Materialsammlung - Softwarequalität

Prof. Dr. Hans-Jürgen Buhl

Wintersemester 2018/2019

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

Praktische Informatik
PIBUW - WS2018/19
Oktober 2018
13. Auflage

Inhaltsverzeichnis

Abbildungsverzeichnis

Tabellenverzeichnis

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 Programmiertechnik; 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.
Statische Codeanalyse, Asserts, Code-Verifikation, Annotationen (C++-Attribute), Unit-Tests, Reaktion auf Exceptions, ...
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, Diensteerfüllung 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
iOS 11.3: Apple fixt Taschenrechner-Bug – diesmal richtig

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
⋮
Meltdown und Spectre: der Supergau
Endlich Spectre-Schutz für ARM im Linux.Kernel 4.18
Endlich Meltdown-Schutz für 32-Bit-x86-Linux

```
Delta-RPM ./x86_64/openssl-1_1-1.1.0h-lp150.3.3.1_lp150.3.6.1.x86_64.drpm wird heruntergeladen
OK
  Delta-RPM /var/cache/zypp/packages/repo-update/x86_64/openssl-1_1-1.1.0h-lp150.3.3.1_lp150.3.6.1.x86_64.drpm wird angewende
OK
Installieren ./noarch/kernel-firmware-20180525-lp150.2.3.1.noarch.rpm: "Linux kernel firmware files"
  OK
Installieren ./x86_64/libopenssl1_1-1.1.0h-lp150.3.6.1.x86_64.rpm: "Secure Sockets and Transport Layer Security"
  OK
Installieren ./x86_64/libopenssl1_1-32bit-1.1.0h-lp150.3.6.1.x86_64.rpm: "Secure Sockets and Transport Layer Security"
  OK
Installieren ./noarch/ucode-amd-20180525-lp150.2.3.1.noarch.rpm: "Microcode updates for AMD CPUs"
  OK
Installieren ./x86_64/yast2-core-4.0.3-lp150.2.3.1.x86_64.rpm: "YaST2 - Core Libraries"
  OK
Installieren ./x86_64/kernel-default-4.12.14-lp150.12.10.1.x86_64.rpm: "The Standard Kernel"
```

Prevent unauthorized disclosure of information to an attacker with local user access caused by speculative .

CPU-Lücken ret2spec und SpectreRSB entdeckt
NetSpectre liest RAM via Netzwerk aus

Die CPU-Sicherheitslücken Meltdown und Spectre

(Google-)Name	Kurzbezeichnung	CVE-Nummer
Spectre V1	Bounds Check Bypass	CVE-2017-5753
Spectre V1.1	Bounds Check Bypass Store	CVE-2018-3693
Spectre V1.2	Read-only Protection Bypass	k.A.
Spectre V2	Branch Target Injection (BTI)	CVE-2017-5715
Meltdown (GPZ V3)	Rogue Data Cache Load	CVE-2017-5754
Spectre-NG:		
Spectre V3a	Rogue System Register Read (RSRE)	CVE-2018-3640
Spectre V4	Speculative Store Bypass (SSB)	CVE-2018-3639
k.A.	Lazy FP State Restore	CVE-2018-3665
Spectre-Varianten via Return Stack Buffer (RSB)		
"Spectre v5"	ret2spec	k.A.
k.A.	SpectreRSB	k.A.
zu fünf weiteren Spectre-NG-Lücken fehlen noch Informationen		
GPZ steht für Google Project Zero, Spectre V1 und V2 werden auch GPZ V1 und GPZ V2 genannt		

Post-Spectre/Meltdown CPU-Designs
Intel erklärt Hardware-Schutz gegen Spectre- & Meltdown-Lücken

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 Gambro 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)

Y2K38

Programmfehler/Bugs, natürlichsprachige APIs

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 $N =$ `last - first`.

Besser (übersichtlicher) ist die Spezifikationen der C++-Standardbibliothek (früher häufig STL genannt) bei `en.cppreference.com`:

std::sort

Defined in header <algorithm>

```
template< class RandomIt >
void sort( RandomIt first, RandomIt last );                                          (1)  (until C++20)
template< class RandomIt >
constexpr void sort( RandomIt first, RandomIt last );                                     (since C++20)

template< class ExecutionPolicy, class RandomIt >
void sort( ExecutionPolicy&& policy, RandomIt first, RandomIt last );                (2)  (since C++17)

template< class RandomIt, class Compare >
void sort( RandomIt first, RandomIt last, Compare comp );                            (3)  (until C++20)
template< class RandomIt, class Compare >
constexpr void sort( RandomIt first, RandomIt last, Compare comp );                       (since C++20)

template< class ExecutionPolicy, class RandomIt, class Compare >
void sort( ExecutionPolicy&& policy, RandomIt first, RandomIt last, Compare comp );  (4)  (since C++17)
```

Sorts the elements in the range `[first, last)` in ascending order. The order of equal elements is not guaranteed to be preserved.

1) Elements are compared using `operator<`.

3) Elements are compared using the given binary comparison function `comp`.

2,4) Same as (1,3), but executed according to `policy`. These overloads do not participate in overload resolution unless `std::is_execution_policy_v<std::decay_t<ExecutionPolicy>>` is true

Parameters

`first, last` - the range of elements to sort

`policy` - the execution policy to use. See execution policy for details.

`comp` - comparison function object (i.e. an object that satisfies the requirements of *Compare*) which returns `true` if the first argument is *less* than (i.e. is ordered *before*) the second.

The signature of the comparison function should be equivalent to the following:

```
bool cmp(const Type1 &a, const Type2 &b);
```

The signature does not need to have `const &`, but the function object must not modify the objects passed to it.
The types `Type1` and `Type2` must be such that an object of type `RandomIt` can be dereferenced and then implicitly converted to both of them.

Type requirements

- RandomIt must meet the requirements of *ValueSwappable* and *RandomAccessIterator*.
- The type of dereferenced RandomIt must meet the requirements of *MoveAssignable* and *MoveConstructible*.
- Compare must meet the requirements of *Compare*.

Return value

(none)

A. Quelloffenes Eclipse mit CDT/UML2.5/OCL2.4-Tools

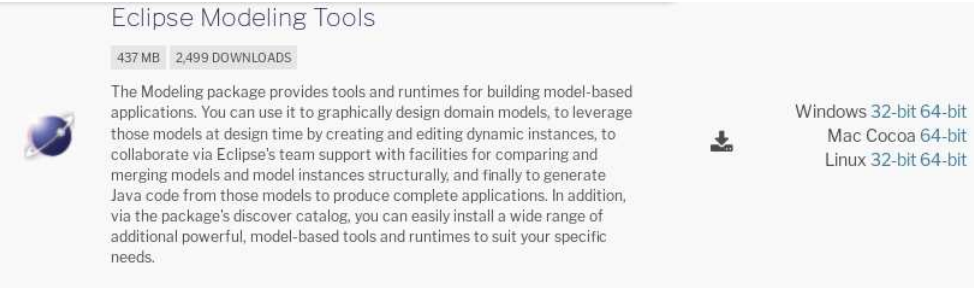
Hilfsmittel (Tools) zur „state of the art“ -Entwicklung von C++-Anwendungen:
Verfügbar (vorinstalliert) ist auf allen Ausbildungsclustern (CIP/IT/PI: l101, ...) der Fachgruppe Mathematik/Informatik der BUW die aktuelle
`eclipse ide 2018-09`

aufrufbar als `eclipse-papyrus1809` mit den Komponenten:

```
Eclipse-Modelling (inkl. Java) mit
   CDT (C/C++ Development Environment inklusive Eclipse standalone Debugger),
   CDT-Linux Tools (für Gcov, Gprof, Perf, Valgrind, ...),
   CUTE (C++ Unit-Tests),
   UML (Papyrus mit Papyrus-Designer (CPP- und Java-Codeerzeugung),
   Eclipse OCL 6.5.0 (Object Constraint Language für Codeverträge/Constraints),
   PyDev (Python),
   D Development Tools (mit gesondert installiertem dmd, dub),
   Scala,
   Kotlin,
   OcaIDE (Ocaml),
   ...,
   cppcheclipse (mit gesondert installiertem cppcheck)
```

Hinweis zur Installation auf dem eigenen Linux-Notebook:

eclipse ide 2018-09 Packages:



Installiere *Eclipse Modeling Tools*, (zur Zeit die Version 2018-09 durch Download der Datei `eclipse-modeling-2018-09-linux-gtk-x86_64.tar.gz` von `http://www.eclipse.org/downloads/packages/`, installiere sie mittels:

```
/Downloads> gunzip eclipse-modeling-2018-09-linux-gtk-x86_64.tar.gz
buhl@rhea3:~/Downloads> ls -al ec*
-rw-r--r-- 1 buhl users 446816013 20. Jun 09:05 eclipse-modeling
    -2018-09-linux-gtk-x86_64.tar
```

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

```
~/sw> tar xf ~/Downloads/eclipse-modeling-2018-09-linux-gtk-x86_64.
    tar
~/sw> ls -al ecli*
drwxr-xr-x 8 buhl users 4096 20. Jun 14:19 eclipse
~/sw> mv eclipse eclipse-modeling-2018-09-linux-gtk-x86_64
~/sw> cd ~/bin
```

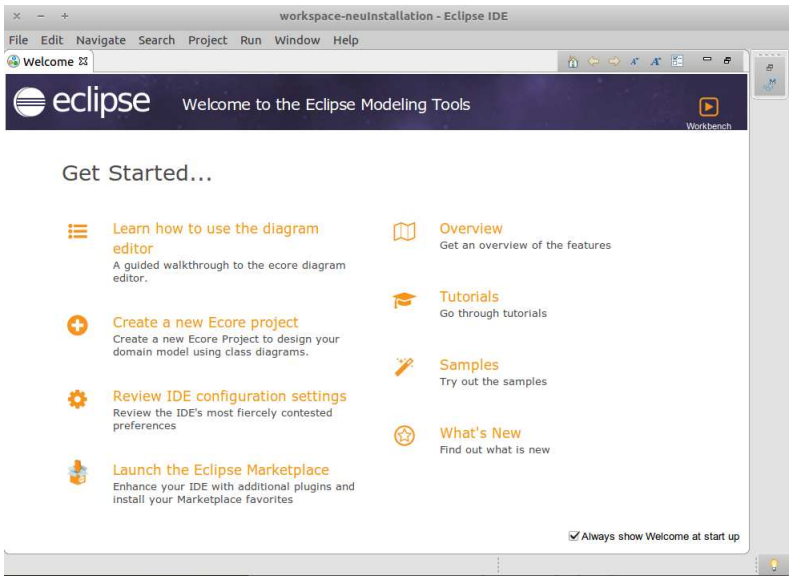
Erzeuge in `$HOME/bin` ein Startskript $HOME/bin/eclipse-papyrus1809 mit dem Inhalt:

```
#!/bin/sh
#
$HOME/sw/eclipse-modeling-2018-09-linux-gtk-x86_64/eclipse $*
```

und gib ihm Ausführbarkeitsrechte:

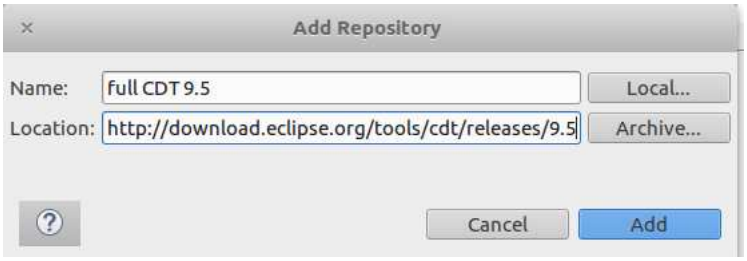
```
~/bin> chmod 755 $HOME/bin/eclipse-papyrus1809
```

Nach Aus- und wieder Einloggen (bzw. Start einer neuen Shell) kann num mittels `exlipse-papyrus1809` die aktuelle Eclipse-IDE gestartet werden:

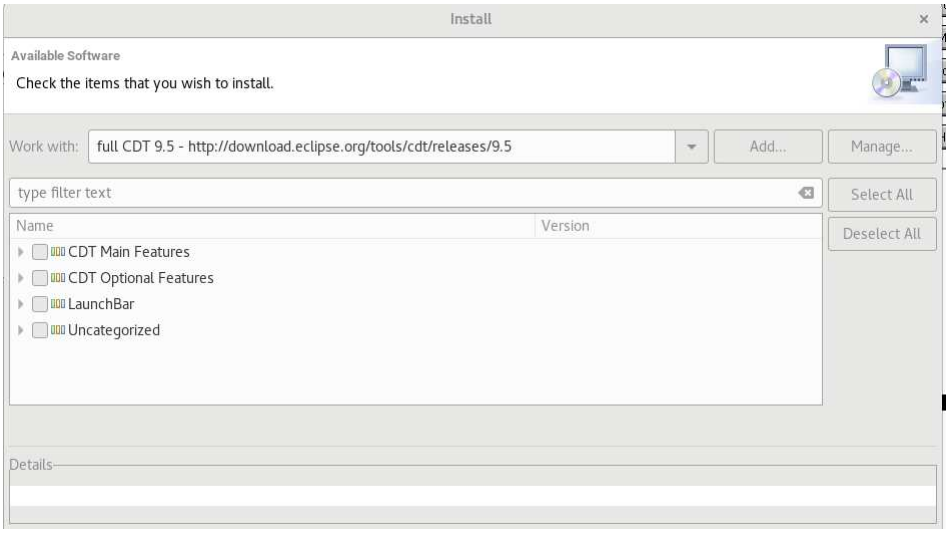


Ergänze dann unter `Help, Install New Software, Add`

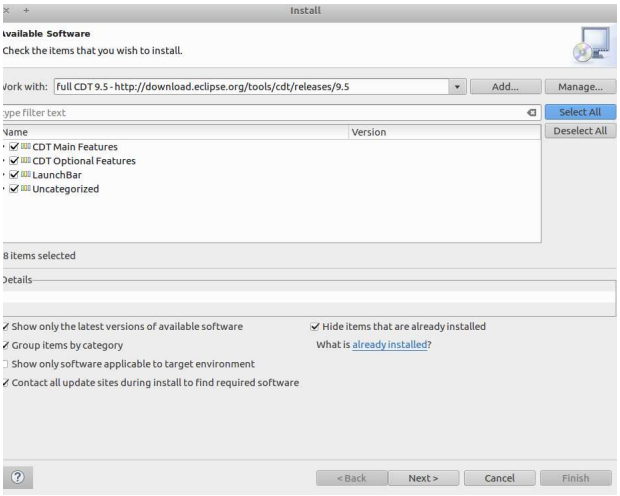
`Name: full CDT 9.5`
`Location: http://download.eclipse.org/toools/cdt/releases/9.5`



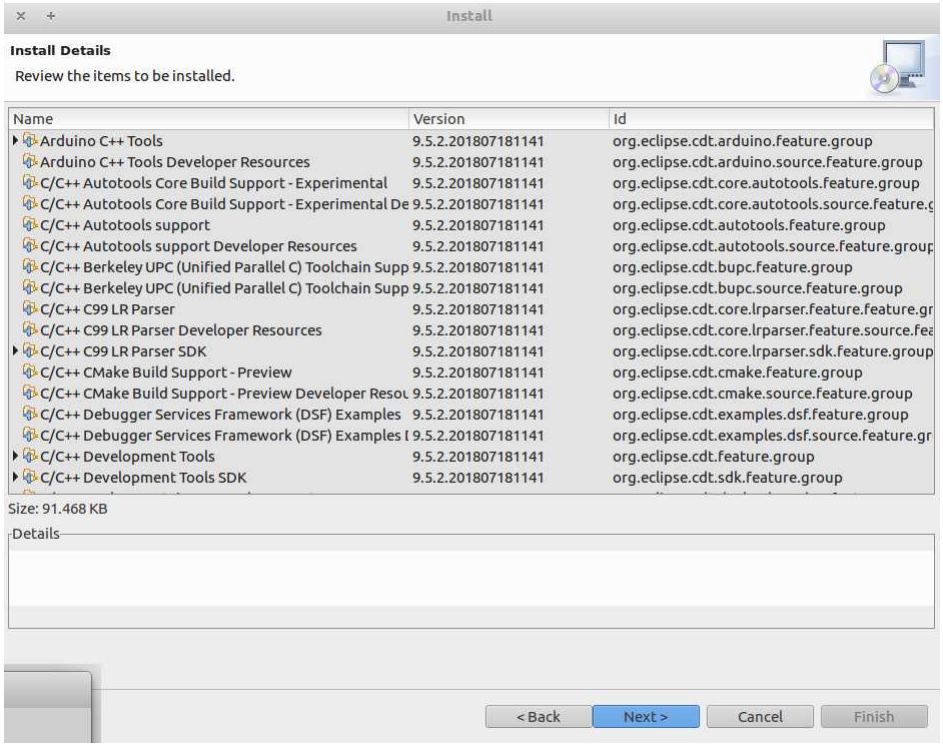
drücke `Add` und im erscheinenden Komponentenüberblick



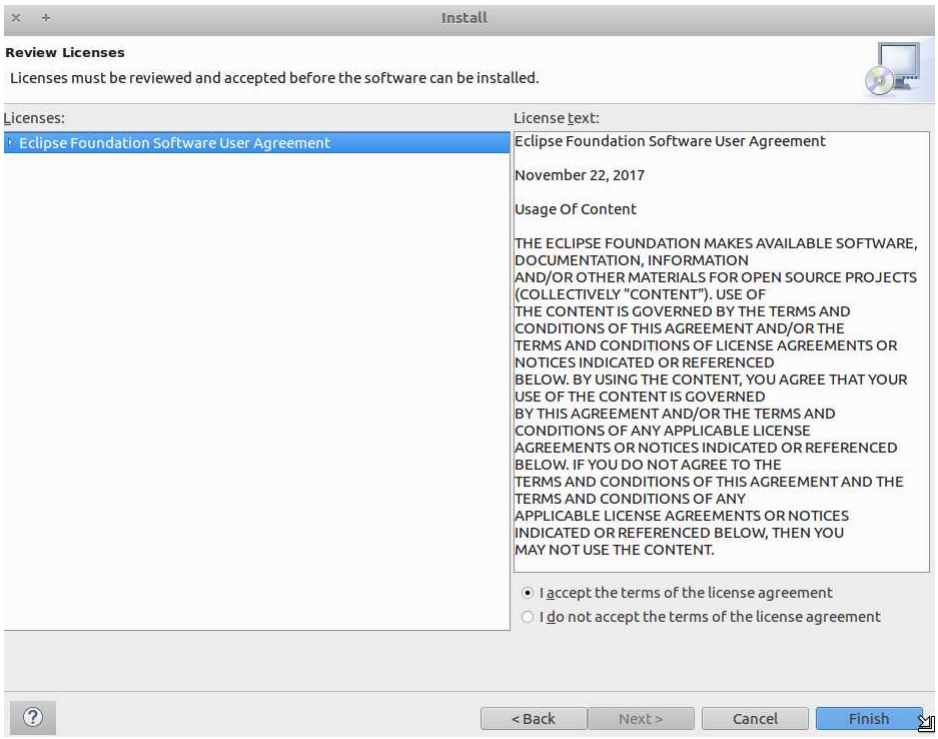
`Select All`:



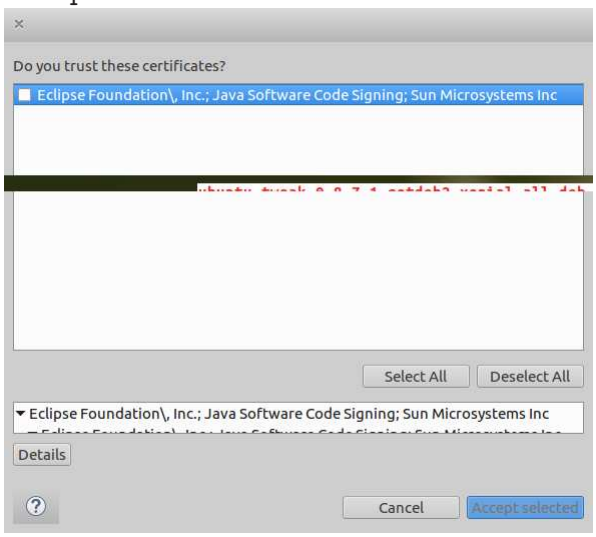
Die Taste `Next>` erzeugt einen Überblick aller zu installierenden Komponenten:



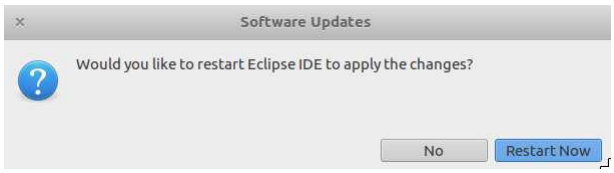
Nach erneuter Betätigung von `Next>`



und Akzeptierung der Lizenzbedingungen (“I accept ...“ gefolgt von der Betätigung der Taste `Finish`) läuft die Installation.
Beantworten Sie dabei ”Do you trust these certificates“ durch `Select All` gefolgt von `Accept Selected`:



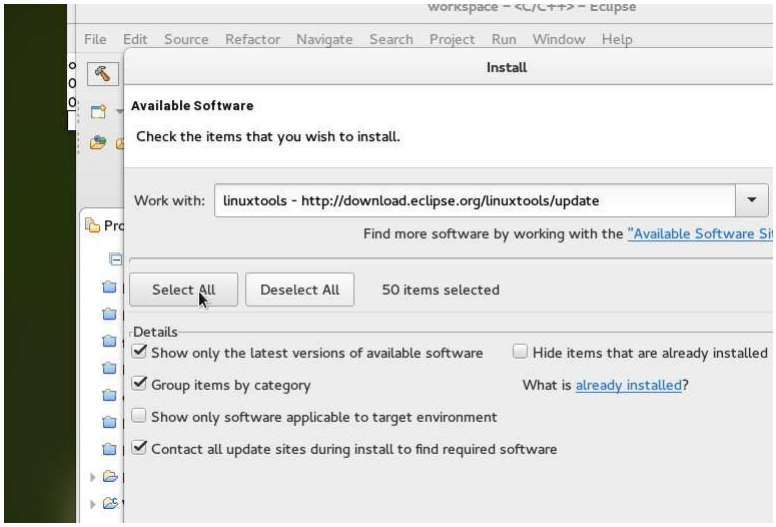
Abschließend (nach Installationsende) sollten Sie in



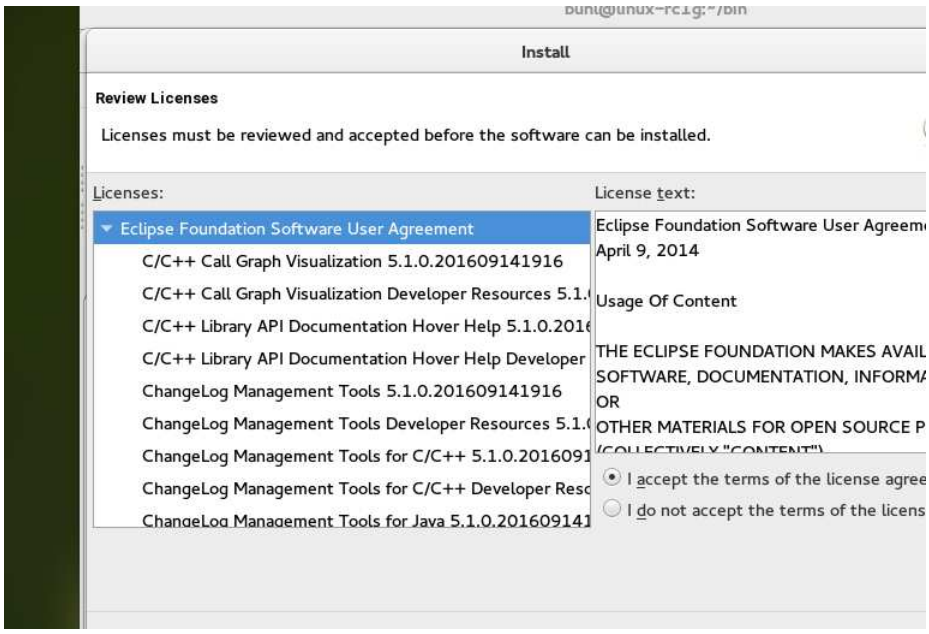
die Taste `Restart Now` betätigen.

Zur analog ablaufenden Installation der Eclipse-Plugins für die Linux-Developertools ergänze unter `Help, Install New Software, Add`

`linuxtools`
`http://download.eclipse.org/linuxtools/update`



das Linuxtools-Repositorium und wähle `Select All` an:

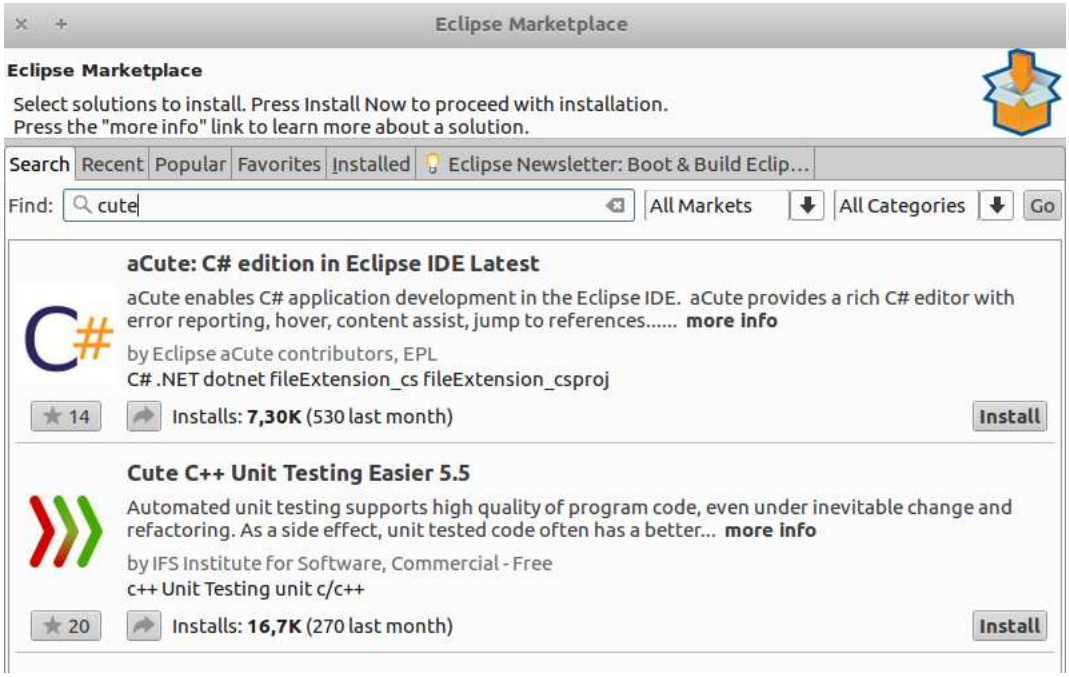


Die Linuxtools bieten Eclipse-Integration qualitätssteigernder Tools für die C++-Entwicklung:

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

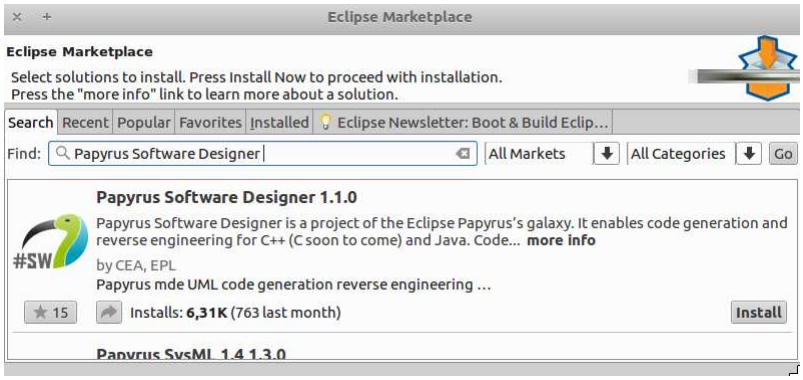
Ähnlich installiert man Cute 5.5, ein C++-Unit-Test-Plugin:

`Help, Eclipse Marketplace`

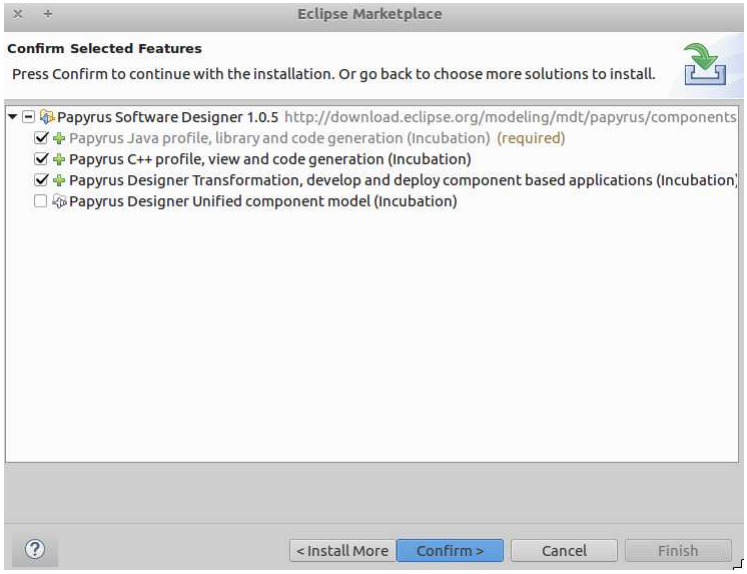


`Install` Cute C++ Unit Testing Easier 5.5, `Confirm`, "I accept ... license...", `Finish`

Ergänze dann unter `Help, Eclipse Marketplace` das UML-Tool `Papyrus Software Designer 1.1.0` (für die Erstellung von UML-Modellen):

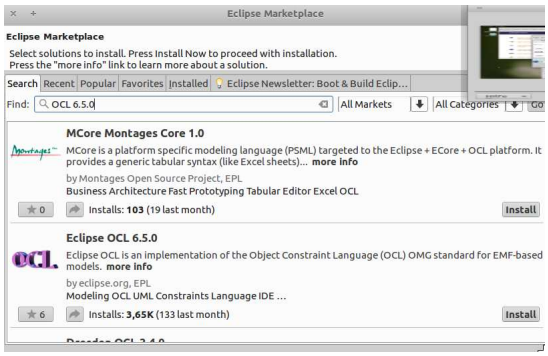


`Install`



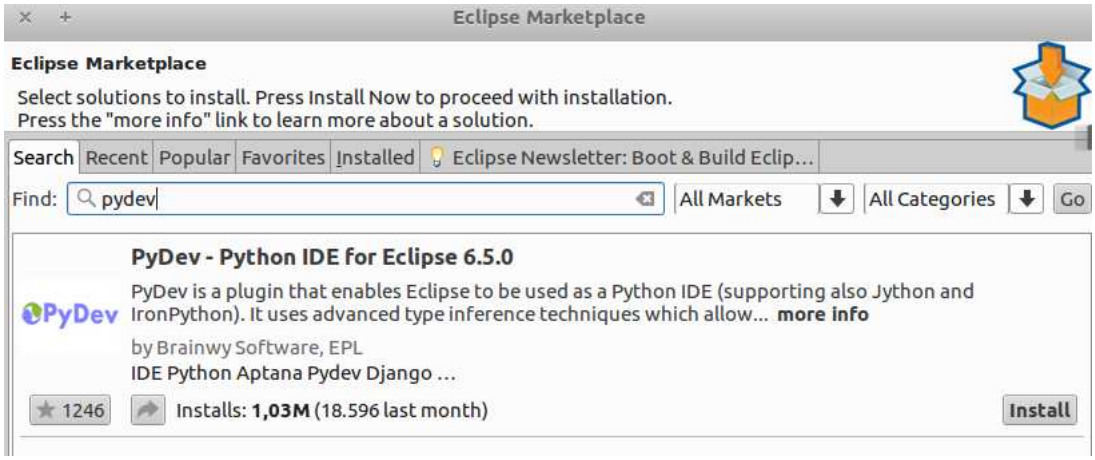
`Confirm`
`Accept Licence, Finish`
`Restart Now`

Ergänze unter `Help, Eclipse Marketplace` dann die `Eclipse OCL 6.5.0` zur Erstellung von formalen Constraints (Codeverträge an die UML-Komponenten):

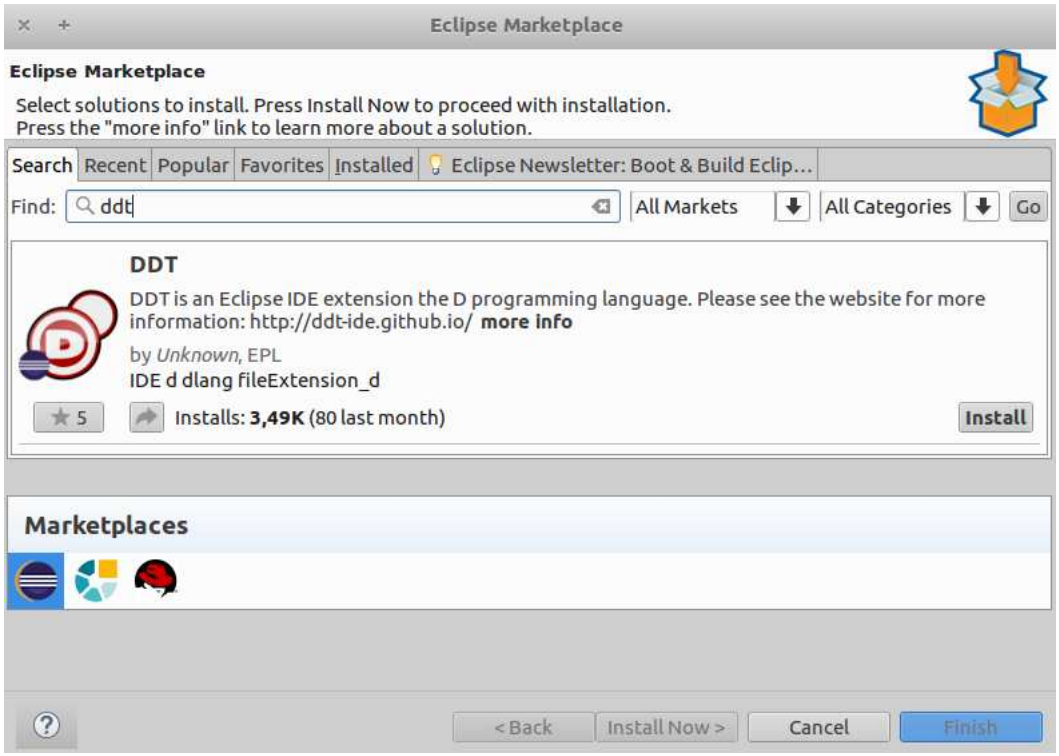


`Install, accept licence, Finish, Restart Now`

Bei Bedarf kann man unter `Help, Eclipse Marketplace` schließlich noch die `Python`-Entwicklungsumgebung `PyDev 6.5.0` installieren



`Install, Confirm, Accept licence, Finish, Restart Now`
und `D-Unterstützung` (ddt 1.0.3, sofern auf Ihrer Maschine `dmd` und `dub` installiert sind)

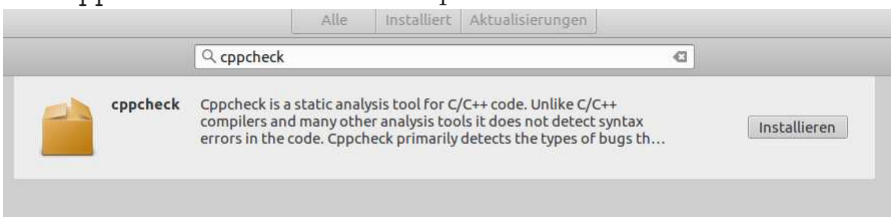


`Install, accept licence, Finish, Select All, Accept selected, Restart Now`

und ... hinzuinstallieren.

(Vorinstalliert auf den Ausbildungsclustern der Fachgruppe als `eclipse-papyrusp` sind zum Beispiel zusätzlich: `Scala IDE 4.7.x, Kotlin 0.8.7, OcaIDE 1.2.21 (für Ocaml)`.)

`cppcheclipse 1.1.0` (`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
GNU Compiler Collection 7.3
GCC 8.2
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

B. Ausblick

PbC/DbC im neuen C++-Standard? (C++20)