Principle Languages

How to Make and Communicate Design Decisions

Christian Rehn

Delphi-Treff

ADUG Sydney Meeting
5th November 2013

About Me

- Christian Rehn
- Began to program in 2001
- Studies Computer Science at TU Kaiserslautern
- Moderator and editor at Delphi-Treff (some German Delphi website)
- Employed at 1&1 Source Center since May
- `http://www.christian-rehn.de/`

Organisational Stuff

- Few Text on the slides
  - Better for Presentation
  - Additionally detailed material online: `http://www.principles-wiki.net/about:start`
- Please give Feedback

Overview

A Story

Remember, remember, the fifth of November...

Once upon a time. . .

QBASIC

Delphi

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

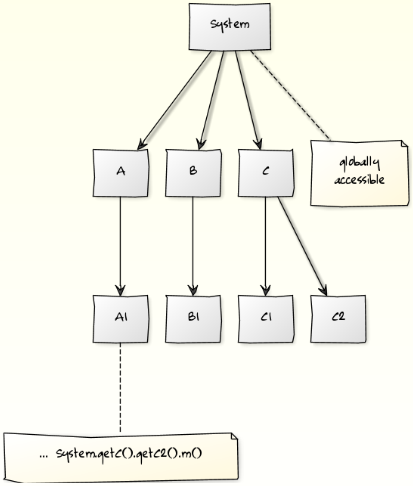
DANGER
MAN AT WORK

But the Code. . .

if (a > b)
{

BibDB

```
TSystem = class(TObject)
 public
   property  SystemPart  ...;
//  ...  aggregates  all  important system  parts   ...
end;

var
  System: TSystem;

// Access:
System.SystemPart.Method();
```


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

Why?

Principles

Why?

How to tell good solutions and bad solutions apart?

Analytic

Some Well-Known Principles

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

Principles

Definition

A **principle** is a rule of thumb which tells good solutions from bad ones—with respect to *one* design aspect.

Murphy’s Law (ML)

„Whatever can go wrong, will go wrong“

Murphy's Law (ML)

Statement	Whatever can go wrong, will go wrong. So a solution is the better the fewer possibilities there are for something to go wrong.
Rationale	Humans make mistakes and this will never change. So in the long run a possibility for a fault will eventually result in a fault.
Example	`Date date1 = new Date(2013, 01, 12);`

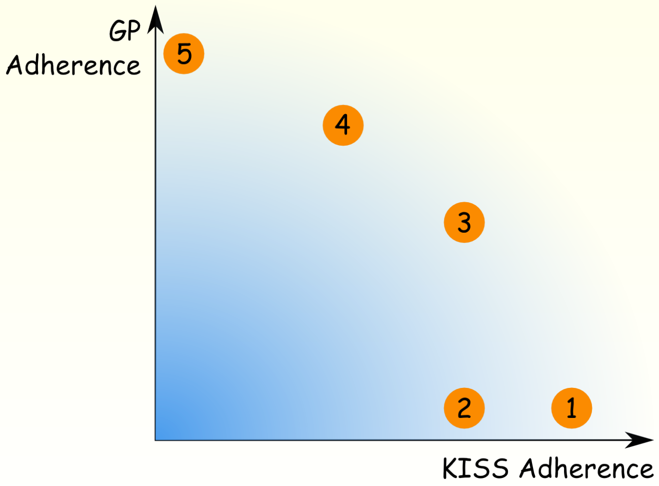
Am example from Java

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

Principles are conflicting

Requirement: $\sqrt{2}$ is needed

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

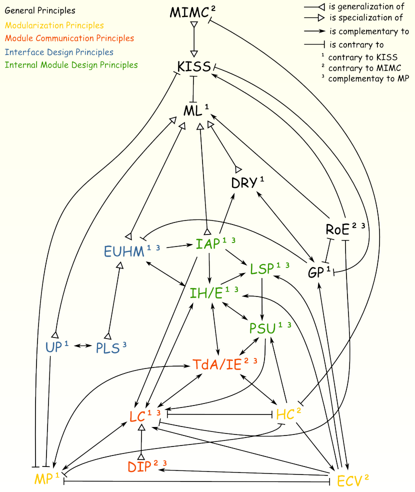

GP Adherence
5
4
3
2
1
KISS Adherence

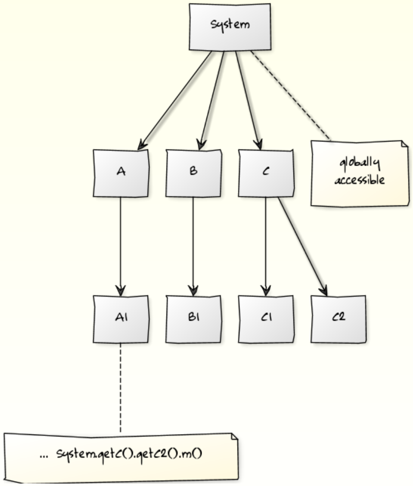
Principle Languages

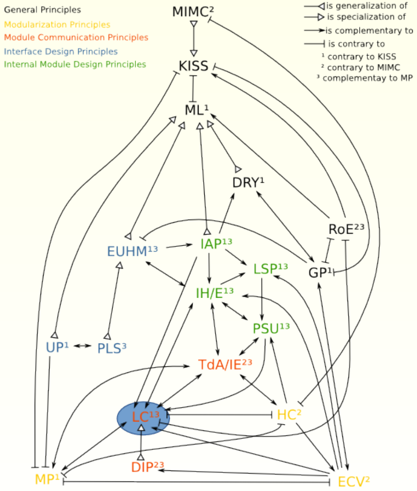
How to find suitable principles?

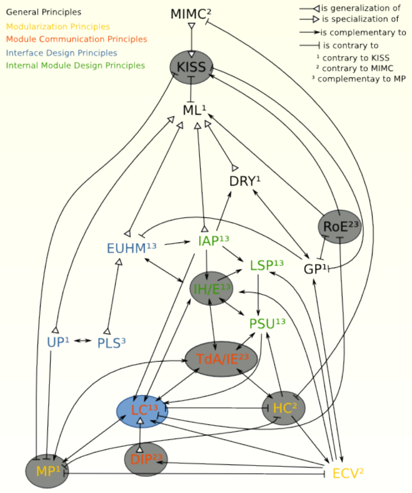
Principle Languages

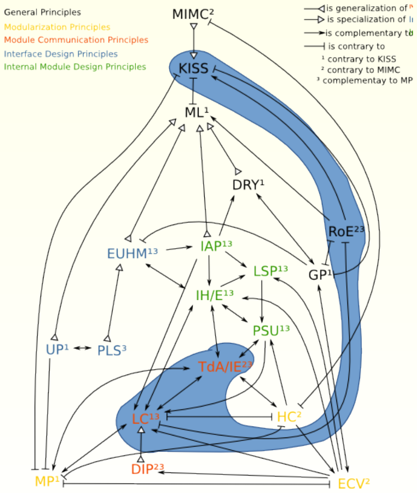
ODD Principle Language

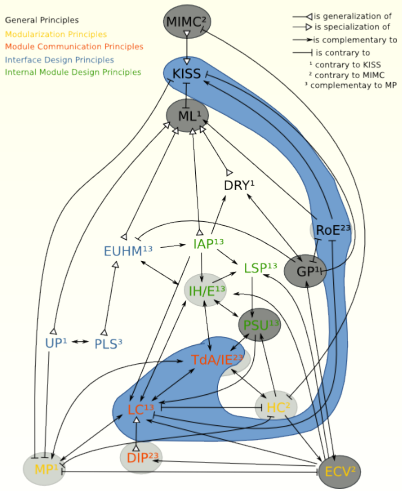


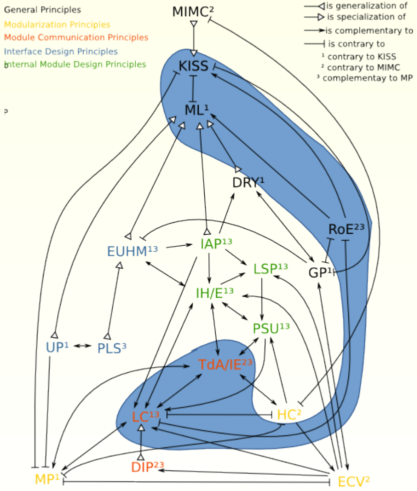

System
A
B
C
globally accessible
A1
B1
C1
C2
... system.getC().getC2().m()

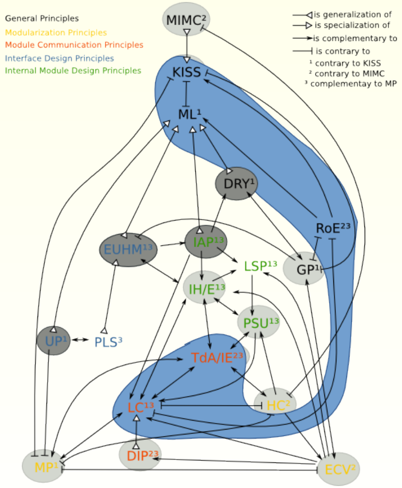

General Principles
Modularization Principles
Module Communication Principles
Interface Design Principles
Internal Module Design Principles
is generalization of
is specialization of
is complementary to
is contrary to
¹ contrary to KISS
² contrary to MIMC
³ complementay to MP
MIMC²
KISS
ML¹
DRY¹
RoE²³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

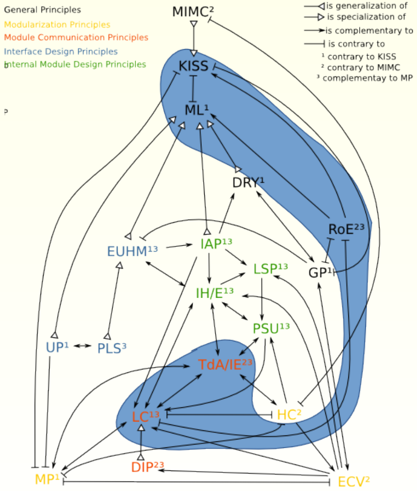

General Principles
Modularization Principles
Module Communication Principles
Interface Design Principles
Internal Module Design Principles
is generalization of
is specialization of
is complementary to
is contrary to
¹ contrary to KISS
² contrary to MIMC
³ complementay to MP
MIMC²
KISS
ML¹
DRY¹
RoE²³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


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
GP¹
EUHM¹³
IAP¹³
LSP¹³
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
GP¹
EUHM¹³
IAP¹³
LSP¹³
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
GP¹
EUHM¹³
IAP¹³
LSP¹³
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
GP¹
EUHM¹³
IAP¹³
LSP¹³
IH/E¹³
PSU¹³
UP¹
PLS³
TdA/IE²³
LC¹³
HC²
DIP²³
MP¹
ECV²

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

What is better?

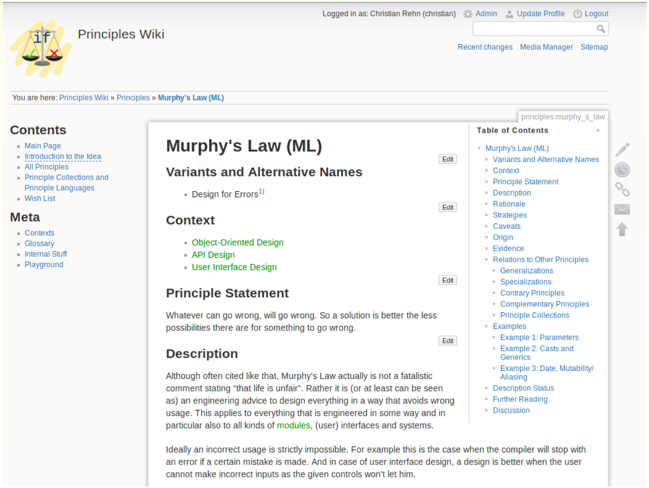
Dependency Injection

The Wiki

Das Wiki

`www.principles-wiki.net`

www.principles-wiki.net



Logged in as: Christian Rehn (christian) Admin Update Profile Logout

Principles Wiki

Recent changes Media Manager Sitemap

You are here: Principles Wiki » Principles » Murphy's Law (ML)

principles:murphy_s_law

Contents

- Main Page
- Introduction to the Idea
- All Principles
- Principle Collections and Principle Languages
- Wish List

Meta

- Contexts
- Glossary
- Internal Stuff
- Playground

Murphy's Law (ML)

Variants and Alternative Names

- Design for Errors[1)]

Context

- Object-Oriented Design
- API Design
- User Interface Design

Principle Statement

Whatever can go wrong, will go wrong. So a solution is better the less possibilities there are for something to go wrong.

Description

Although often cited like that, Murphy's Law actually is not a fatalistic comment stating "that life is unfair". Rather it is (or at least can be seen as) an engineering advice to design everything in a way that avoids wrong usage. This applies to everything that is engineered in some way and in particular also to all kinds of modules, (user) interfaces and systems.

Ideally an incorrect usage is strictly impossible. For example this is the case when the compiler will stop with an error if a certain mistake is made. And in case of user interface design, a design is better when the user cannot make incorrect inputs as the given controls won't let him.

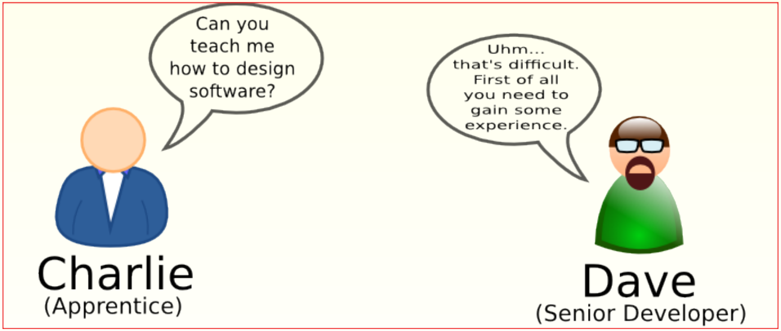
Table of Contents

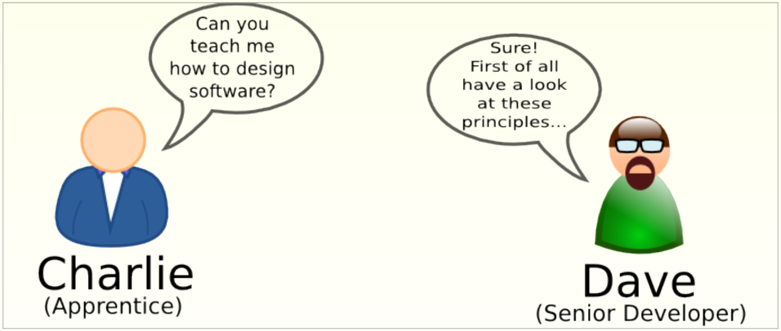
Advantages

Advantages

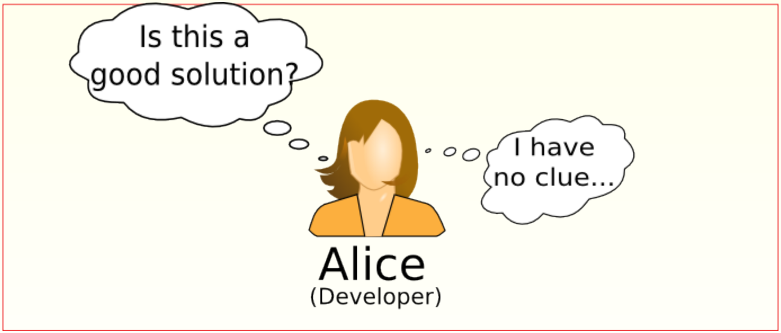
- Learning Design
- Making Design Decisions
- Communication

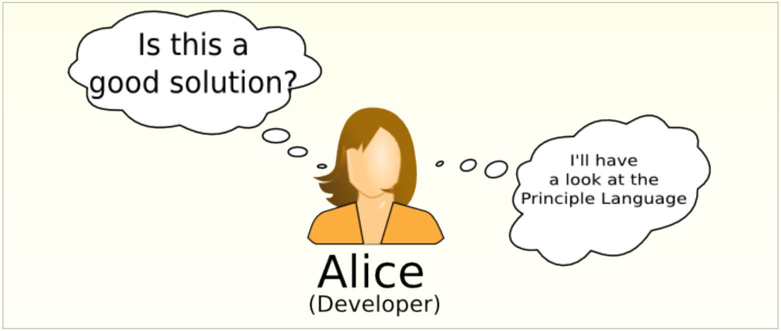
Learning Design


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)

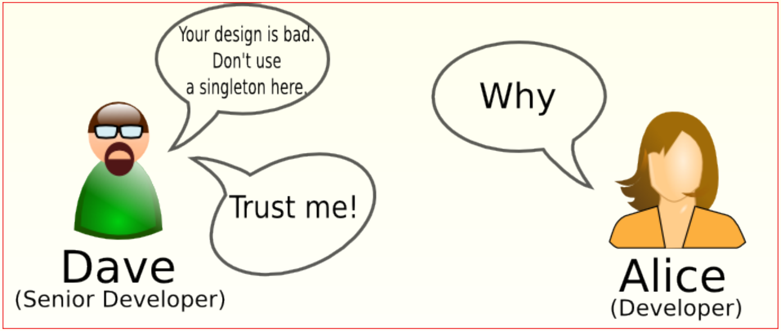

Can you teach me how to design software?
Sure! First of all have a look at these principles...
Charlie
(Apprentice)
Dave
(Senior Developer)

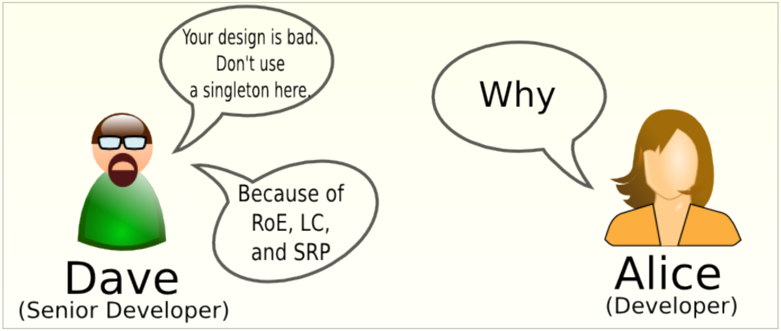
Making Design Decisions


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)

Communication


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


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

Principle *Language*

Conclusion

- Principles or rules of thumb are a form of experience reuse—just like patterns are
- You can reason about design using principles
- Principle languages point to further aspects to consider
- Principle language for a vocabulary
- `www.principles-wiki.net`

Outlook

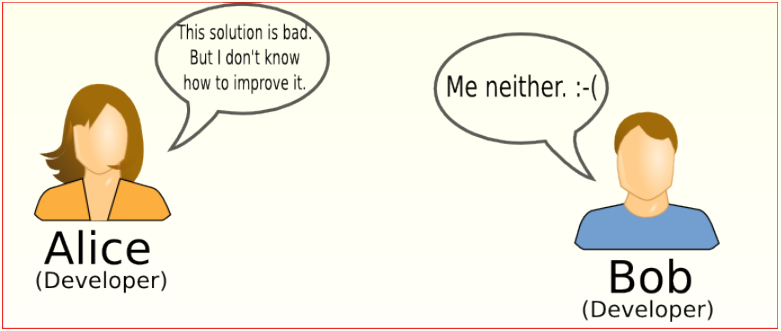
- The wiki gets enhanced and improved slowly but continuously
- Further principles and principle language will follow
- Patterns and principles will be interconnected
- Contributions are welcome

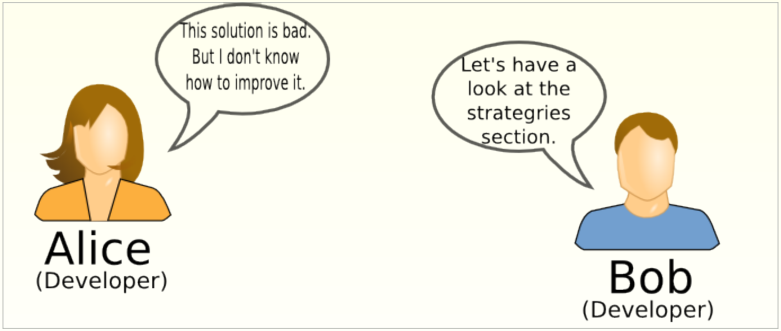
Thank you!

Questions?

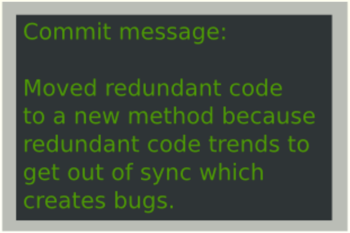
Appendix

Strategies


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

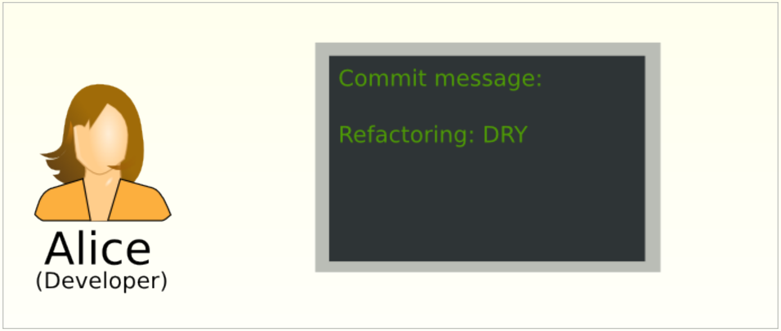

This solution is bad. But I don't know how to improve it.
Let's have a look at the strategries section.
Alice (Developer)
Bob (Developer)

Commit



Commit message:

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



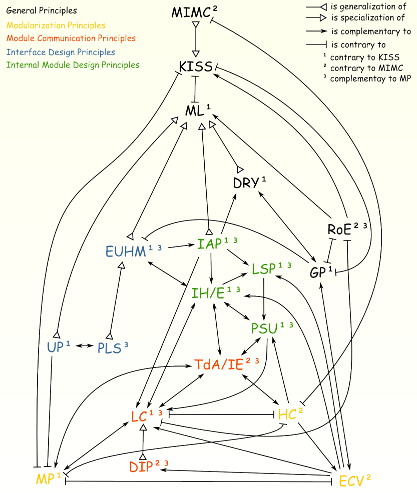
```
Commit message:

Refactoring: DRY
```

The Big Picture

- Principles
- Patterns
- Anti-Patterns
- Refactorings

- Glossary Terms
- Non-Principles

ODD Principle Language


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
MIMC ²
KISS
ML ¹
DRY ¹
RoE ² ³
GP ¹
EUHM ¹ ³
IAP ¹ ³
LSP ¹ ³
IH/E ¹ ³
PSU ¹ ³
UP ¹
PLS ³
TdA/IE ² ³
LC ¹ ³
HC ²
DIP ² ³
MP ¹
ECV ²