Prinzipiensprachen

Entwurfsentscheidungen treffen und darüber reden

Christian Rehn aka R2C2

Delphi-Treff

Delphi-Tage 2013

Vorstellung

- Christian Rehn aka R2C2
- 2001 angefangen zu programmieren
- Informatikstudium an der TU Kaiserslautern
- Moderator und Redakteur bei Delphi-Treff
- Seit Mai im 1&1 Source Center
- `http://www.christian-rehn.de/`

Organisatorisches

- Folien enthalten nur wenig Text
  - Besser für Präsentation
  - Zusätzlich ausführlichere Vortragsnotizen online: `http://www.christian-rehn.de/`
- Feedback erwünscht (persönlich, per Mail, im Forum, im Blog, ...)

Überblick

Geschichte

Es war einmal. . .

QBASIC

Delphi


CSS
Delphi
ML
C#
Ruby
Haskell
Java
C++
C
TurboPascal
PHP
Groovy
HTML
JavaScript


DANGER
MAN AT WORK

Aber der Code. . .

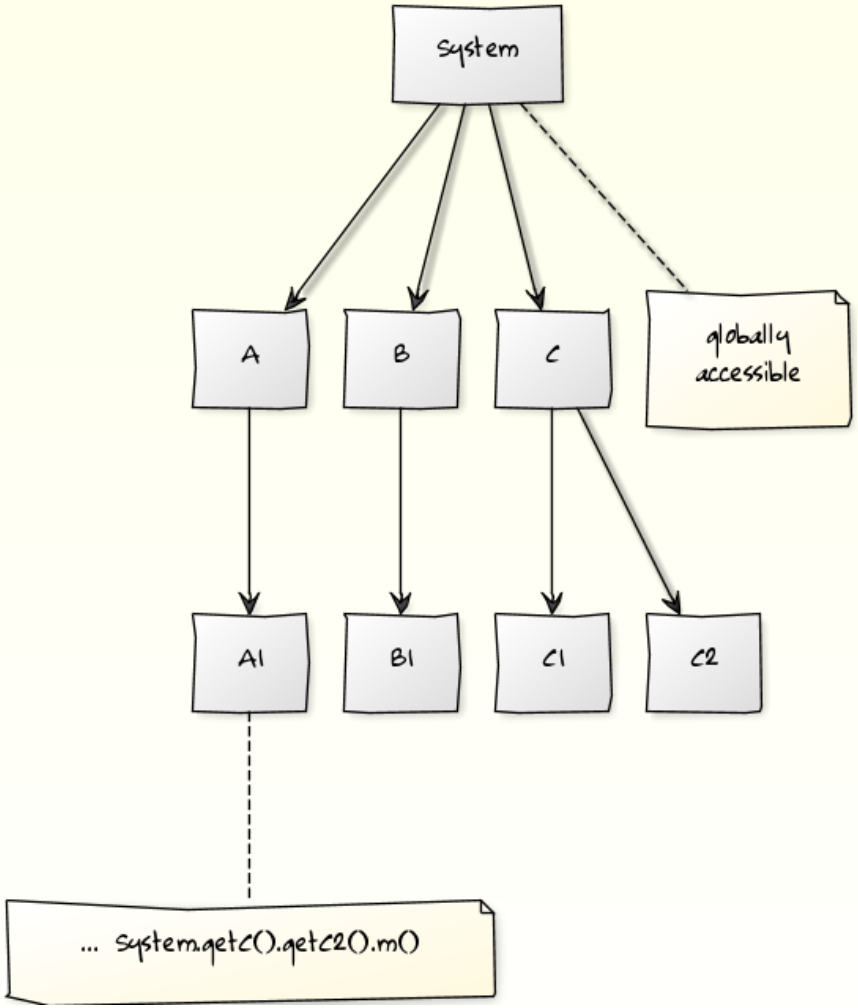

if (a > b)
{

BibDB

```
TBib = class(TObject)
public
  property Systemteil ...;
// ... aggregiert alle wichtigen Systemteile ...
end;

var
  Bib: TBib;

// Zugriff :
Bib.Systemteil.Methode();
```


System
A
B
C
globally accessible
A1
B1
C1
C2
... system.getC().getC2().m()

Warum?

Prinzipien

Warum?

Was unterscheidet gute Lösungen von schlechten?

Analytisch

Ein paar bekannte Prinzipien

- KISS
- Murphy's Law
- Starke Bindung, lose Kopplung
- DRY
- SOLID (SRP, OCP, LSP, ISP, DIP)
- Kapselung/Information Hiding
- ...

Prinzipien

Definition

Ein **Prinzip** ist eine Daumenregel, die gute von schlechten Lösungen unterscheidet – in Bezug auf *einen* Entwurfsaspekt.

Murphy's Law (ML)

„Whatever can go wrong, will go wrong“

Murphy's Law (ML)

Aussage Alles was schief gehen kann, wird irgendwann schiefgehen. Also ist eine Lösung dann besser, wenn sie weniger Möglichkeiten zulässt, dass etwas schiefgeht.

Begründung Menschen machen Fehler und das wird sich nie grundlegend ändern. Statistisch gesehen wird also auf lange Sicht, eine Fehler-Möglichkeit auch zu einem Fehler führen.

Beispiel `Date date1 = new Date(2013, 01, 12);`

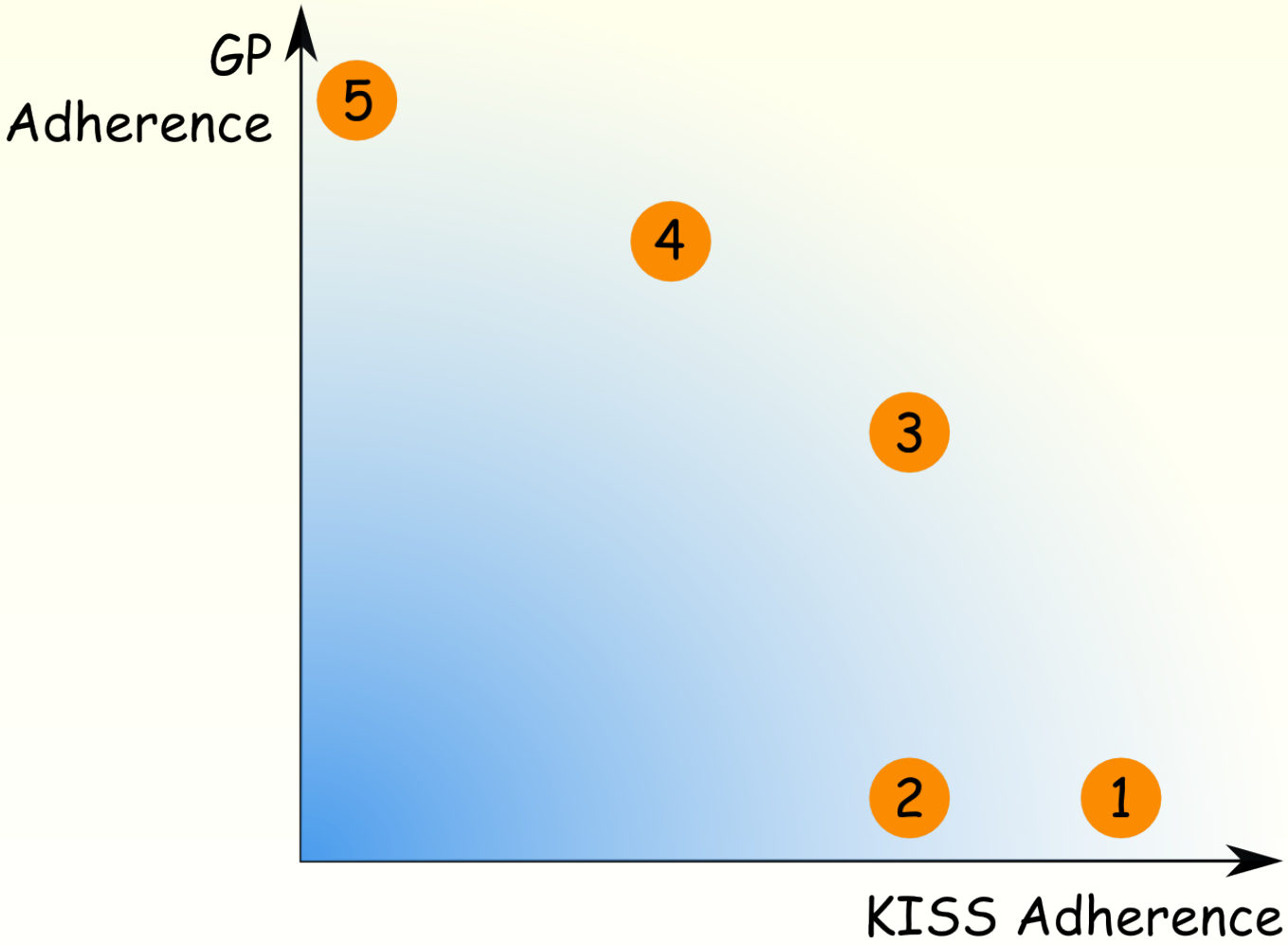
Ein Beispiel aus Java

```
new Date(2013, 01, 12);
```

Prinzipien widersprechen einander

Anforderung: $\sqrt{2}$ wird benötigt

1. `const SQRT_2 = 1.4142135623730951;`
2. `function sqrt_2: Real;`
3. `function sqrt(r: Real): Real;`
4. `function power(base, exponent: Real): Real;`
5. `class TComplexPolynomRootCalculator`

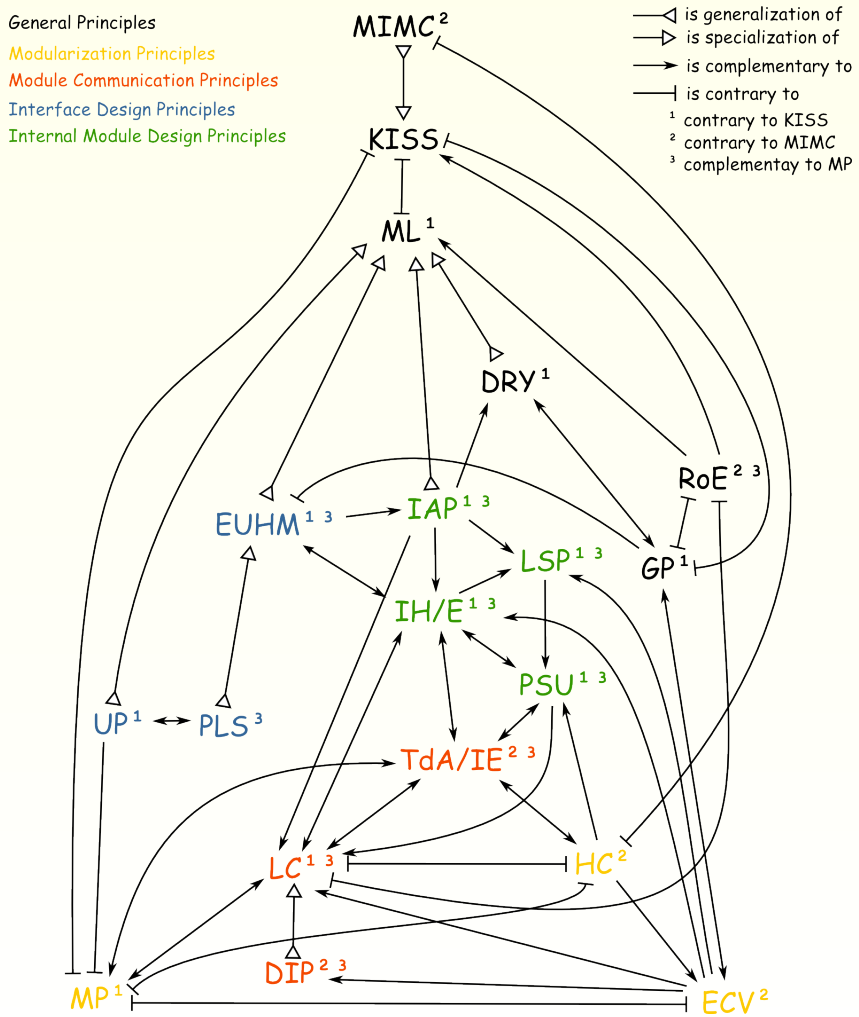

GP
Adherence
5
4
3
2
1
KISS Adherence

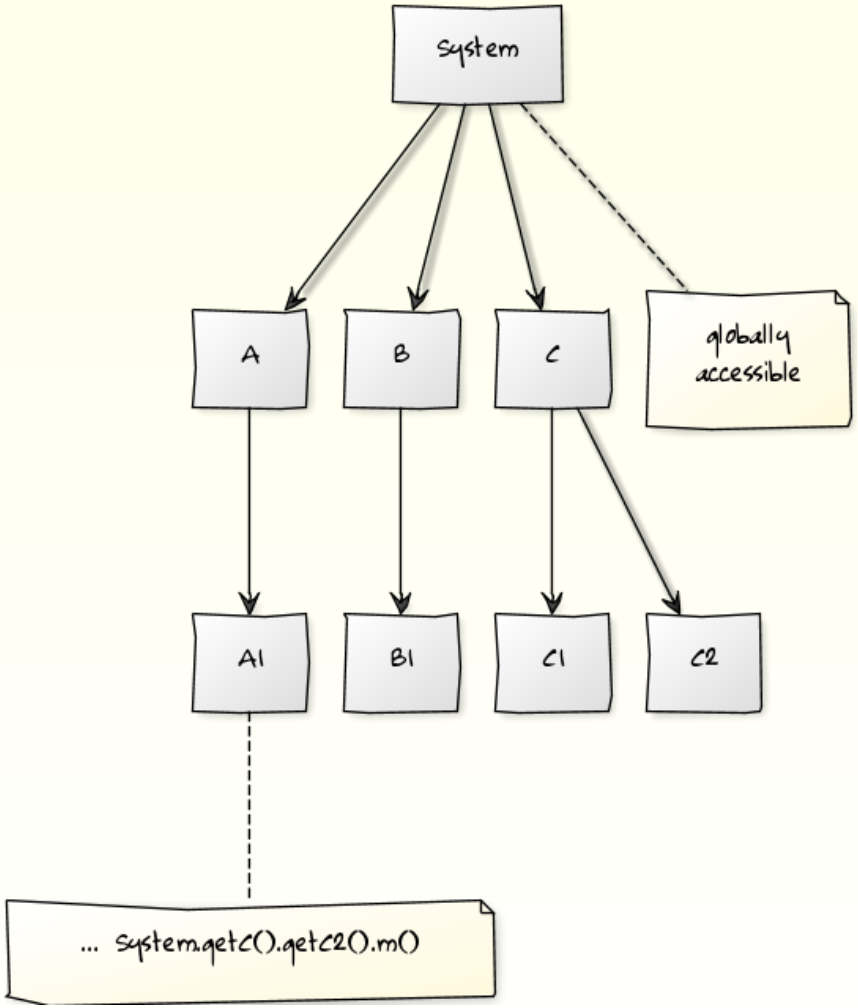
Prinzipiensprachen

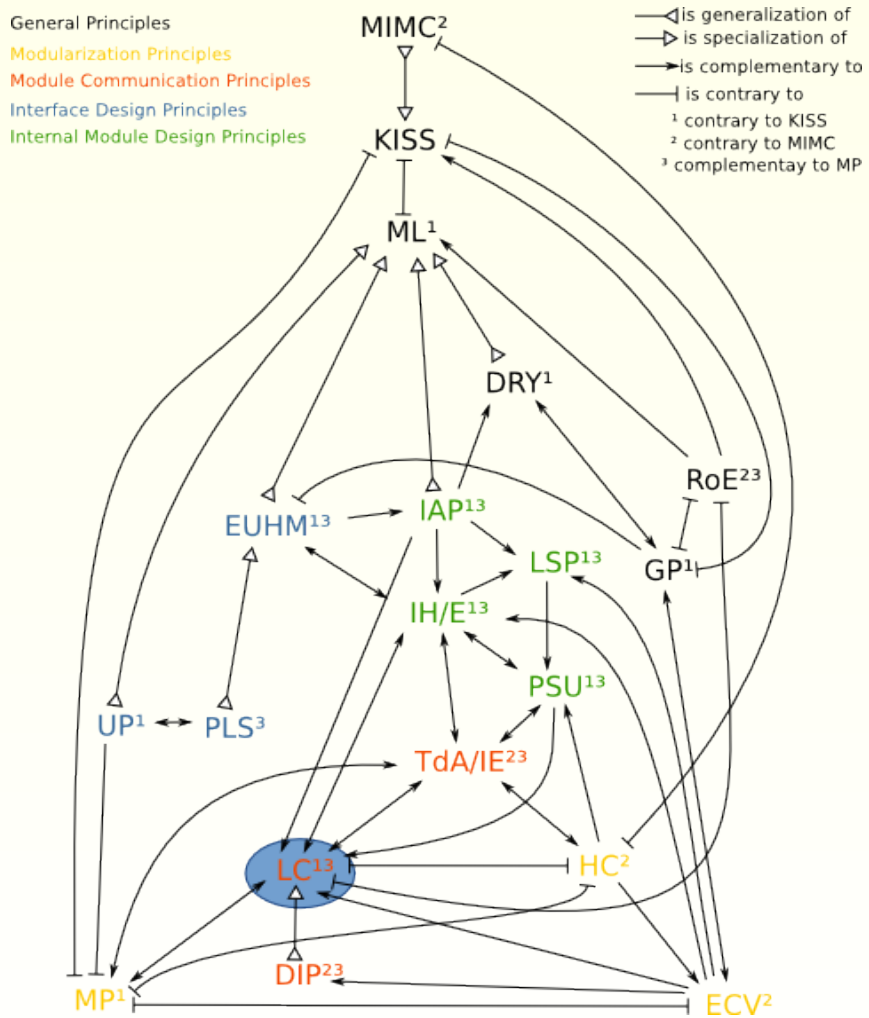
Wie findet man passende Prinzipien?

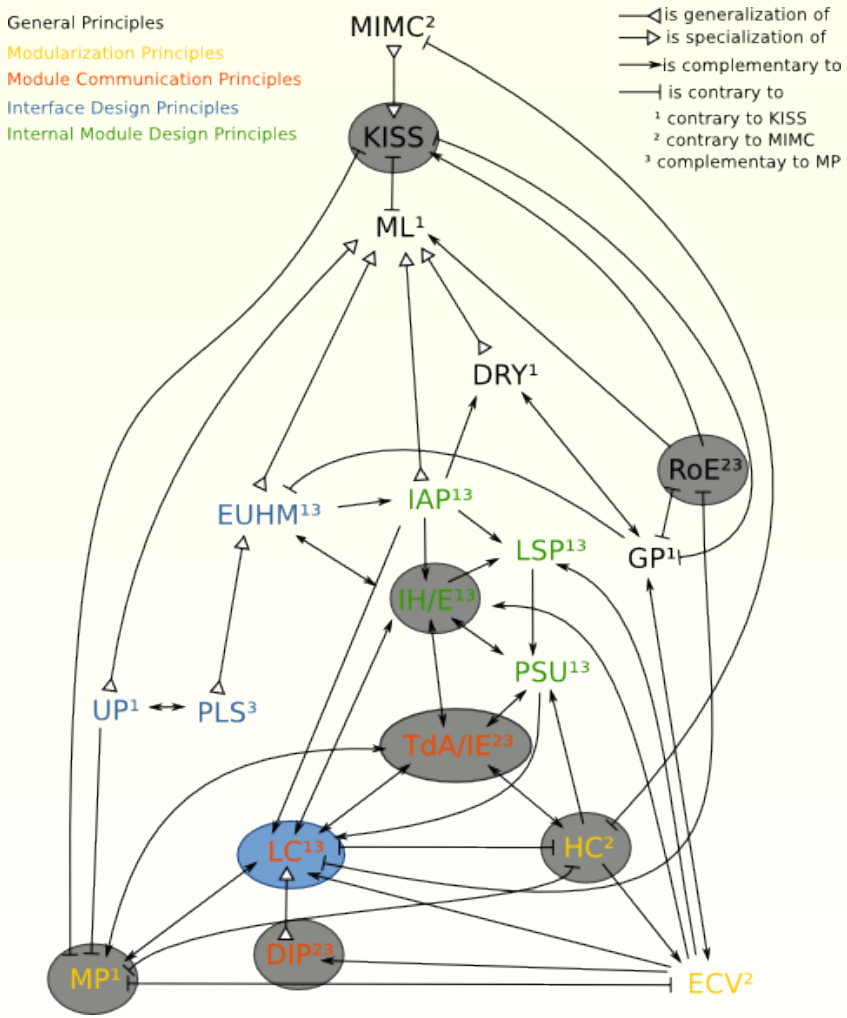
Prinzipiensprachen

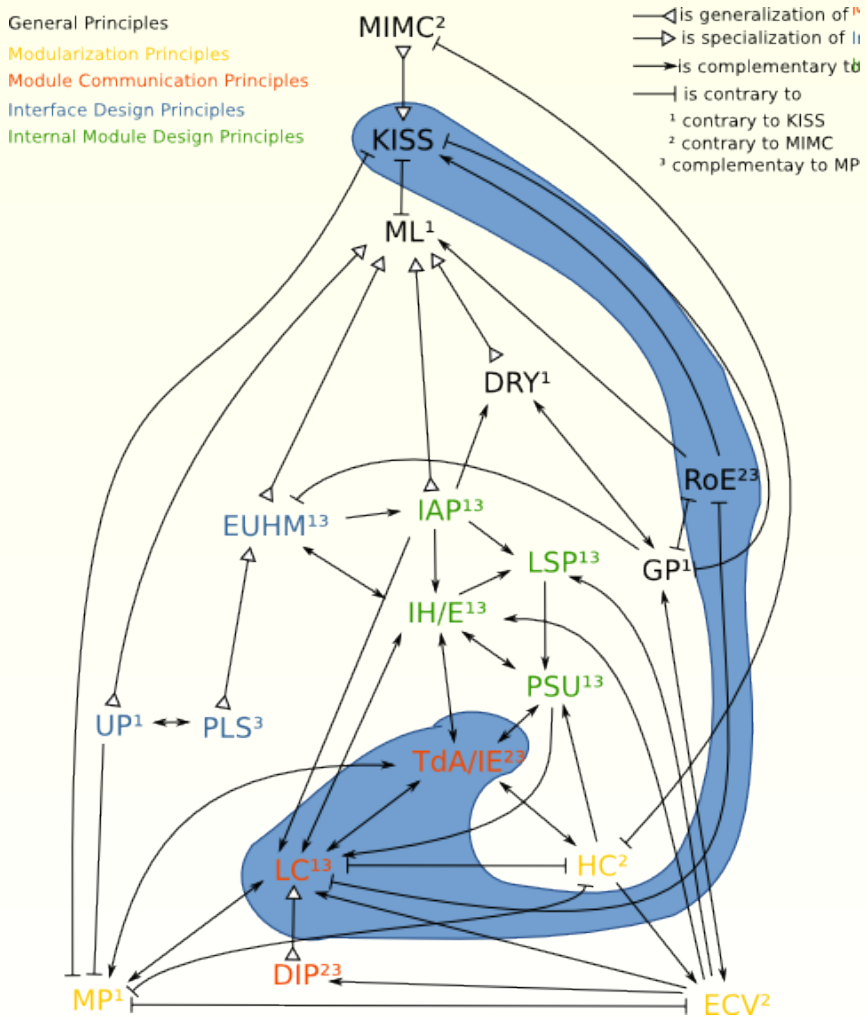
ODD Principle Language

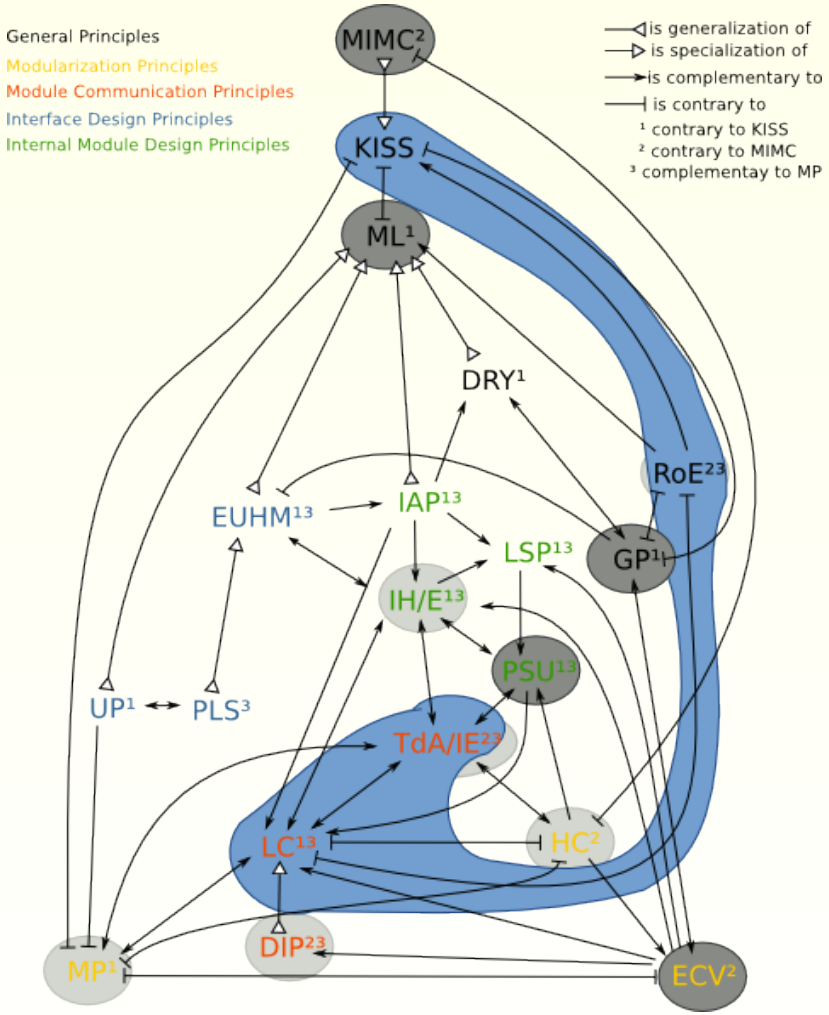


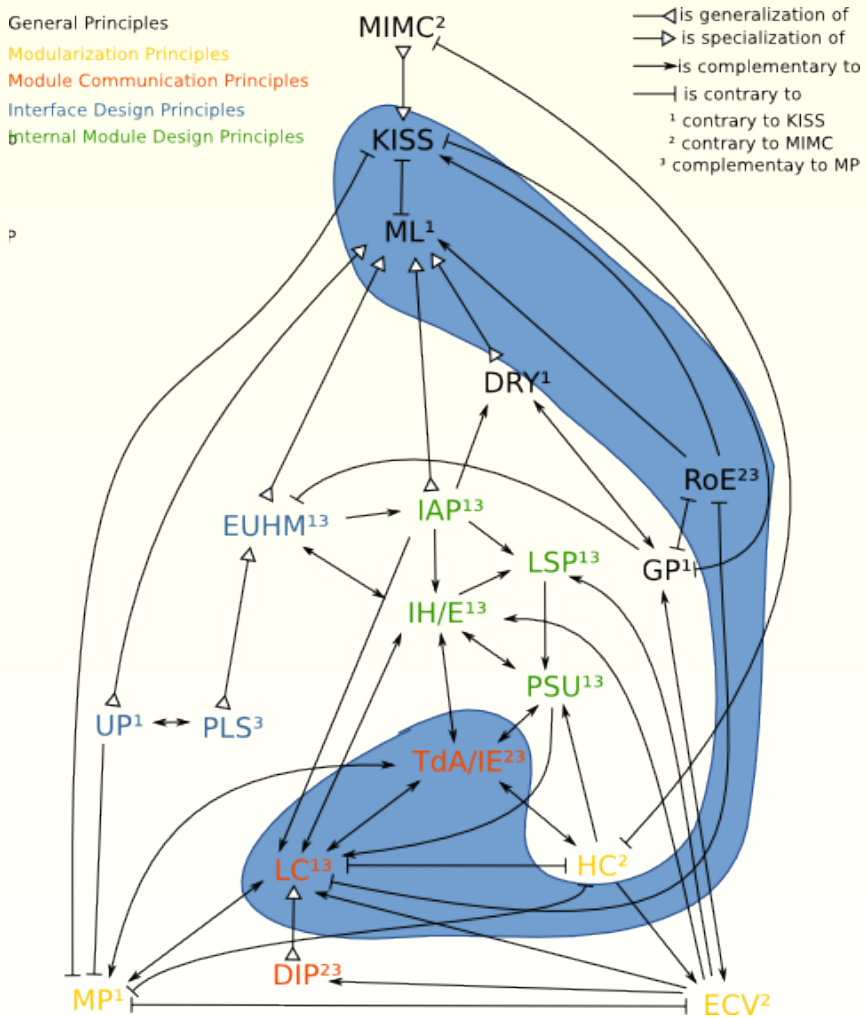

System
A
B
C
globally accessible
A1
B1
C1
C2
... system.getC().getC2().m()

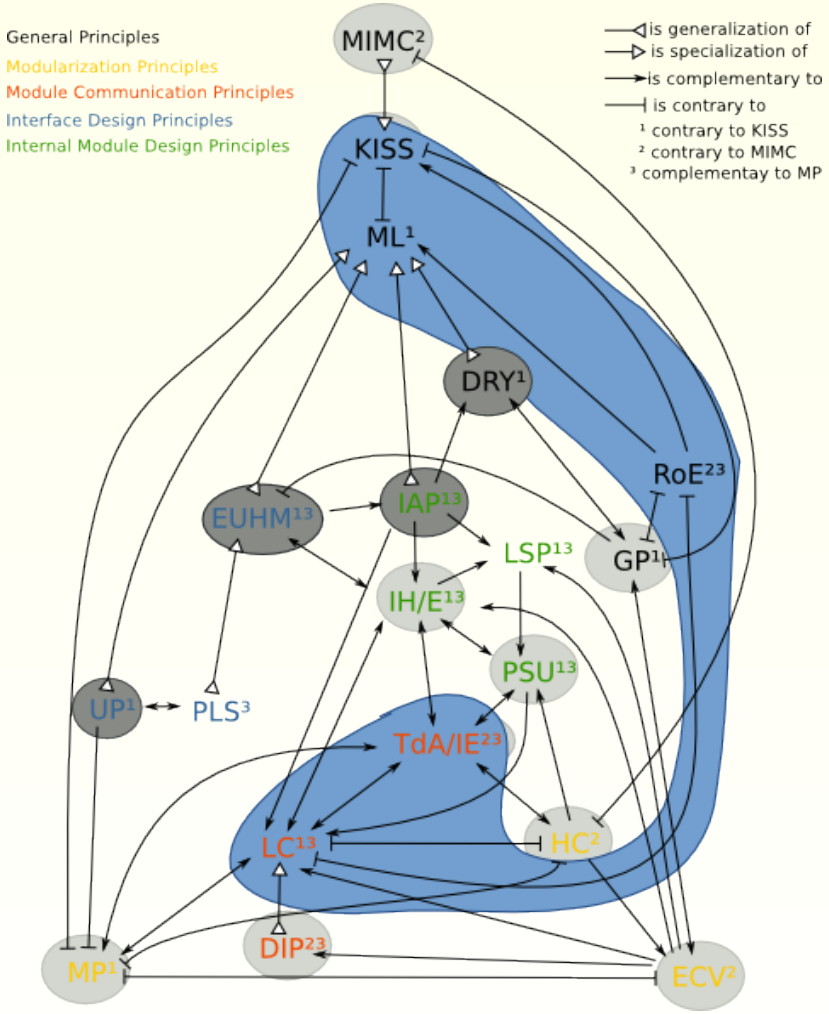

General Principles
Modularization Principles
Module Communication Principles
Interface Design Principles
Internal Module Design Principles
is generalization of
is specialization of
is complementary to
is contrary to
¹ contrary to KISS
² contrary to MIMC
³ complementay to MP
MIMC²
KISS
ML¹
DRY¹
RoE²³
EUHM¹³
IAP¹³
LSP¹³
GP¹
IH/E¹³
PSU¹³
UP¹
PLS³
TdA/IE²³
HC²
LC¹³
DIP²³
MP¹
ECV²

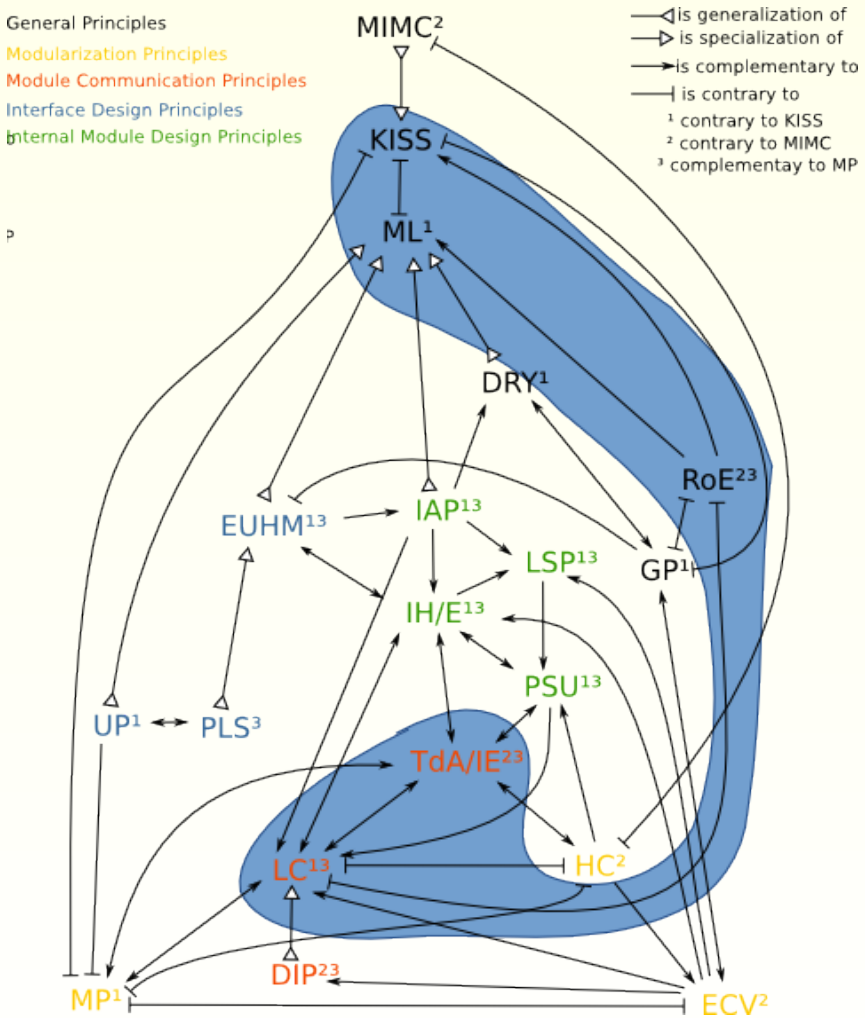

General Principles
Modularization Principles
Module Communication Principles
Interface Design Principles
Internal Module Design Principles
is generalization of
is specialization of
is complementary to
is contrary to
¹ contrary to KISS
² contrary to MIMC
³ complementay to MP
MIMC²
KISS
ML¹
DRY¹
RoE²³
EUHM¹³
IAP¹³
LSP¹³
GP¹
IH/E¹³
PSU¹³
UP¹
PLS³
TdA/IE²³
LC¹³
HC²
DIP²³
MP¹
ECV²


General Principles
Modularization Principles
Module Communication Principles
Interface Design Principles
Internal Module Design Principles
is generalization of
is specialization of
is complementary to
is contrary to
¹ contrary to KISS
² contrary to MIMC
³ complementay to MP
MIMC²
KISS
ML¹
DRY¹
RoE²³
EUHM¹³
IAP¹³
LSP¹³
GP¹
IH/E¹³
PSU¹³
UP¹
PLS³
TdA/IE²³
LC¹³
HC²
DIP²³
MP¹
ECV²


General Principles
Modularization Principles
Module Communication Principles
Interface Design Principles
Internal Module Design Principles
is generalization of
is specialization of
is complementary to
is contrary to
¹ contrary to KISS
² contrary to MIMC
³ complementay to MP
MIMC²
KISS
ML¹
DRY¹
RoE²³
EUHM¹³
IAP¹³
LSP¹³
GP¹
IH/E¹³
PSU¹³
UP¹
PLS³
TdA/IE²³
LC¹³
HC²
DIP²³
MP³
ECV²


General Principles
Modularization Principles
Module Communication Principles
Interface Design Principles
Internal Module Design Principles
is generalization of
is specialization of
is complementary to
is contrary to
¹ contrary to KISS
² contrary to MIMC
³ complementay to MP
MIMC²
KISS
ML¹
DRY¹
RoE²³
EUHM¹³
IAP¹³
LSP¹³
GP¹
IH/E¹³
PSU¹³
UP¹
PLS³
TdA/IE²³
LC¹³
HC²
DIP²³
MP¹
ECV²


General Principles
Modularization Principles
Module Communication Principles
Interface Design Principles
Internal Module Design Principles
is generalization of
is specialization of
is complementary to
is contrary to
¹ contrary to KISS
² contrary to MIMC
³ complementay to MP
MIMC²
KISS
ML¹
DRY¹
RoE²³
EUHM¹³
IAP¹³
LSP¹³
GP¹
IH/E¹³
PSU¹³
UP¹
PLS³
TdA/IE²³
LC¹³
HC²
DIP²³
MP¹
ECV²


General Principles
Modularization Principles
Module Communication Principles
Interface Design Principles
Internal Module Design Principles
is generalization of
is specialization of
is complementary to
is contrary to
¹ contrary to KISS
² contrary to MIMC
³ complementay to MP
MIMC²
KISS
ML¹
DRY¹
RoE²³
EUHM¹³
IAP¹³
LSP¹³
GP¹
IH/E¹³
PSU¹³
UP¹
PLS³
TdA/IE²³
LC¹³
HC²
DIP²³
MP¹
ECV²

- LC ✗
- KISS ✓
- RoE ✗
- TdA/IE ✗
- ML ✓

Was ist besser?

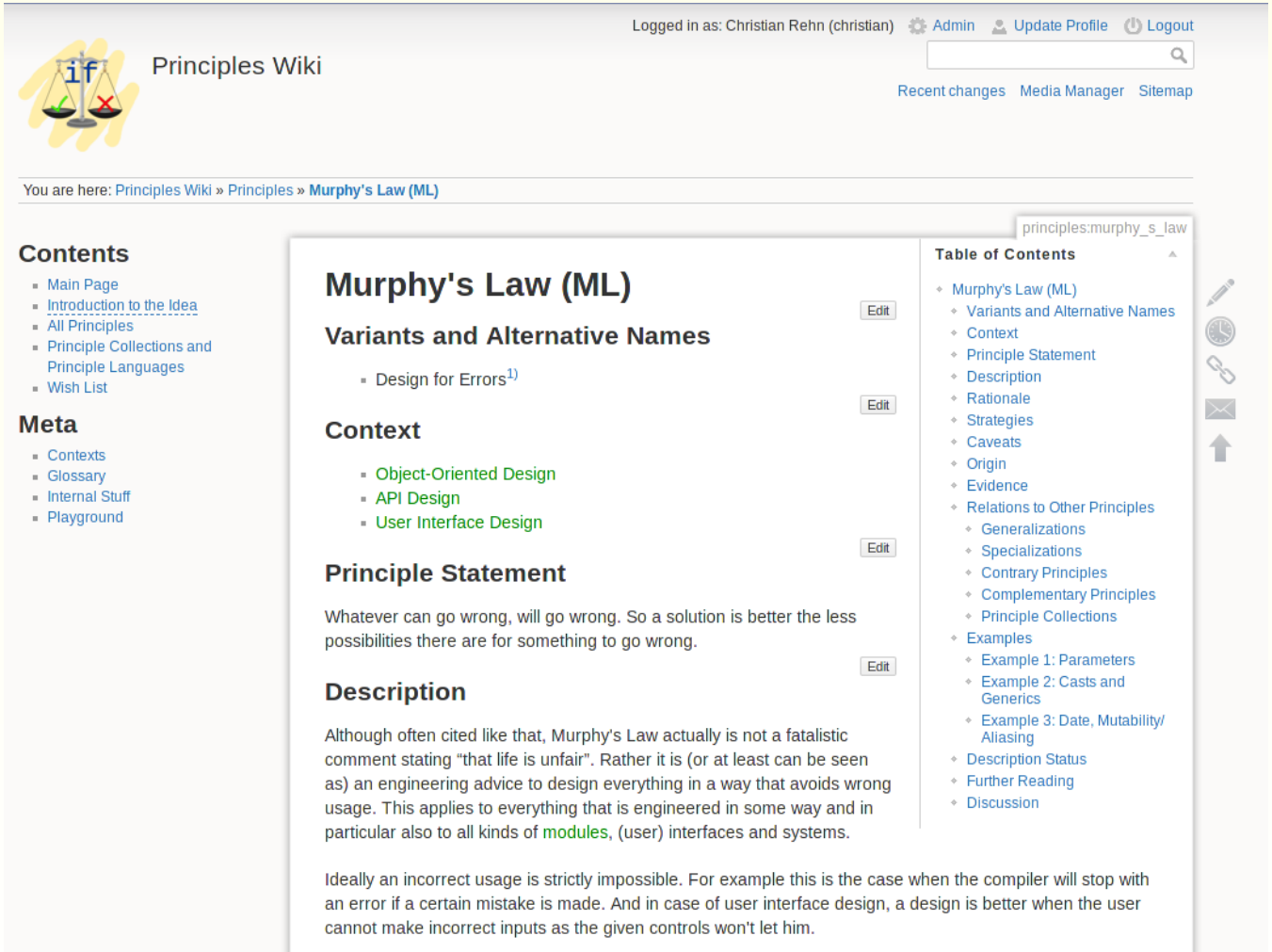
Dependency Injection

Das Wiki

Das Wiki

www.principles-wiki.net

www.principles-wiki.net

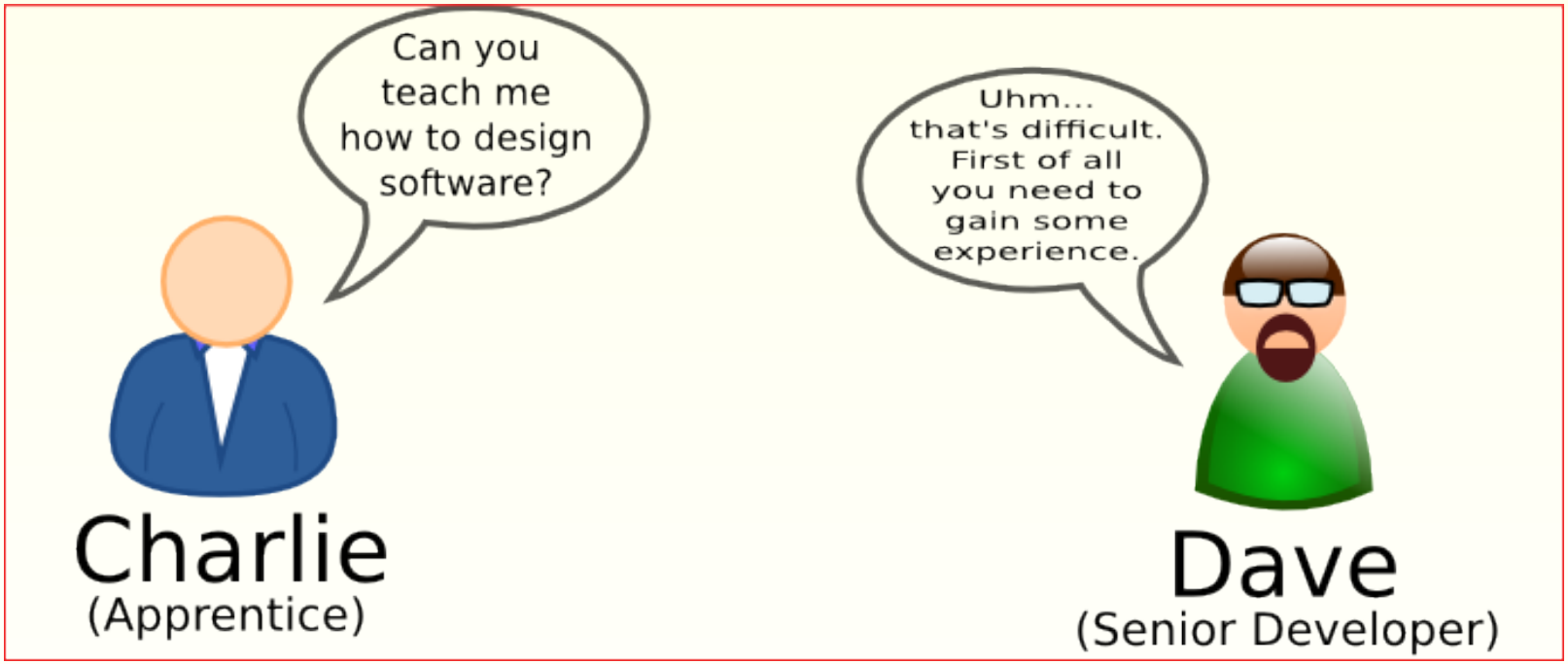


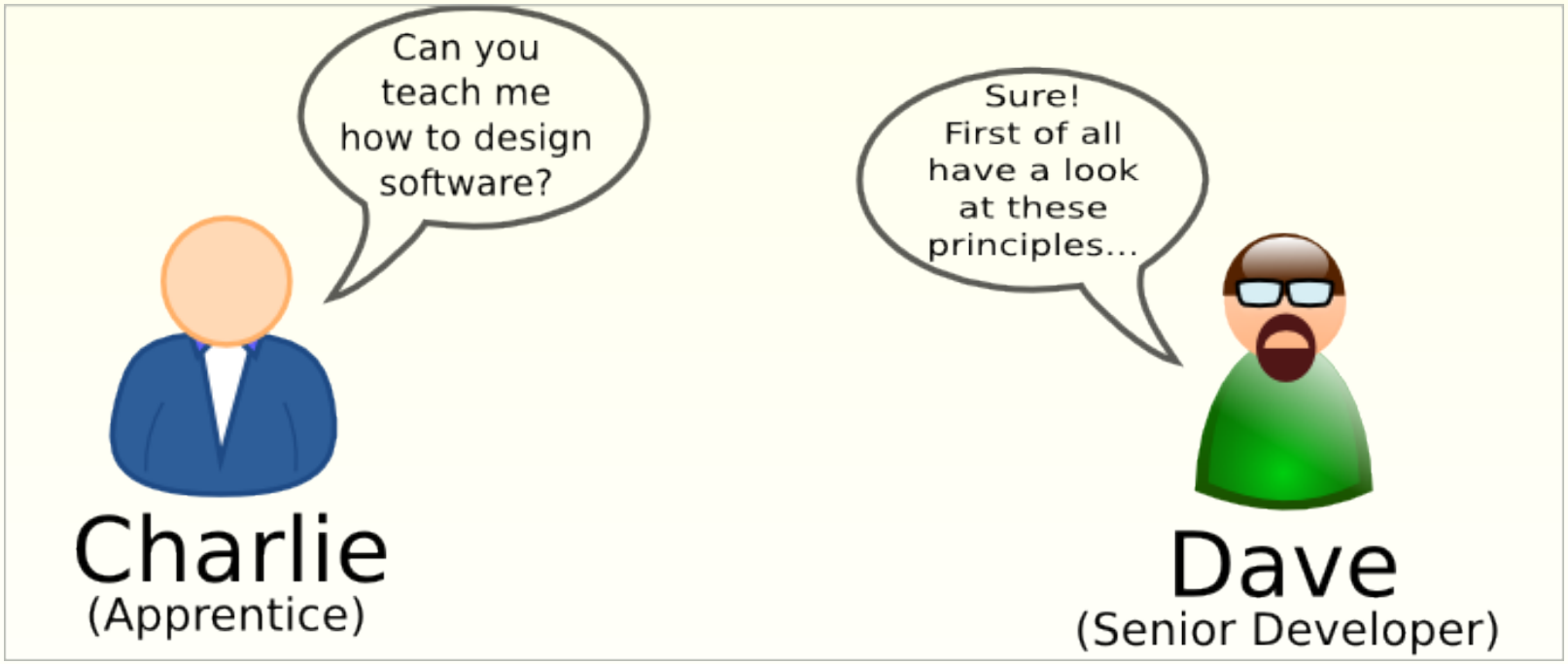
Vorteile

Vorteile

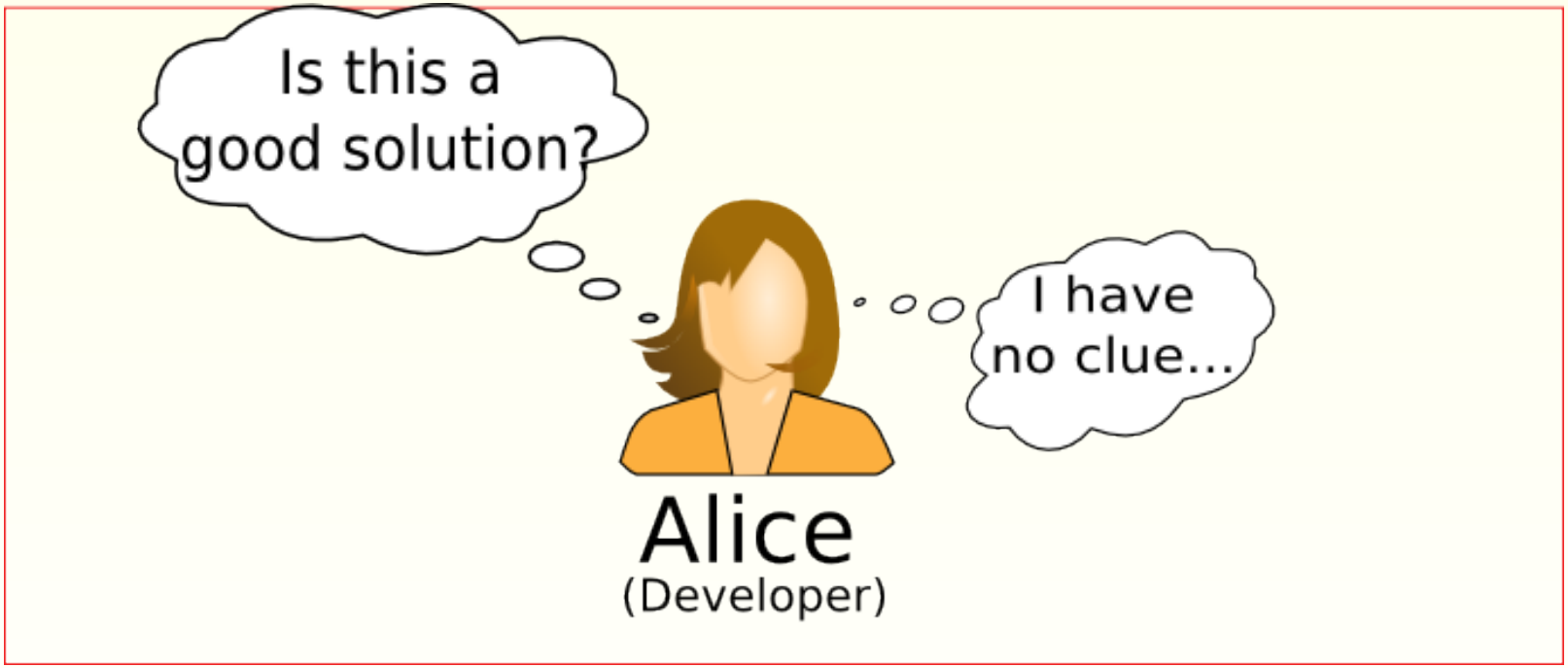
- Lernen
- Entscheiden
- Kommunizieren

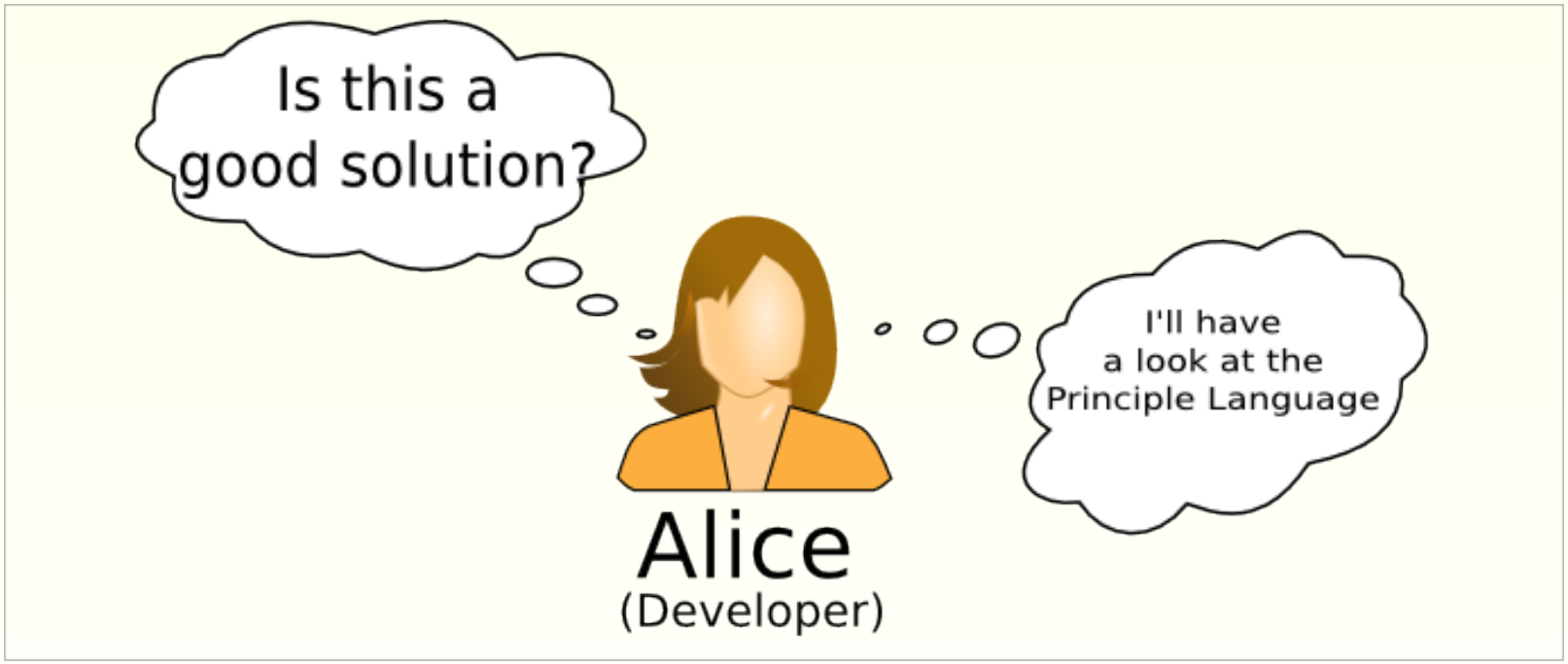
Lernen


Can you teach me how to design software?
Charlie
(Apprentice)
Uhm... that's difficult. First of all you need to gain some experience.
Dave
(Senior Developer)


Can you teach me how to design software?
Charlie
(Apprentice)
Sure! First of all have a look at these principles...
Dave
(Senior Developer)

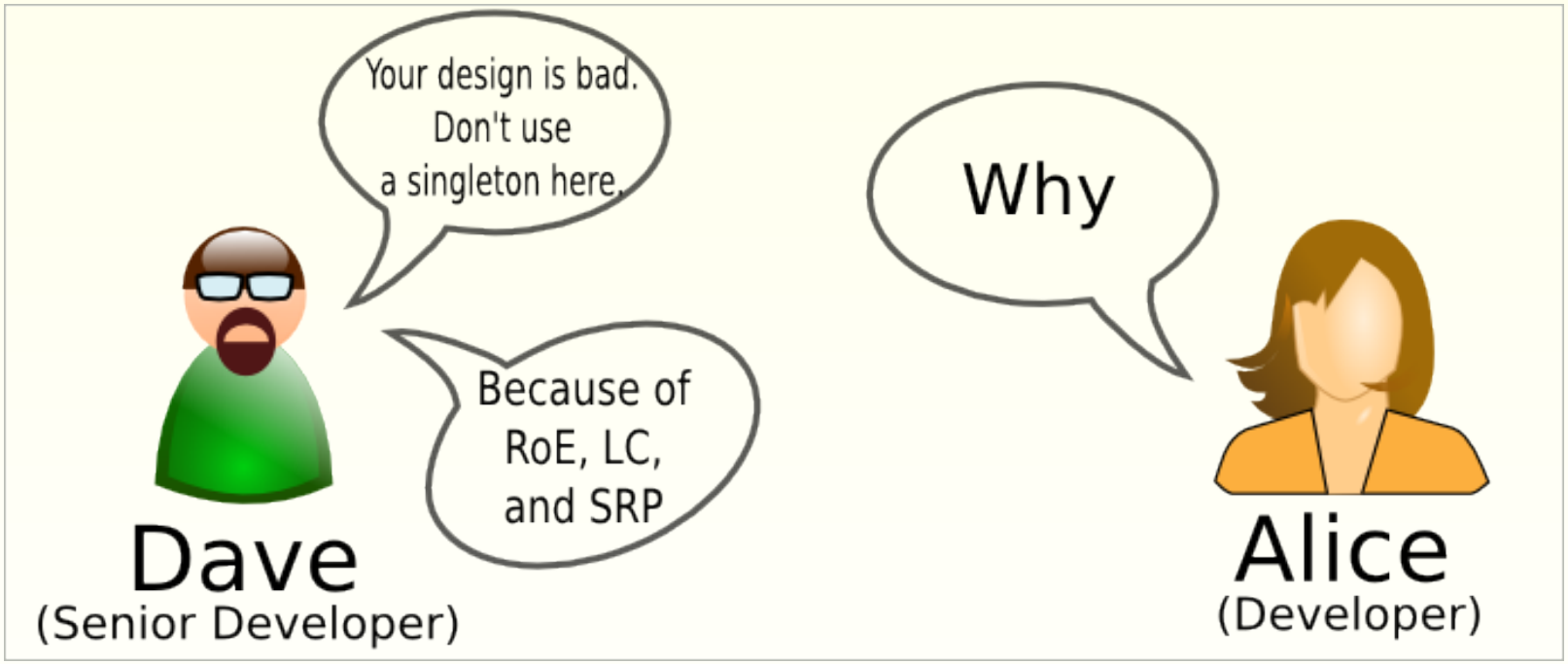
Entscheiden


Is this a good solution?
I have no clue...
Alice
(Developer)


Is this a good solution?
I'll have a look at the Principle Language
Alice
(Developer)

Kommunizieren


Your design is bad.
Don't use
a singleton here.
Trust me!
Dave
(Senior Developer)
Why
Alice
(Developer)


Your design is bad.
Don't use
a singleton here.
Because of
RoE, LC,
and SRP
Dave
(Senior Developer)
Why
Alice
(Developer)

Prinzipien-*Sprachen*

Fazit

- Prinzipien/Daumenregeln sind wie Patterns
  Wiederverwendung von Erfahrung
- Mit Prinzipien kann man Entwurfsentscheidungen begründen
- Prinzipiensprachen vernetzen Prinzipien und zeigen weiterführende Aspekte auf
- Prinzipiensprachen bilden ein Vokabular
- `www.principles-wiki.net`

Ausblick

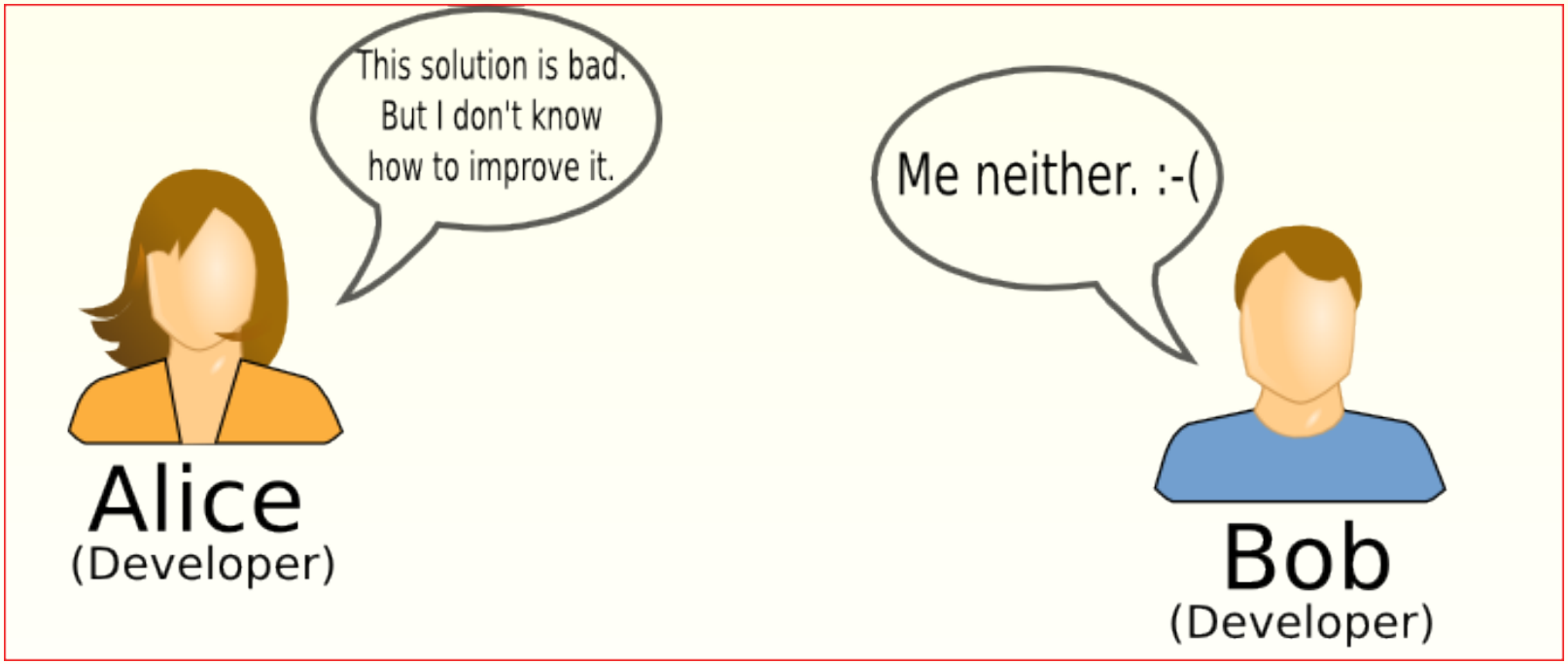
- Das Wiki wird ständig erweitert und ausgebaut
- Weitere Prinzipien und Prinzipiensprachen werden folgen
- Vernetzung von Patterns mit Prinzipien wird folgen
- Mitarbeit willkomemn

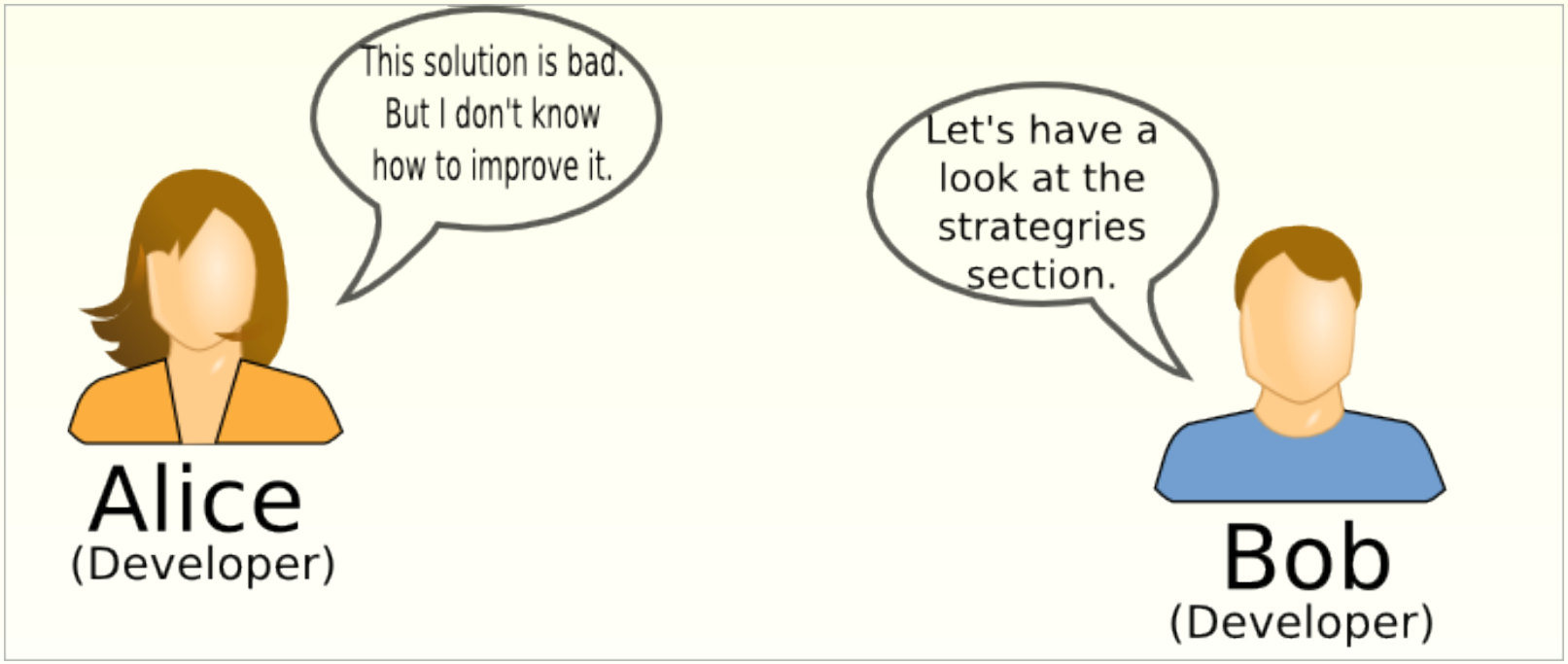
Vielen Dank!

Fragen?

Anhang

Strategien


This solution is bad.
But I don't know
how to improve it.
Alice
(Developer)
Me neither. :-(
Bob
(Developer)


This solution is bad.
But I don't know
how to improve it.
Alice
(Developer)
Let's have a
look at the
strategries
section.
Bob
(Developer)

Commit

Alice
(Developer)

Commit message:

Moved redundant code
to a new method because
redundant code trends to
get out of sync which
creates bugs.

Alice
(Developer)

Commit message:

Refactoring: DRY

Das Große Ganze

- Prinzipien
- Patterns
- Anti-Patterns
- Refactorings

- Glossary Terms
- Non-Principles

ODD Principle Language