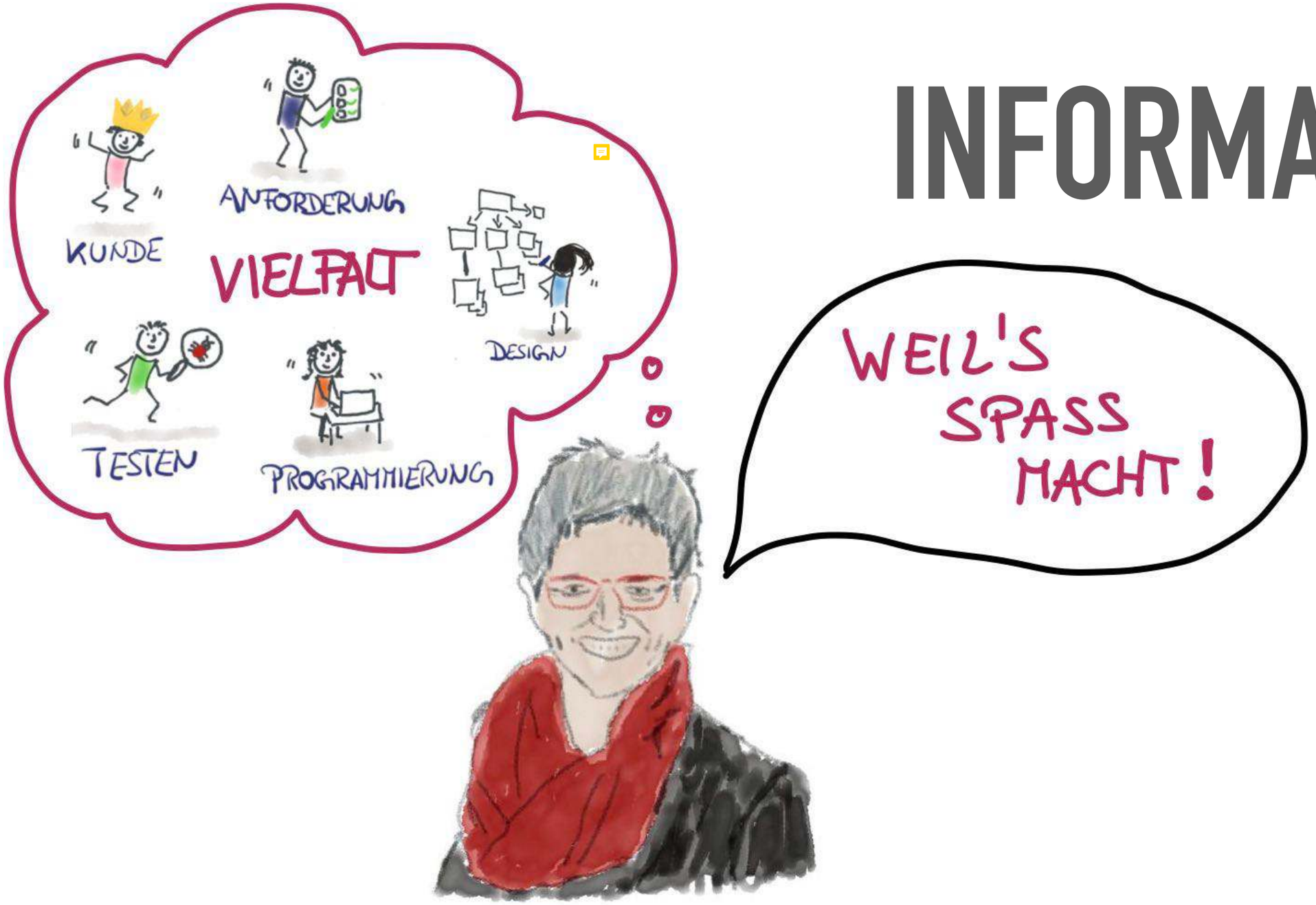
INFORMATIK?


KUNDE
ANFORDERUNG
VIELFALT
DESIGN
TESTEN
PROGRAMMIERUNG
WEIL'S SPASS MACHT!

WER, WO, WIE , WAS?

- Wer bin ich?
- Wo komme ich her?
- Wie bin ich dort hingekommen?
- Was mache ich?

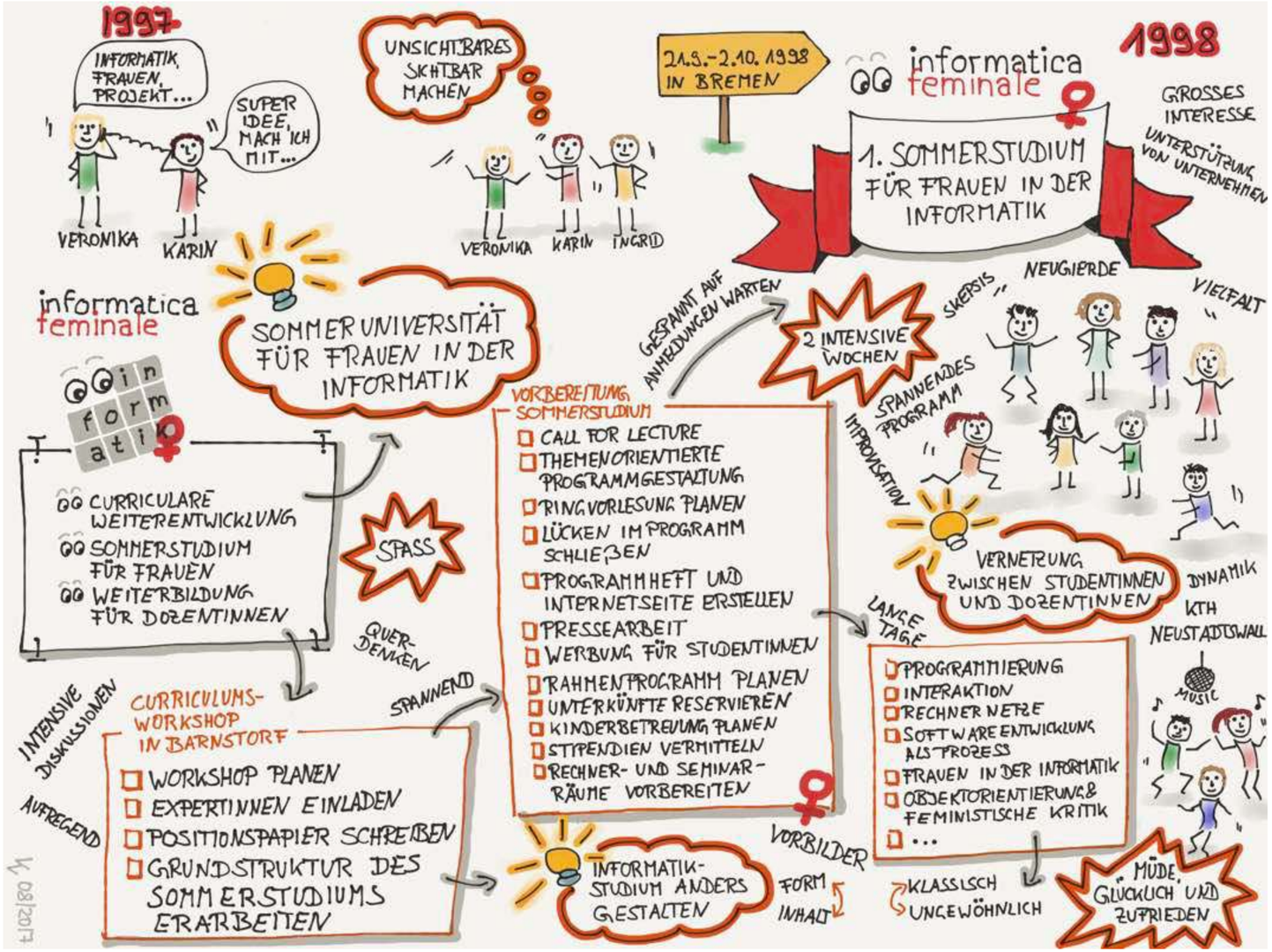
STUDIUM UND PROMOTION

- 1981-1987: Informatikstudium an der Universität Bremen
  - Schwerpunkt Betriebssysteme und Rechnernetze
  - Projektstudium: Erste Berührungspunkte zur Forschung
- 1987-1990: wiss. Mitarbeiterin in der AG Softwaretechnik & Rechnernetze
- 1990-1994: wiss. Mitarbeiterin in der AG Softwaretechnik, Betriebssysteme & Rechnerarchitektur < 1992: 1. KIND
  - Erste internationale Veröffentlichungen
- 1996: Abschluss der Promotion < 1996: 2. KIND
  - „Sichere Ausführungsumgebungen für Objekte“

INFORMATICA FEMINALE

- 1997-2000: Sommeruniversität für Frauen in der Informatik
  - Curricula-Entwicklung
  - Sommerstudium
  - Weiterqualifizierung von Lehrenden

INFORMATICA FEMINALE



COMMERZ SYSTEMS

- 2001-2009: Business Development
  - Unternehmensprozesse, Qualitätssicherung
  - Einführung eines Projektmanagement-Werkzeugs
- Erfahrungen sammeln außerhalb der Universität
  - Voraussetzung für eine Professur an der Fachhochschule

HOCHSCHULE BREMERHAVEN

- seit 2009: Professorin in den Studiengängen Informatik, Wirtschaftsinformatik und Digitalisierung, Innovation und Informationsmanagement
  - IT-Systemintegration
  - Qualität von Software
  - Curricula-Diskussionen
  - Frauen in der Informatik

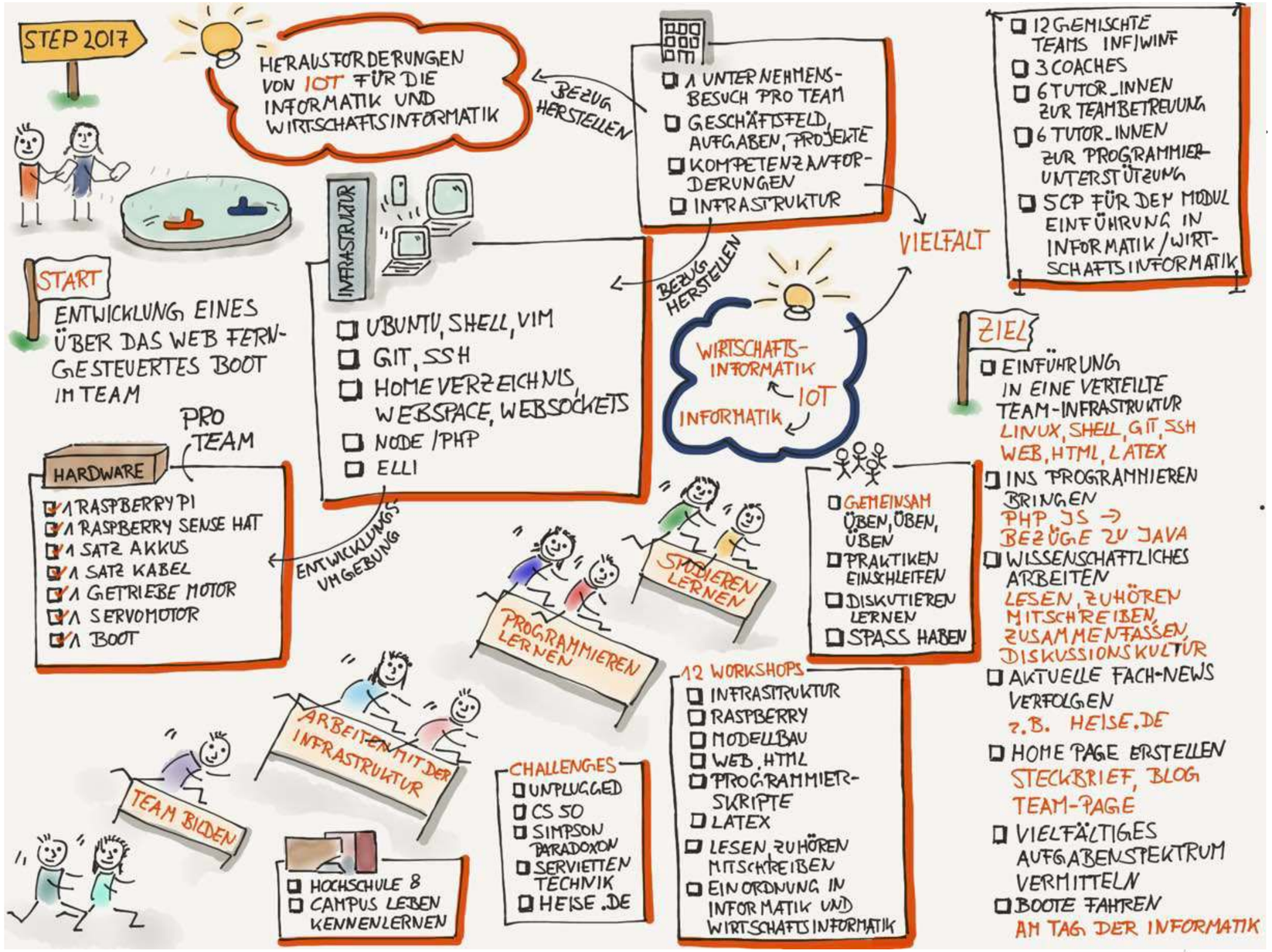
Hochschule Bremerhaven

LEHRE

- Vielfältige Themen:
  Modellierung, Qualitätssicherung
  insbesondere Testmethoden und Testautomatisierung,
  Softwareentwicklungsprozesse,
  Softwarearchitekturen - Systemintegration
- Experimentieren mit Lernformen
- Gender in der Lehre

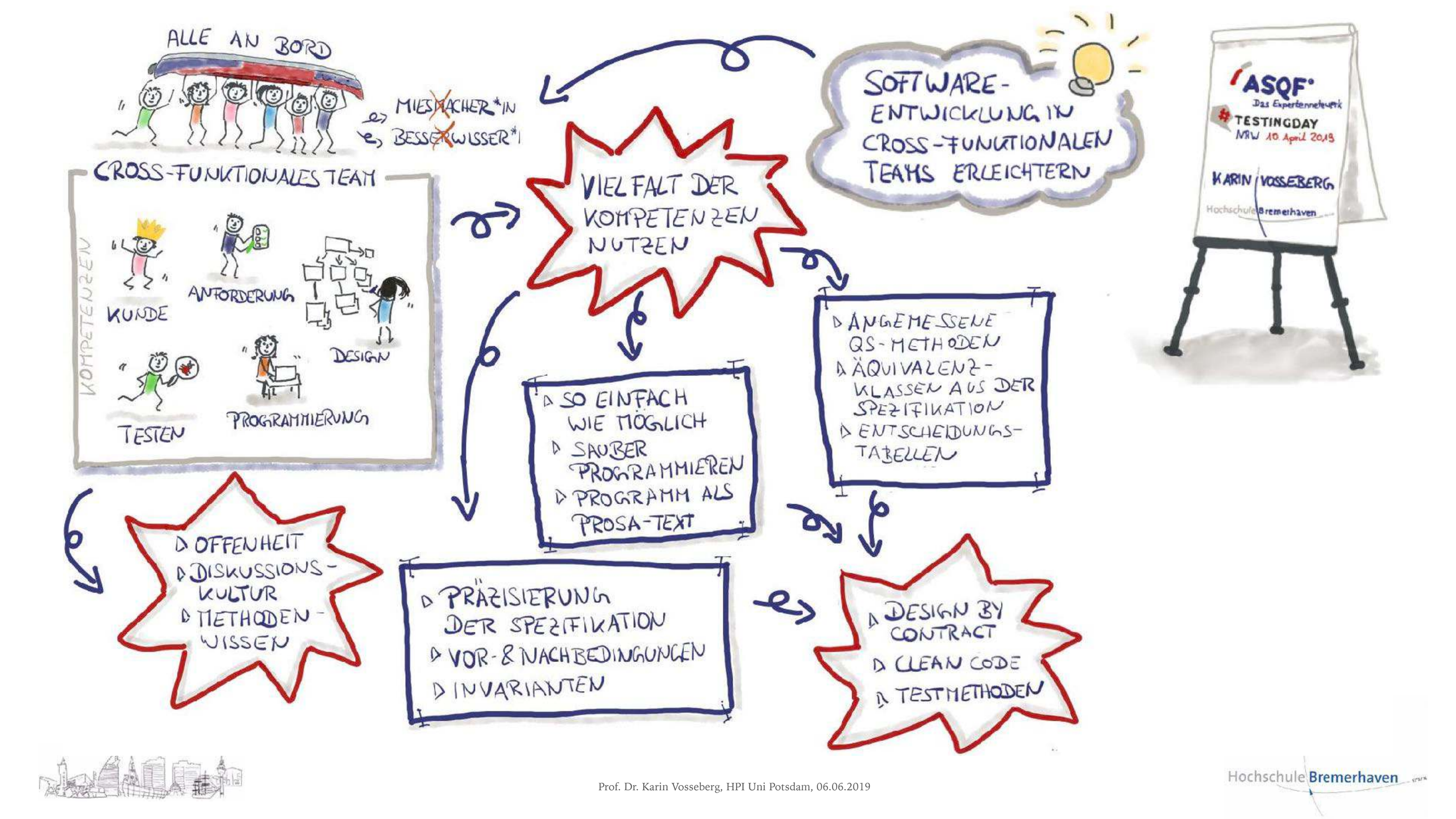
STEP

- Mit Projekten ins Studium starten
- Workshops in großen Gruppen
- Studierende als Coaches begleiten



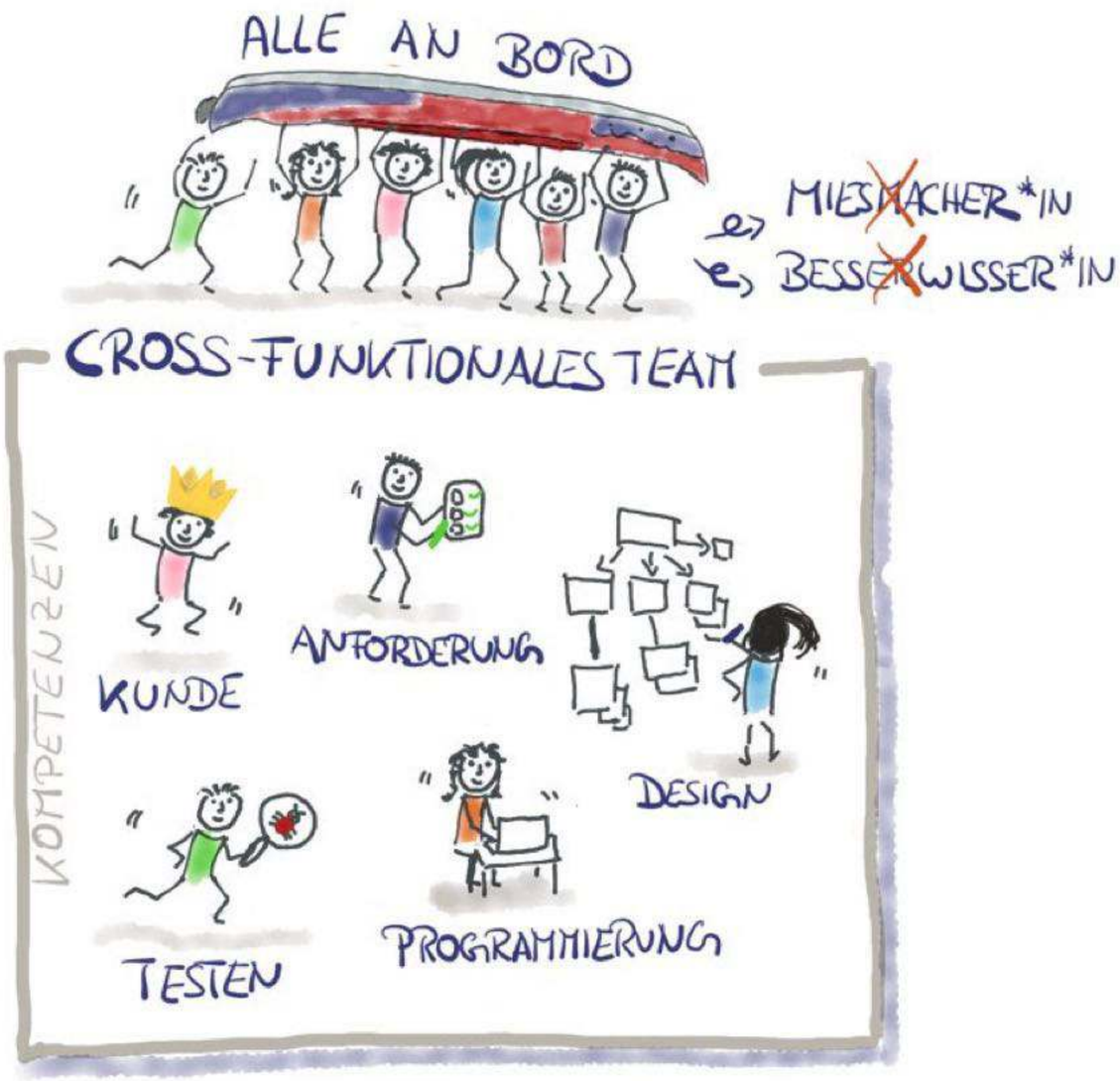
FORSCHUNG

- Projekt-Studium: Kommunikationsprotokolle - RPC
- Promotion: Sicherheitsmechanismen in objektorientierten Betriebssystemen
- Informatica Feminale: Genderforschung in der Informatik
- Commerz Systems: Maintenance-Prozesse und Qualitätssicherung
- Hochschule Bremerhaven: Qualitätssicherung, Softwareentwicklungsprozesse und Genderforschung


ALLE AN BORD
MIESMACHER*IN
BESSERWISSER*I
SOFTWARE-ENTWICKLUNG IN CROSS-FUNKTIONALEN TEAMS ERLEICHTERN
ASQF
Das Expertennetzwerk
TESTINGDAY
MRW 10. April 2019
KARIN VOSSEBERG
Hochschule Bremerhaven
CROSS-FUNKTIONALES TEAM
KOMPETENZEN
KUNDE
ANFORDERUNG
DESIGN
TESTEN
PROGRAMMIERUNG
VIELFALT DER KOMPETENZEN NUTZEN
SO EINFACH WIE MÖGLICH
SAUBER PROGRAMMIEREN
PROGRAMM ALS PROSA-TEXT
ANGEMESSENE QS-METHODEN
ÄQUIVALENZ-KLASSEN AUS DER SPEZIFIKATION
ENTSCHEIDUNGS-TABELLEN
OFFENHEIT
DISKUSSIONS-KULTUR
METHODEN-WISSEN
PRÄZISIERUNG DER SPEZIFIKATION
VOR- & NACHBEDINGUNGEN
INVARIANTEN
DESIGN BY CONTRACT
CLEAN CODE
TESTMETHODEN

CROSS-FUNKTIONALES TEAM

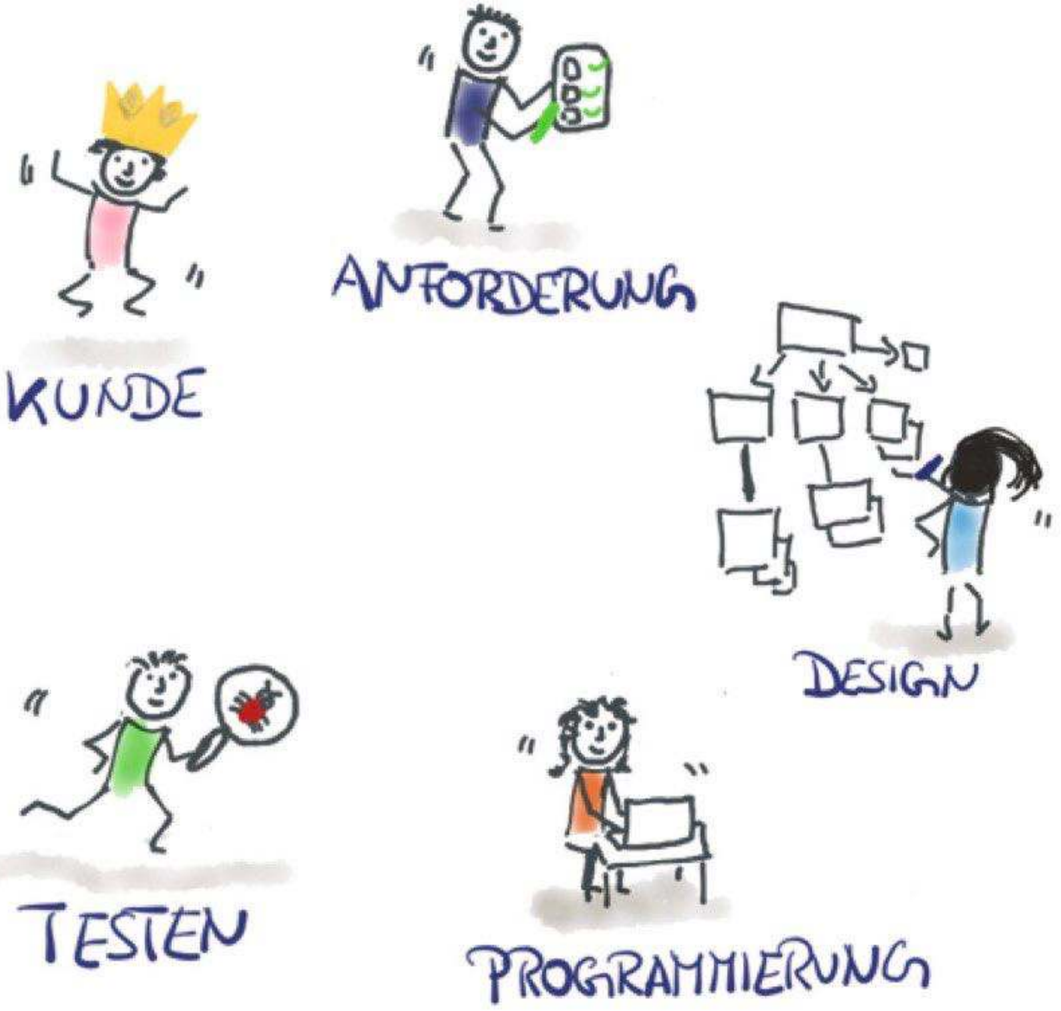
- Kontinuierlicher Austausch im Cross-funktionalen Team verhindert viele Missverständnisse und Fehler.
- Er fördert die fachliche Diskussion bzw. ermöglichen diese erst.
- Wird die Kommunikation darüber hinaus noch methodisch unterstützt, ergeben sich weitere Vorteile.
- Beispiele für einzunehmende Sichten im Cross-funktionalen Team:



AUFGABEN DER SOFTWAREENTWICKLUNG

- Was ist zu entwickeln?
- Welche Struktur ist angemessen?
- Wie erfolgt die Umsetzung?
- Wie sieht die Prüfung aus?

unabhängig vom gewählten Vorgehen/Modell



BEISPIEL – WAS IST ZU ENTWICKELN?

- Die Kosten für eine Taxifahrt sind zu ermitteln.
- Der Fahrpreis setzt sich aus einem Grundpreis von 3,50€ und einem Preis pro gefahrenen Kilometer von 2,10€ zusammen.
- Nachtfahrten mit einem Zuschlag von 20% und Gepäckstücke mit 3,00€ werden ebenso berücksichtig und verteuern den Fahrpreis.

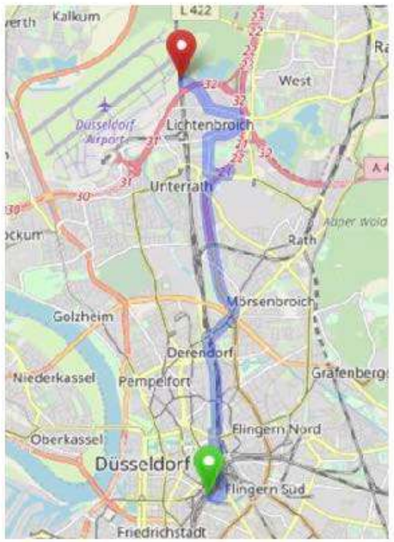
Gepäck?

Tag-Fahrt?

BEISPIEL – WAS IST ZU ENTWICKELN?

- Rabattberechnung in Abhängigkeit von den gefahrenen Kilometern
- Bei einer Taxifahrt von mehr als 10 Kilometer gibt es einen Rabatt von 5%, ab 50 Kilometer steigt der Rabatt auf 10%.

bis Düsseldorf Hbf
4,6 km



bis Düsseldorf Flughafen
11 km

https://www.openstreetmap.org

bis HS Bremerhaven
341 km

aus


IST DOCH KLAR WAS ICH WILL
LASS ES UNS LIEBER GENAUER FORMULIEREN
EIN PAAR TESTS WÜRDEN MIR AUCH EINFALLEN
JA, UNBEDINGT. ICH BIN DABEI
WIESO DENN? ICH KÖNNTE DOCH SCHON ANFANGEN.

FORMALERE BESCHREIBUNG DES SACHVERHALTS

```
streckeInKm <= 10            kein Rabatt

10 < streckeInKm <= 50       5% Rabatt

streckeInKm > 50             10% Rabatt
```

BESSERE BESCHREIBUNG DES SACHVERHALTS

```
streckeInKm <= 10            kein Rabatt

10 < streckeInKm <= 50       5% Rabatt

50 < streckeInKm             10% Rabatt
```

NOCH BESSERE BESCHREIBUNG DES SACHVERHALTS

```
0 < streckeInKm <= 10                kein Rabatt
10 < streckeInKm <= 50               5% Rabatt
50 < streckeInKm <= MaxStrecke       10% Rabatt
```

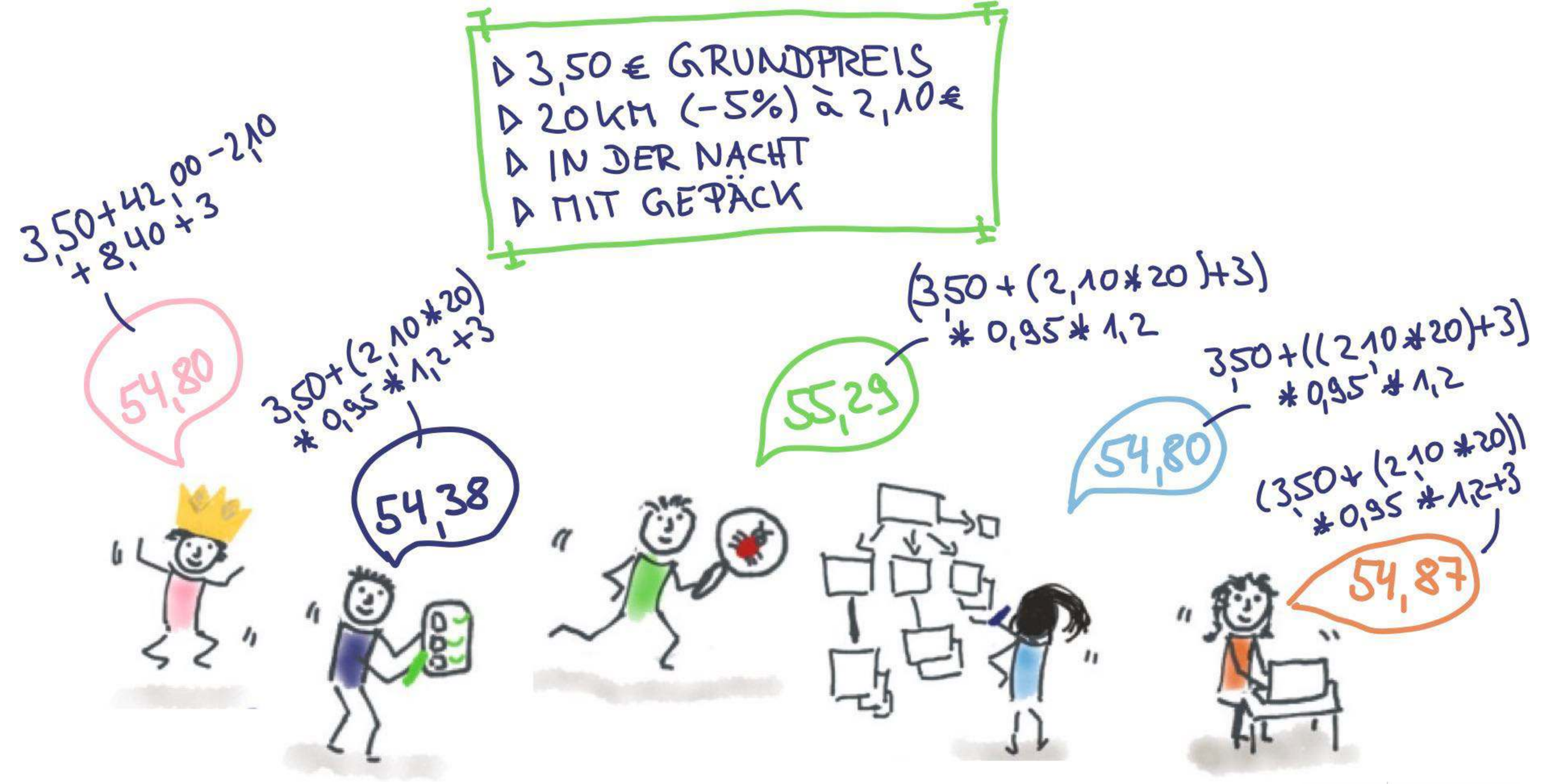
ERGEBNIS DER DISKUSSION DER ANFORDERUNG (BEI INTEGER ALS DATENTYP)

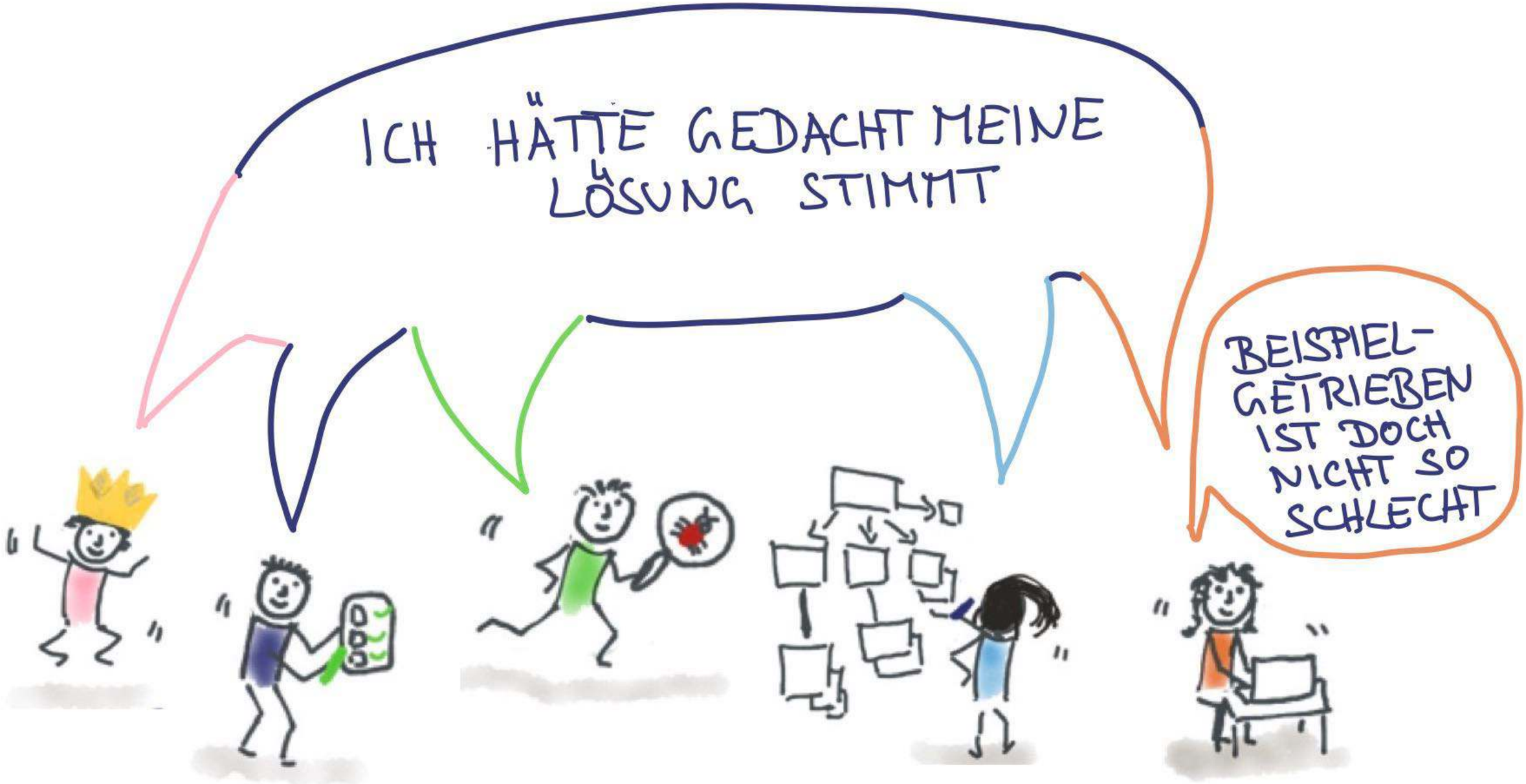
`MinInt <= streckeInKm <=0`	`Fehlermeldung`
`0 < streckeInKm <= 10`	`kein Rabatt`
`10 < streckeInKm <= 50`	`5% Rabatt`
`50 < streckeInKm <= MaxStrecke`	`10% Rabatt`
`MaxStrecke < streckeInKm <= MaxInt`	`Fehlermeldung`

Der gesamte Datenbereich für Integer von `MinInt` *(kleinster Integerwert auf dem Rechner) bis* `MaxInt` *(größter Integerwert) ist mit den fünf* ***lücken- und überschneidungsfreien*** *Unterbereichen abgedeckt.*


PERFEKT
DER TEIL IST NUN KLAR
"ÄQUIVALENZ-KLASSEN FREI HAUS
SEHR PRÄZISE FORMULIERT
ICH KANN NUN ENDLICH ANFANGEN, ODER?


OK
NA GUT, WENN IHR MEINT
LASST UNS MAL EINE BEISPIEL-RECHNUNG MACHEN
VON MIR AUS
IST DOCH NICHT NÖTIG!


Δ 3,50 € GRUNDPREIS
Δ 20 KM (-5%) à 2,10 €
Δ IN DER NACHT
Δ MIT GEPÄCK
3,50+42,00-2,10 +8,40+3
54,80
3,50+(2,10*20) *0,95*1,2+3
54,38
(3,50+(2,10*20)+3) *0,95*1,2
55,29
3,50+((2,10*20)+3) *0,95*1,2
54,80
(3,50+(2,10*20)) *0,95*1,2+3
54,87


ICH HÄTTE GEDACHT MEINE LÖSUNG STIMMT
BEISPIEL-GETRIEBEN IST DOCH NICHT SO SCHLECHT

WELCHE STRUKTUR IST ANGEMESSEN?

- Festlegung der Schnittstellen zwischen den einzelnen Systembausteinen
- Einzelne Bausteine sollen vom restlichen System entkoppelt werden, wodurch sie separat entwickelt und getestet werden können.
- »Design by Contract«
  ist ein Konzept zur Festlegung von Schnittstellen

DESIGN BY CONTRACT

- Definition formaler Verträge zur Verwendung von Schnittstellen
- Vertrag besteht aus
  - Vorbedingungen (precondition),
    also den Zusicherungen, die eine aufrufende Einheit einzuhalten hat
  - Nachbedingungen (postcondition),
    also den Zusicherungen, die die aufgerufene Einheit einhalten wird, sowie den
  - Invarianten (class invariants),
    quasi dem Gesundheitszustand einer Klasse, der nicht verletzt werden darf.
- Sofern sich Aufrufende an Vorbedingungen und Invarianten halten, können keine Fehler auftreten und die Methode liefert garantiert keine unerwarteten Ergebnisse, die Einhaltung der Nachbedingungen und die Unversehrtheit der Invarianten sind gegeben.

BEISPIEL ZUR BERECHNUNG DES PREISES EINER TAXIFAHRT

- Die Schnittstelle der Methode sieht wie folgt aus:

```
int fahrpreis( int grundpreisInCent,
               int preisInCentProKm,
               int streckeInKm,
               bool nachtfahrt,
               bool gepaeckVorhanden)
```

VORBEDINGUNGEN

- für die beiden ersten Parameter:

```
grundpreisInCent > 0    AND
                  grundpreisInCent <= MaxGrundpreis

preisInCentProKm > 0    AND
                  preisInCentProKm <= MaxKilometerPreis
```

Die Werte für `MaxGrundpreis` *und* `MaxKilometerPreis`
müssen aus den Anforderungen hervorgehen

VORTEIL VON DESIGN BY CONTRACT

- Der Aufrufende der Methode `fahrpreis()` ist für die Einhaltung der Vorbedingungen verantwortlich.
- Der Programmierer kann davon ausgehen, dass für die Parameter nur gültige Werte übergeben werden
- Der Programmierer der Methode garantiert, dass der Fahrpreis korrekt berechnet wird.
- Die Aufgabenverteilung ist klar geregelt.

DESIGN BY CONTRACT – AGILE QUALITY MATURITY MODEL DER DB VERTRIEB

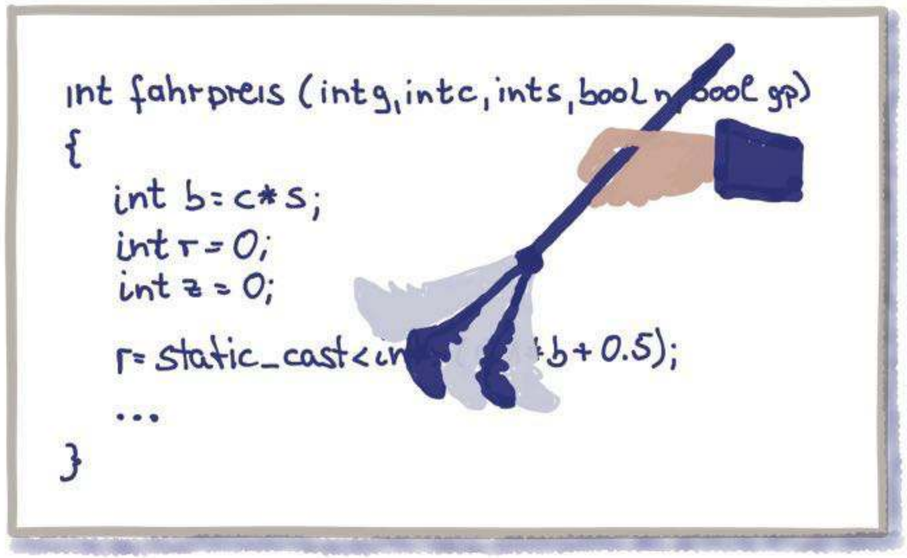
	Stufe 1	Stufe 2	Stufe 3	Stufe 4	Stufe 5
Funktionalität	DoR enthält Testkriterien	DoD enthält Test neuer Funktionalität	DoD enthält Regressions-tests	BDD Behaviour Driven Development	Continuous Testing
Schnittstellen	Design by Contract	Test von - Mappings - Steuerung - Verarbeitung	Mappings werden mit Unit-Tests automatisiert	Schnittstellen werden gemockt	Continuous Testing
Prozesse	Klare Teamverant-wortlichkeit	Regressions-tests werden durch zentrale QA ausgeführt	Prozess-Tests werden automatisiert	Regressions-tests werden vom Team ausgeführt	Continuous Testing
Policy	Transparenz im Test	Abbau von Defects	Abbau von technischen Schulden	„Keine Fehler“ Zielsetzung	Testfälle = Single Point of Truth

Andreas Kühl, telexiom AG:
How to ... Get a Common Understanding for Quality in a Scaled Agile World. Das Agile Quality Maturity Modell der DB Vertrieb, QS_Tag, 2017, 20.10.2017


NA WENN DIE AUFTEILUNG ZU WENIGER FEHLERN FÜHRT...
JEDE METHODE BEKOMMT EINEN VERTRAG
SPART KEINE TESTS, ABER ES WIRD KLARER WO ETWAS ZU TESTEN IST
WIEDERVERWENDUNG WIRD ERLEICHTERT
OK, DANN LEG ICH NUM MAL LOS

WIE ERFOLGT DIE UMSETZUNG?

- Zusammenarbeit ist auch bei der Programmierung sehr sinnvoll - und nicht nur unter Programmier*innen.
- Voraussetzung ist allerdings, dass der Programmcode möglichst einfach und klar verständlich ist
- Clean-Code-Bewegung hat eine »saubere« Programmierung zum Ziel



WIE ERFOLGT DIE UMSETZUNG?

- Unsauberer Code


```
// gibt Preis in Eurocent zurück
int Fahrpreis( int g,          //Grundpreis
               int c,          //Preis pro km
               int s,          //Strecke
               bool n,         //Nachtfahrt
               bool gp) {      //Gepäck ja/nein
  int b = c * s;               //Basispreis
  int r = 0;                   //Rabatt
  int z = 0;                   //Zuschlag für Nachtfahrt
  if(s > 50)
      r = static_cast<int>(0.1 * b + 0.5);
  else if(s > 10)
     r= static_cast<int>(0.05 * b + 0.5);
  if(n)
     z = static_cast<int>(0.2 * b + 0.5);
  if(gp)
     z += 300;                 //Zuschlag für Gepäck
  return g + b + z - r;
}
```

WIE ERFOLGT DIE UMSETZUNG?

- Sauberer Code


```
// gibt Preis in Eurocent zurück
int Fahrpreis( int grundpreisInCent,
               int preisInCentProKm,
               int streckeInKm,
               bool nachtfahrt,
               bool gepaeckVorhanden) {
  int streckenPreisInCent = preisInCentProKm * streckeInKm;
  int langstrecke = 50;
  int mittelstrecke = 10;
  int rabatt = 0;
  float rabattLangstrecke = 0.1;
  float rabattMittelstrecke = 0.05;
  if(streckeInKm > langstrecke)
      rabatt = std::round(rabattLangstrecke * streckenPreisInCent);
  else if(streckeInKm > mittelstrecke)
     rabatt = std::round(rabattMittelstrecke * streckenPreisInCent);

  int zuschlag = 0;
  float zuschlagNachtProzent = 0.2;
  if(nachtfahrt)
     zuschlag = std::round(zuschlagNachtProzent * streckenPreisInCent);

  int zuschlagGepaeckInCent = 300;
  if(gepaeckVorhanden)
     zuschlag += zuschlagGepaeckInCent;

  return grundpreisInCent + streckenPreisInCent + zuschlag - rabatt;
}
```


ICH KANN DEN CODE VERSTEHEN
ICH AUCH- HÄTTE ICH NICHT GEDACHT
SPART KOMMENTARE, DIE SO ODER SO NIE MIT GEPFLEGT WERDEN
LIEST SICH DOCH WIE PROSA-TEXT
VERSTEHE MEINEN CODE AUCH NOCH NACH WOCHEN

WIE SIEHT DIE PRÜFUNG AUS?

- Durch die präzise Formulierung der Anforderungen sind die Äquivalenzklassen quasi frei Haus geliefert worden.
- Als Ergänzung wurde die Grenzwertanalyse eingesetzt.
- Mögliche Kombinationen der Parameter sind bisher nicht berücksichtigt.
- Prüfung der möglichen Kombinationen einer Taxifahrt
  - ob lang oder kurz, in der Nacht und mit oder ohne Gepäck.

ENTSCHEIDUNGSTABELLENTEST

Bedingungen	1	2	3	4	5	6	7	8	9	10	11	12
> 10 km	–	–	–	–	j	j	j	j	n	n	n	n
> 50 km	j	j	j	j	n	n	n	n	n	n	n	n
Nachtfahrt	j	j	n	n	j	j	n	n	j	j	n	n
Gepäck	j	n	j	n	j	n	j	n	j	n	j	n
Aktionen												
Rabatt 5%					x	x	x	x				
Rabatt 10%	x	x	x	x								
Zuschlag 20%	x	x			x	x			x	x		
Zuschlag 3 €	x		x		x		x		x		x	
Normalpreis												x

Konsolidierte Entscheidungstabelle für den Taxifahrpreis (j ja / n nein / - don`t care)

ENTSCHEIDUNGSTABELLENTEST

- Mit den 12 Kombinationen werden alle sinnvollen Möglichkeiten abgedeckt.
- Jede der 12 Spalten entspricht einem Testfall. Im oberen Teil sind die jeweiligen Bedingungen abzulesen und im unteren Teil der Entscheidungstabelle weisen die Kreuze darauf hin, welche Aktionen bei der Konstellation zu erwarten ist.
- Durch das Vorgehen werden doppelte Testfälle verhindert ebenso ein Übersehen von möglichen Kombinationen.


12 TEST-FÄLLE SIND JA RICHTIG WENIG
GUT, DASS ALLE SINNVOLLEN KOMBINATIONEN DABEI SIND
ANGEMESSEN STATT AUFWENDIG – DARAUF KOMMT ES EBEN AN!
ALLE METHODEN SIND SO SYSTEMATISCH ZU TESTEN
TESTEN WÄRE DOCH GAR NICHT NÖTIG GEWESEN

VORTEILE

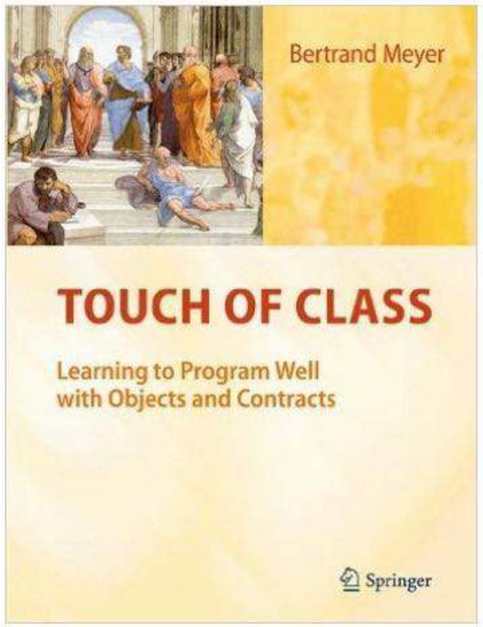
- Präzise Spezifikation
  - erleichtert Design, Programmierung und Test
- Design by Contract
  - Einfache Programmierung - Voraussetzung für Clean Code
  - Test wird einfacher - Zuständigkeiten und Testaufwand geklärt
- Clean Code
  - einfache Änderbarkeit und Testbarkeit

RESÜMEE

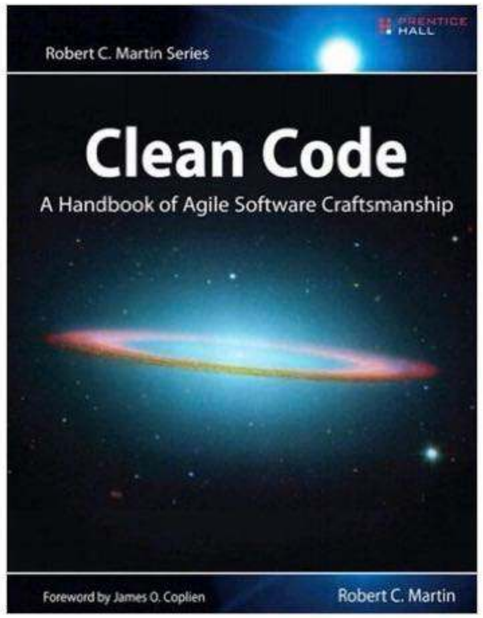
- Standup Meeting

BÜCHER

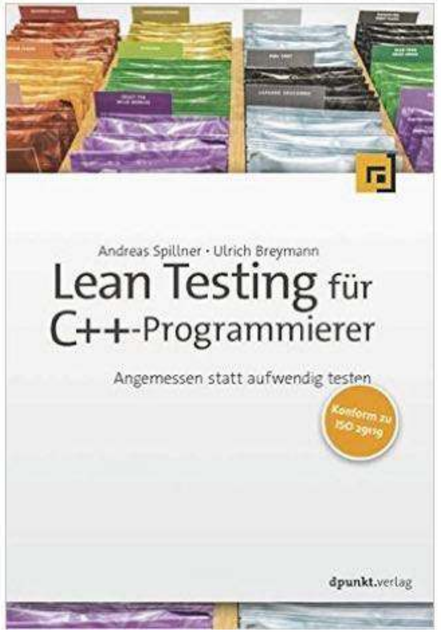
Bertrand Meyer
Touch of Class
Springer Verlag, Berlin, 2009



Robert C. Martin
Clean Code
Prentice Hall, 2008



Andreas Spillner, Ulrich Breymann
Lean Testing für C++-Programmierer
dpunkt.verlag, Heidelberg, 2016



VERWALTUNG

- Studiengangsleiterin der Informatik und Wirtschaftsinformatik
- Zentrale Frauenbeauftragte
- Vorsitzende von Berufungskommissionen
- Konrektorin für Studium und Lehre
  - IUP-Amt, Servicestelle Lernen und Lehren, Qualitätspakt Lehre, Rechenzentrum
  - Strategiediskussionen: Hochschulausbau

PROFESSORIN?