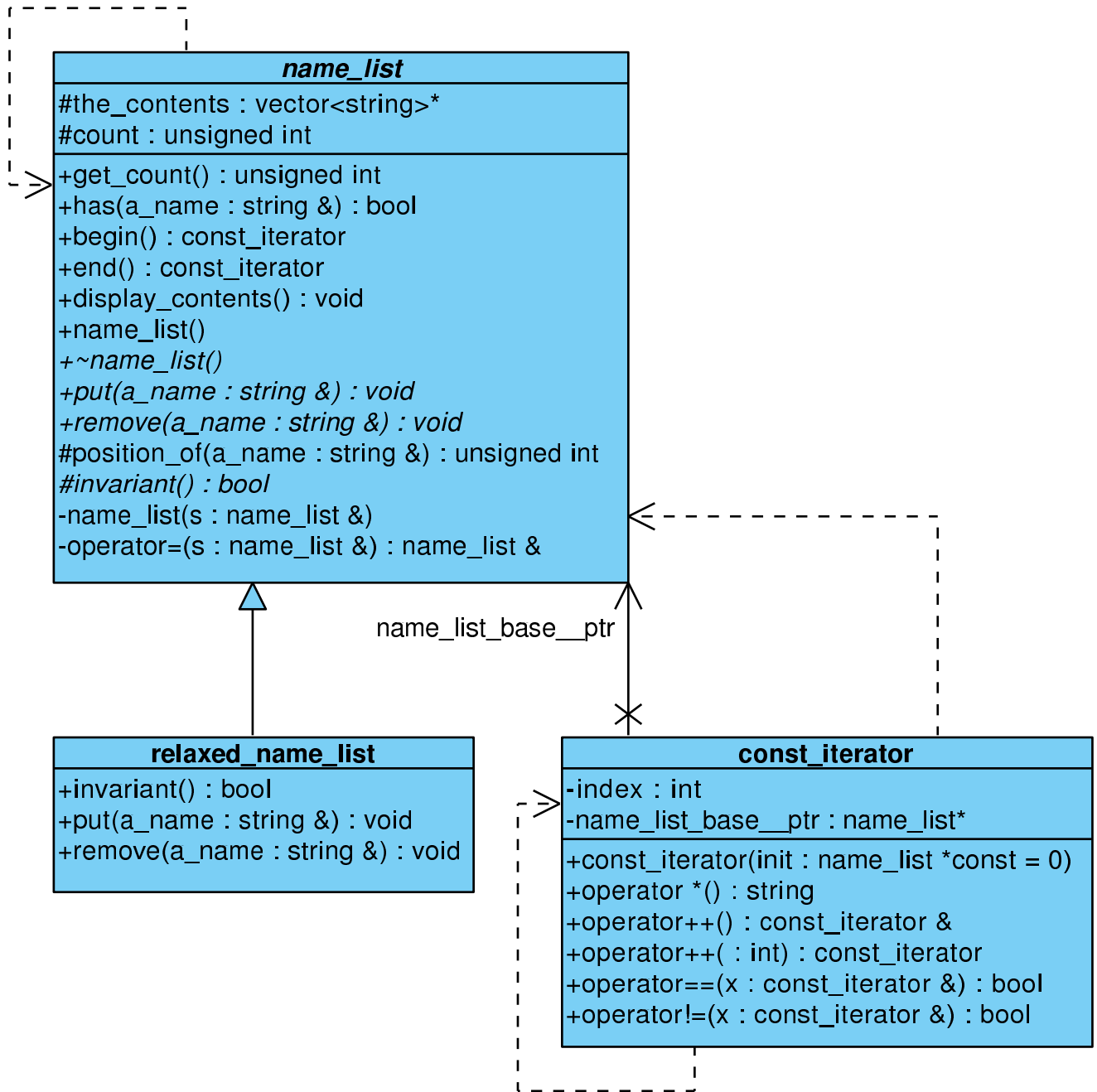
ENSURE(has(a_name));
DI( (!pre) || (this->get_count() == $count_old + 1));
ID(set<string> values(the_contents->begin(),
                    the_contents->begin()+get_count()));
IS(values_old.insert(a_name)); // stilistisch unschoen!
ENSURE(values_old == values);
END;
...

```

Zum Download: [name\\_list3.cc](#)

## A.6.4. Mit Iterator-Methode (Design)

Visual Paradigm for UML Standard Edition(University of Wuppertal)



```

...
class name_list{
...
    //////////// Iteratoren:

    class const_iterator;
  
```

```

friend class const_iterator;

const_iterator begin() const;
const_iterator end() const;
...
}
...
////////// class name_list::const_iterator //////////

class name_list::const_iterator{

    int index;
    const name_list* name_list_base_ptr;

public:

    const_iterator(const name_list* const init = 0);

    string operator*() const;          // Zugriff auf Element am Iterator-Ort
    const_iterator& operator++();      // Praefix-Inkrement
    const_iterator operator++(int);    // Postfix-Inkrement

    bool operator==(const const_iterator& x) const; // Vergleich von Iteratoren
    bool operator!=(const const_iterator& x) const;

};

/// contract for Iterator: (as a basic query)

name_list::const_iterator::const_iterator(const name_list* const init){

    // ...
    // (*this)="first" element of name_list (*init) if (init != 0),
    // this is an iterator not in any name_list if (init == 0)
};

name_list::const_iterator name_list::begin() const {
    // return ...
    // returns const_iterator pointing to "first" element of name_list
};

name_list::const_iterator name_list::end() const {
    // return ...
    // returns const_iterator denoting to be not any more in name_list
};

```

```

string name_list::const_iterator::operator*() const {
    // return ...
    // return element const_iterator is pointing to
};

name_list::const_iterator& name_list::const_iterator::operator++(){ // Praefix
    // return ...
    // increment position of const_iterator and return reference to this
    //      incremented const_iterator afterwards
};

name_list::const_iterator name_list::const_iterator::operator++(int){//Postfix
    // return ...
    // return copy of const_iterator and as a side effect increment position
    // of the actual const_iterator
};

bool name_list::const_iterator::operator==(const name_list::const_iterator& x) const{
    // return ...
    // return if const_iterator and x point to the same element in the
    // same name_list
};

bool name_list::const_iterator::operator!=(const name_list::const_iterator& x) const{
    bool result; // = ...
    ENSURE(!((*this)==x));          // Konsistenzbedingung
};
...

```

Zum Download: [name\\_list4.cc](#)

## A.6.5. Implementierung der Iterator-Methode

```

...
///< contract for Iterator: (as a basic query)

name_list::const_iterator::const_iterator(const name_list* const init){

    name_list_base_ptr = init;
    if ((name_list_base_ptr != 0)&&(name_list_base_ptr->count > 0)){
        index = 0;
    } else
        index = -1;
}

```

```

    if (index == -1) name_list_base_ptr = 0;
    // (*this)=="first" element of name_list (*init) if (init != 0),
    // this is an unique iterator not in any name_list if (init == 0)
};

name_list::const_iterator name_list::begin() const{
    return const_iterator(this);
    // returns const_iterator pointing to "first" element of name_list
};

name_list::const_iterator name_list::end() const{
    return const_iterator();
    // returns const_iterator denoting to be not any more in name_list
};

string name_list::const_iterator::operator*() const {
    REQUIRE(index != -1);
    return name_list_base_ptr->the_contents->at(index);
    // return element const_iterator is pointing to
    // name_list of const_iterators not changed
};

...

```

Zum Download: [name\\_list5.cc](#)

### A.6.6. Test des Iterators in display\_contents() und main()

```

...
////////// derived queries:

void name_list::display_contents() const{    // print contents of list to cout

    cout << endl << "Anzahl der Listenelemente: " << count << endl;
    for (name_list::const_iterator i = begin(); i != end(); i++)
        cout << " " << *i;
    cout << endl << endl;
    // ENSURE('Drucke alle Namen in Name_list');
};

...
////////// class test main() program //////////

int main(){

```

```

cout << "\nTest der Klasse name_list: -----" << endl;

{ name_list s;
s.display_contents();

s.put("Richard"); s.display_contents();
// s.put("Richard"); s.display_contents(); // Test fuer pre-Verletzung
s.put("Helen"); s.display_contents();
s.put("Yu"); s.display_contents();
s.put("Jim"); s.display_contents();
s.put("Chen"); s.display_contents();
s.put("Moirra"); s.display_contents();
...

```

Zum Download: [name\\_list6.cc](http://name_list6.cc)

### A.6.7. Qstl.h: Framebedingungen mit Hilfe eines Iterators

```

...

#include <eiffel.h>
#include <nana.h>
#include <Qstl.h>          //<<<////////////////////////////////////// NEU!

using namespace std;
...
unsigned int name_list::position_of(const string& a_name) const{

    unsigned int index(1);
    unsigned int result;
    for(; (index<=count) && (the_contents->at(index-1)!=a_name); index++);
    if (index <= count)
        result = index;
    else
        result = 0;

    ENSURE(/* a_name in list */
           ((1<=result)&&(result<=get_count())&&
            (the_contents->at(result-1)==a_name)) ||
           /* otherwise: */
           ((0 == result)&&(!EO(i,(*this),(*i)==a_name))));
    return result;
};

```

```

...
bool name_list::has(const string& a_name) const{

    bool result = (position_of(a_name) > 0);
    ENSURE((get_count()>0) || !result);           // Konsistenz
    ENSURE(result == EO(i, (*this), (*i)==a_name)); // Konsistenz
    return result;
};
...
////////// (pure) modifiers:

void name_list::put(const string& a_name)    // Push a_name into list
DO
    REQUIRE(/* name not in list */    !has(a_name));
    ID(set<string> contents_old(begin(),end()));
    ID(int count_old = get_count());
    ID(bool not_in_list = !has(a_name));

    count++;
    the_contents->at(count-1) = a_name;

    ENSURE(has(a_name));
    ENSURE( (!not_in_list) || (get_count() == count_old + 1));
    ID(set<string> contents(begin(),end()));
    IS(contents_old.insert(a_name));
    ENSURE(contents == contents_old);
END;
...

```

Zum Download: [name\\_list7.cc](http://name_list7.cc)



## A.6.8. Hilfsoperatoren für die STL

```
...
//////////////////////////////////// Hilfsfunktionen //////////////////////////////////////

template <class T>
set<T> operator+(const set<T>& s, const T& e){
    set<T> result(s);
    result.insert(e);
    return result;
};

template <class T>
set<T> operator-(const set<T>& s, const T& e){
    set<T> result(s);
    result.erase(e);
    return result;
};

...
void relaxed_name_list::remove(const string& a_name) // delete a_name in list
DO
    REQUIRE(/* nothing */ true);          // pre_parent || !has(a_name)
    ID(set<string> contents_old(begin(),end()));
    ID(int count_old = get_count());
    ID(bool pre_parent = has(a_name));

    if (has(a_name)){
        the_contents->at(position_of(a_name)-1) = the_contents->at(count-1);
        count--;
    };

    ENSURE(!has(a_name));
    ENSURE((!pre_parent) || (get_count() == count_old -1)); // &&
    ENSURE( pre_parent  || (get_count() == count_old));
    // Menge der Eintraege in list == Menge der Eintraege in list_old ohne a_name
    ID(set<string> contents(begin(),end()));
    ENSURE( pre_parent  || (contents == contents_old));;
    ENSURE((!pre_parent) || (contents == contents_old - a_name));
END;
...
```

Zum Download: [name\\_list9.cc](#)

## A.7. Neuformulierung: Regeln und Leitlinien für PbC in C++

1. Formuliere *grundlegende Observatoren*, die den Zustand eines Exemplars vollständig beschreiben können und eine *Klasseninvariante*, die gültige von ungültigen Exemplaren trennt. Falls grundlegende Observatoren mit Parametern existieren, so gib den zur vollständigen Exemplarbeschreibung nötigen Parameter-Wertebereich an. Im Contract sollte kein Bezug auf Implementierungsdetails sondern lediglich auf die grundlegenden Observatoren genommen werden! Nachbedingungen von grundlegenden Observatoren spezifizieren deshalb lediglich Konsistenzbedingungen zwischen den Methoden. Observatoren sind const-Methoden.

2. *Abgeleitete Observatoren* sind i.a. besser lesbar als eine (komplizierte) Kombination grundlegender Observatoren, können evtl. effizienter implementiert sein und sollten dann in Vorbedingungen unbedingt statt der grundlegenden Observatoren benutzt werden. Sie sind const-Methoden. Ihre Nachbedingungen sollten die Return-Werte mit Hilfe der grundlegenden Observatoren vollständig spezifizieren.

3. Konstruktoren (default, Kopier-): ...

4. Zuweisungsoperator: ...

5. `operator==`, `operator!=`: ...

6. Destruktor: ...

7. Modifikatoren: ...

8. `friend`-Methoden und Operatoren: ...

9. Iteratoren: ...