Функциональная безопасность

И парадоксы её требований. Лучшие практики против худших рекомендаций

KasperskyOS

АННА

19 лет в ИТ

Виртуализация, архитектура, а теперь микрокернел

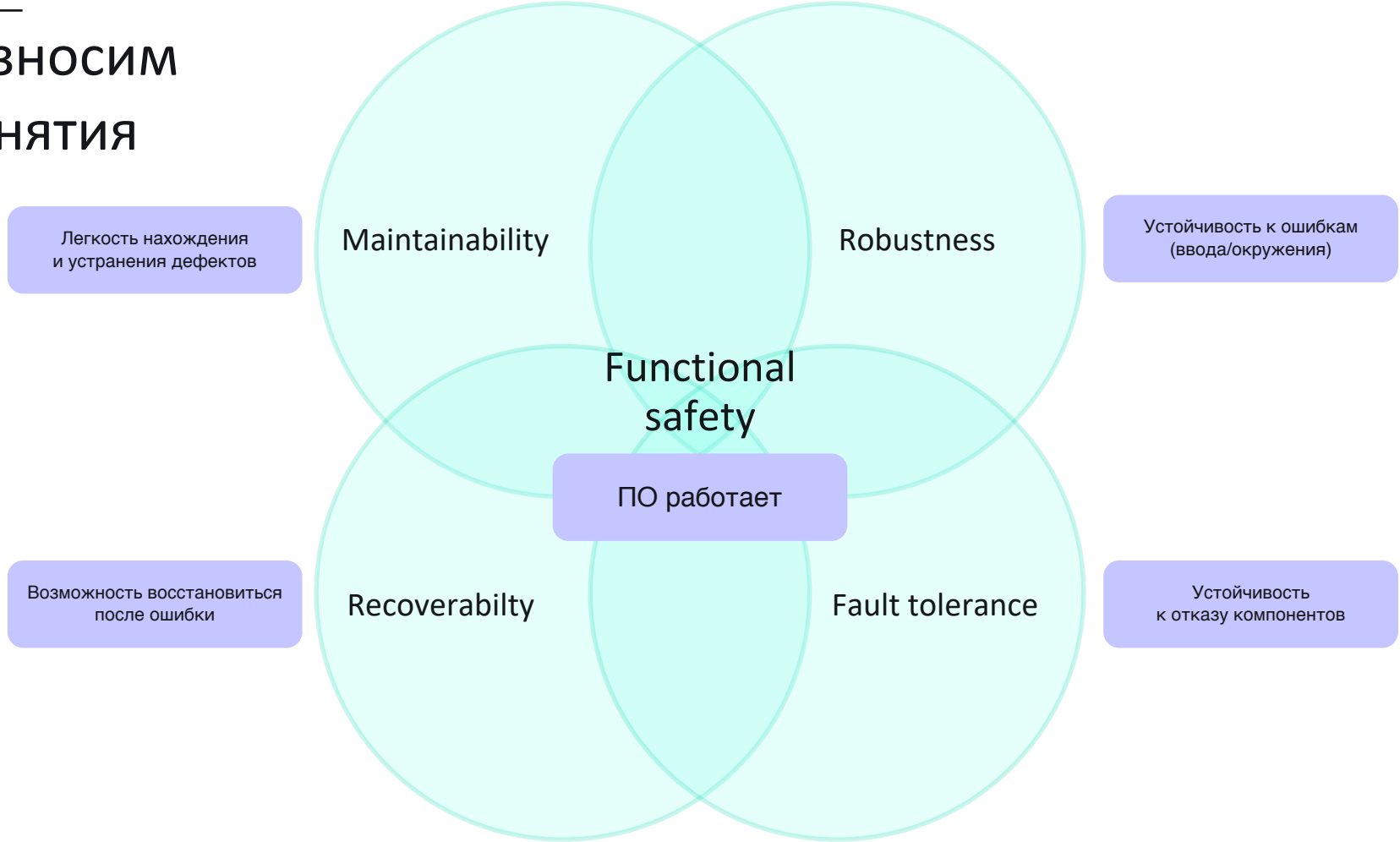
Функциональная безопасность

Определения

Functional safety is the detection of a potentially dangerous condition resulting in the activation of a protective or corrective device or mechanism to prevent hazardous events arising or providing mitigation to reduce the consequence of the hazardous event. (IEC 61508)

Functional safety is absence of unreasonable risk due to hazards caused by malfunctioning behaviour of E/E systems (ISO26262)

Разносим понятия



Как описывается?

ISO26262	IEC 62061	IEC 60601	EN50128
транспорт - автомобильная промышленность	машиностроение	медицинское оборудование	железные дороги

...и другие

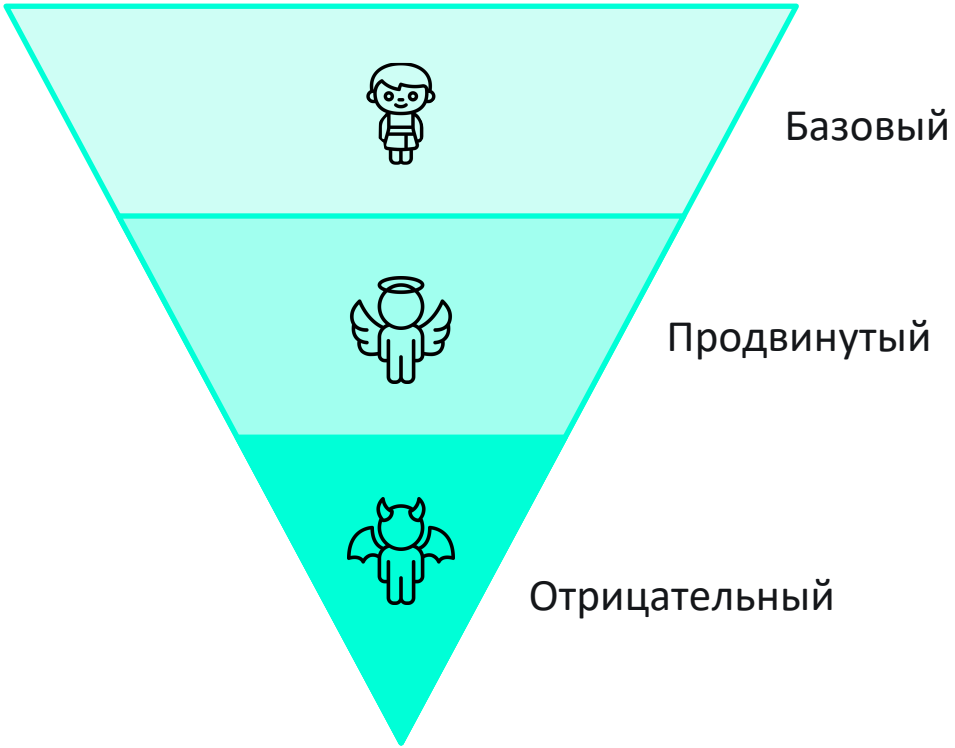
И зачем она мне?

Мне важно качество софта, который я делаю

Софт работает в критических областях

Мы собираемся выходить на регулируемых стандартами рынках

Три уровня функциональной безопасности



Базовая функциональная безопасность

Продвинутый уровень

Уход в минус

База.

Убираем потенциальные опасности (1)

ISO26262:6-1-1b Use of language subsets

Exclusion of ambiguously-defined constructs
Exclusion of language constructs which by experience lead to mistakes, e.g. assignments in conditions
Exclusion of constructs which may lead to unhandled run-time errors”

ПРЕСТУПНО РЕДУЦИРУЕМ ОПАСНОСТИ ЯЗЫКА

База.

Убираем потенциальные опасности (2)

- No multiple use of variable names (ISO26262:6-6-1d)
- No implicit type conversion (ISO26262:6-6-1g)
- No unconditional jumps (ISO26262:6-6-1i)
- Restricted use of pointers (ISO26262:6-6-1f)

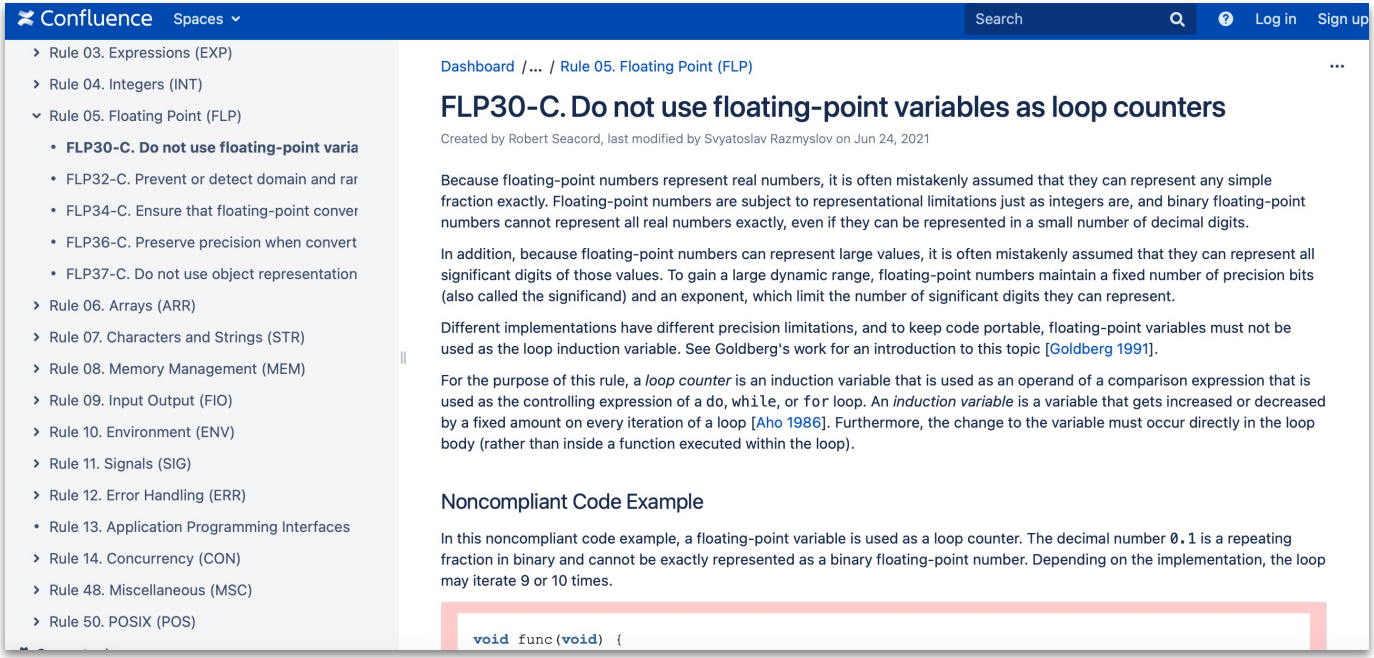
← MISRA C

MISRA C++ →

- A typedef name shall be a unique identifier
- The first operand of a conditional-operator shall have type bool
- Functions shall not be declared at block scope.

База.

Убираем потенциальные опасности (3) – SEI CERT C++



https://wiki.sei.cmu.edu/confluence/display/c/FLP30-C.+Do+not+use+floating-point+variables+as+loop+counters

База.

Убираем потенциальные опасности (4) - SEI CERT C++

Tool	Version	Checker	Description
Astrée	20.10	**for-loop-float**	Fully checked
Axivion Bauhaus Suite	7.2.0	**CertC-FLP30**	Fully implemented
Clang	3.9	`cert-flp30-c`	Checked by `clang-tidy`
CodeSonar	6.1p0	**LANG.STRUCT.LOOP.FPC**	Float-typed loop counter
Compass/ROSE			
Coverity	2017.07	**MISRA C 2004 Rule 13.4** **MISRA C 2012 Rule 14.1**	Implemented
ECLAIR	1.2	**CC2.FLP30**	Fully implemented
Helix QAC	2021.2	**C3339, C3340, C3342** **C++4234**	
Klocwork	2021.1	**MISRA.FOR.COND.FLT** **MISRA.FOR.COUNTER.FLT**	
LDRA tool suite	9.7.1	**39 S**	Fully implemented
Parasoft C/C++test	2021.1	**CERT_C-FLP30-a**	Do not use floating point variables as loop counters
PC-lint Plus	1.4	**9009**	Fully supported
Polyspace Bug	R2021a	CERT C: Rule FLP30-C	Checks for use of float variable as loop counter (rule fully

https://wiki.sei.cmu.edu/confluence/display/c/FLP30-C.+Do+not+use+floating-point+variables+as+loop+counters

База.

Keep it simple stupid

ISO26262:6-6-1h No hidden data flow or control flow

База.

Защитное программирование (1)

ISO26262:6-1-1d Use of defensive techniques

E.g. verify the divisor before the operation of division; check an identifier passed as parameter to verify that the caller is the intended caller; Use the “default” in switch cases to detect an error.

База.

Защитное программирование (2)

ОБНУЛЯЙТЕ
ПЕРЕМЕННЫЕ ПЕРЕД
ПЕРВЫМ ИСПОЛЬЗОВАНИЕМ

- Обрабатывайте ошибки (и ставьте в функциях [[nodiscard]] в C++ 17)
- Используйте контракты C++ 20
- Assertive programming
- Don’t Repeat Yourself

База.

Защитное программирование (3)

```
tt = new T[n];
```

```
If (n > 0 && n > (size_t(-1) - 8) / sizeof(T))
  throw std::length_error(“Too large array”);
tt = new T[n];
```

База.

Защитное программирование (4)

Move security-critical code out of templated code

so that it can be patched in a central place if necessary

the need for ABI consistency for the sake of security patches

RedHat defensive coding guide

База.

Защитное программирование – где найти

Гайды

RedHat defensive guide;
Qt Coding Conventions;
Google Code Style C++

Стандарты

High Integrity C++ Coding standard;
SEI CERT C++;
AUTOSAR C++ 14;
ISO/IEC 9899:2011

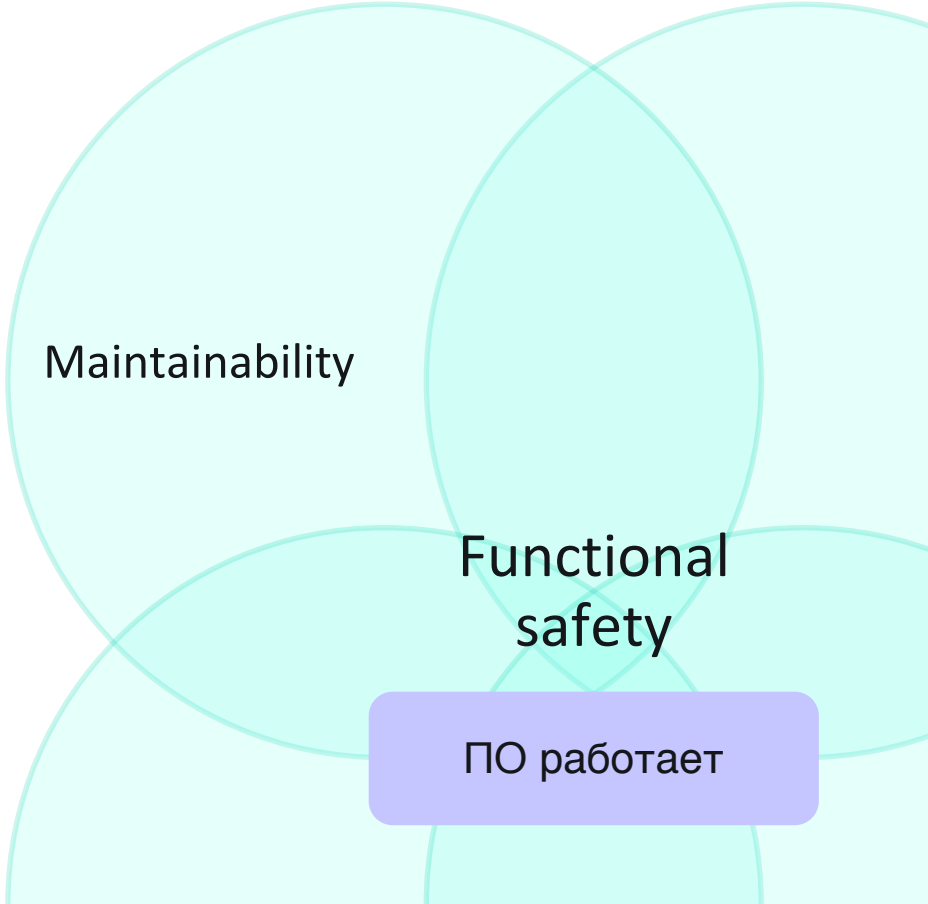
Утилиты

PVS Studio;
Helix QAC;
CodeSonar

База.

Защитное программирование и Secure Coding

Легкость нахождения и устранения дефектов



Базовая функциональная безопасность

Продвинутый уровень

Уход в минус

Продвинутый. Error detection and handling

DETECT

Check for errors of 3rd parties: libraries, hardware, network connectivity, etc

Range checks for input data

Self-test/plausible checks

Detection of data errors

Access violation control mechanisms

Monitoring of availability/heart-beat

RECOVER

Deactivation in order to achieve and maintain a safe state

Static recovery mechanism (e.g. recovery blocks, backward recovery, forward recovery and recovery through repetition);

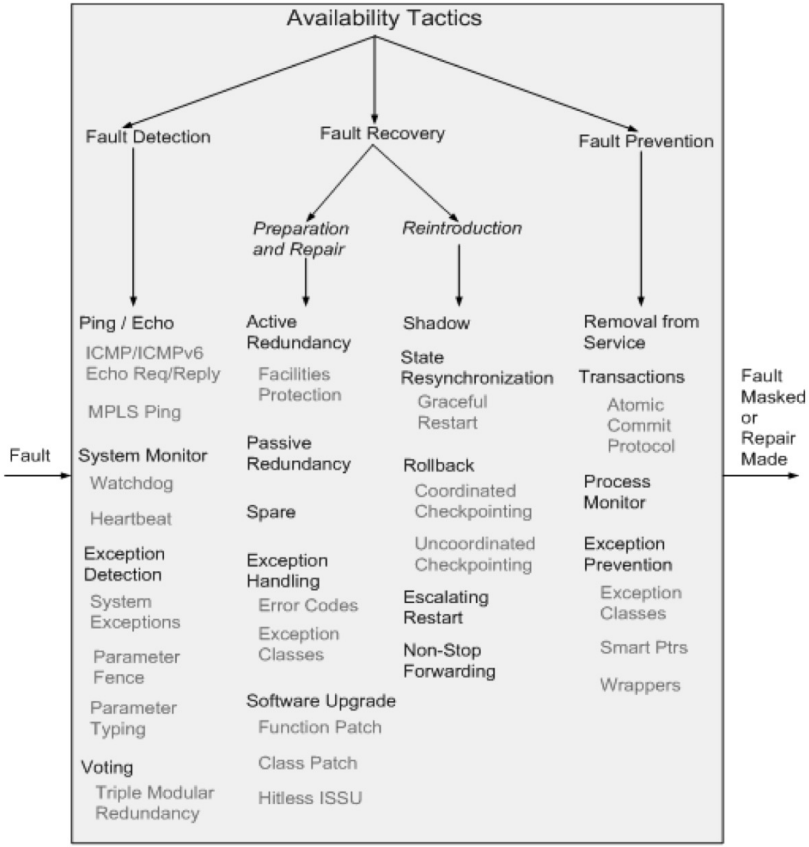
Graceful degradation

Redundancy (diverse and homogenous)

Correct codes for data

Restart

Error handling by ISO26262 vs SEI availability



https://resources.sei.cmu.edu/asset_files/TechnicalReport/2009_005_001_15101.pdf

Продвинутый.

Enforcement of low complexity

ISO26262:6-1-1a: “Enforcement of low complexity”

ISO26262:6-3-1b: “Restricted size and complexity of SW components”

Indicators for high complexity can be:

- highly branched control or data flow;
- excessive number of requirements allocated to single design elements;
- excessive number of interfaces of one design element or interactions between design elements;
- complex types or excessive number of parameters;
- excessive number of global variables;
- difficulty in providing evidence for suitability and completeness of error detection and handling;

Продвинутый.

Metrics of complexity

Halstead's volume

количество уникальных операторов\операндов и общее число операторов\операндов

Цикломатическая сложность (McCabe's Number)

Вершины и ребра графа

Maintainability index

зависит от предыдущих показателей

MI = 171 - 5.2 * log2(V) - 0.23 * (G) - 16.2 * log2 (LOC) + 50 * sin (sqrt(2.4 * CM)) (SEI)

MI = MAX(0,(171 - 5.2 * ln(V) - 0.23 * (G) - 16.2 * ln(LOC))*100 / 171) (MS VS),

где G = цикломатическая сложность, LOC = строки кода, CM = процент строк комментариев, V = Halstead Volume

Продвинутый.

Coupling and cohesion

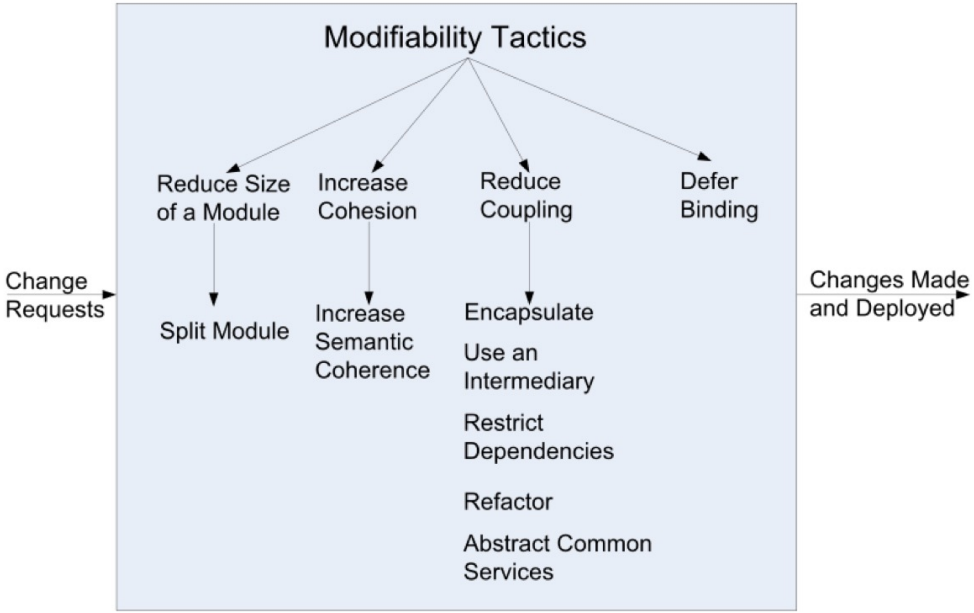
ISO26262:6-3-1d: “Strong cohesion within each SW components”

ISO26262:6-3-1e: “Loose coupling between SW components”

Low coupling and high cohesion - абстрактные слова или практические рекомендации с измеримым эффектом?

Продвинутый.

Coupling и cohesion (2)



© SEI CMU

Single responsibility principle:

"Gather together the things that change for the same reasons. Separate those things that change for different reasons." ©Robert Martin

S in SOLID

Продвинутый.

Coupling и cohesion. Metrics

Возможные индикаторы низкой согласованности:

- сложно сформулировать что делает метод
- сложно придумать название для метода

Продвинутый.

Coupling и cohesion. Metrics (2)

Fenton and Melton metric as a measure of coupling between two components x and y :
C(x,y) = i + n/(n+1)
where n=number of interconnections between x and y, and i= level of highest (worst) coupling type found between x and y .

Copling Type	Coupling Level	Modified Definition between components x and y
Content	5	Component x refers to the internals of component y, i.e., it changes data or alters a statement in y.
Common	4	Component x passes a control parameter to y.
Stamp	3	Component x passes a record type variable as a parameter to y.
Data	2	Component x and y communicate by parameters, each of which is either a single data item or a homogenous structure that does not incorporate a control element.
No Coupling	1	Component x and y have no communication, i.e., are totally independent.

Продвинутый.

БЫЛО:

	MI	Coupling	LOC
Personnel	73	271	5632
Personnel. General	71	50	428
Personnel. Report	68	93	365
Personnel. Payroll	73	79	1835
listUpdate	49	15	24
takeFormaData	40	21	46
dataSave	49	15	24

СТАЛО (после SOLID):

	MI	Coupling	LOC
Personnel	79	259	5629
Personnel. General	73	49	430
Personnel. Report	68	92	367
Personnel. Payroll	75	76	1833
listUpdate	51	14	23
takeFormaData	43	16	33
dataSave	52	14	19

MI =
Maintainability Index;

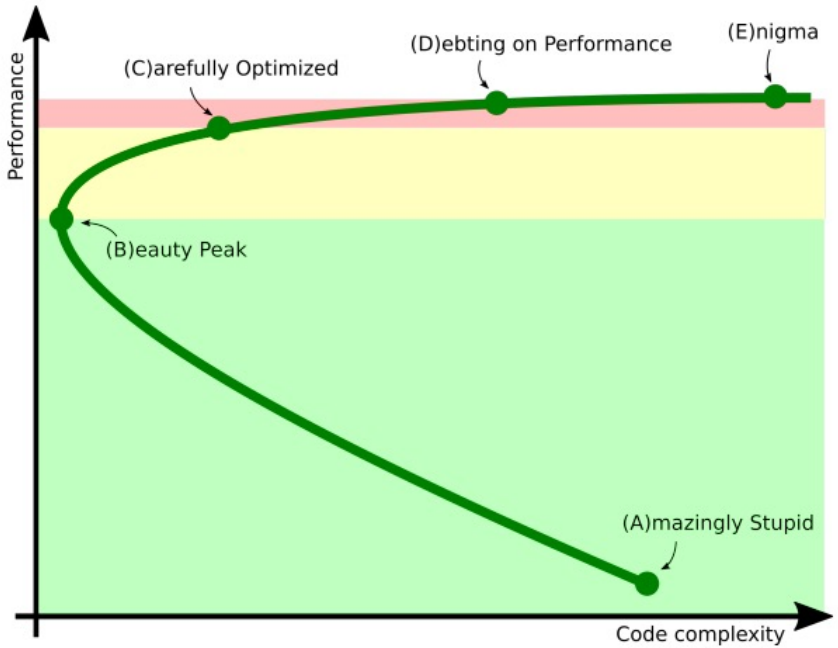
O.Turan, O.O. Tanriover «An Experimental Evaluation of the Effect of SOLID Principles to Microsoft VS Code Metrics»

Базовая функциональная безопасность

Продвинутый уровень

Уход в минус

Иногда чересчур хорошо - это плохо



Slide 8/66. «Keynote: Performance», Aleksey Shipilëv, 2017, D:20170404035127+02'00'

(c) Алексей Шипилев

Со звездочкой

Формальные методы

ISO26262:6-7-1d: “Semi-formal verification”
ISO26262:6-7-1e, ISO26262:6-4-1e : “Formal verification”

- TLA+
- Alloy
- Или вообще SPARK

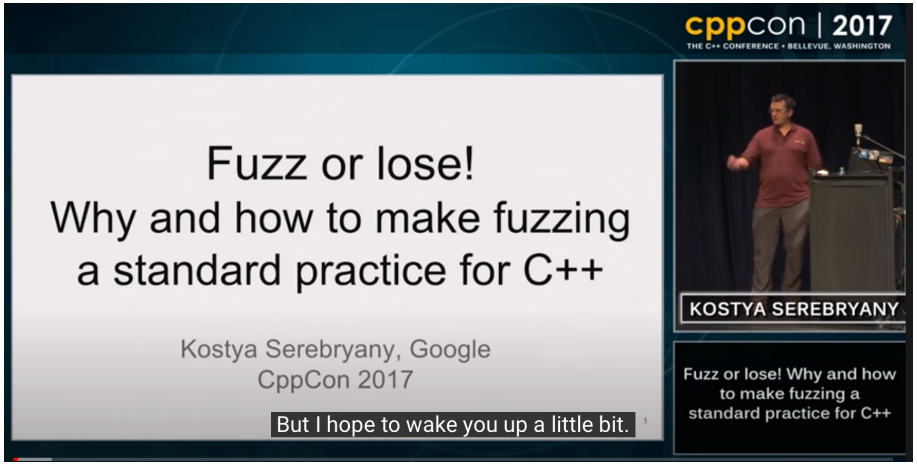
```
---- MODULE Transfer ----
EXTENDS Naturals, TLC

(* --algorithm transfer
variables alice_account = 10, bob_account = 10, money \in 1..20,
          account_total = alice_account + bob_account;

begin
Transfer:
  if alice_account >= money then
    A: alice_account := alice_account - money;
       bob_account := bob_account + money; \* Both now part of A
end if;
C: assert alice_account >= 0;

end algorithm *)

MoneyNotNegative == money >= 0
MoneyInvariant == alice_account + bob_account = account_total

====
```

Со звездочкой

Fuzzing

ISO26262:6-3-1b: “Restricted size and complexity of SW components”

- Libfuzzer
- American Fuzzing Loop (AFL++)
- Syzkaller
- Trinity



С минусом.

One-exit-one-entry

ISO26262:6-6-1a: “One entry and one exit point in subprograms and functions”

С минусом: One-exit-one-entry


```
Future<float> read_stream(std::istream& in)
{
    int count{};
    uint8_t byte;
    while (in >> byte) {
        data = data << 8 | byte;
        if (++count == 4) {
            co_yield *reinterpret_cast<float*>(&data);
            data = 0;
            count = 0;
        }
    }
}
int main() {
    auto raw_data = read_stream(std::cin);
    while (auto next = raw_data.next()) {
        std::cout << *next << std::endl;
   }
   return 0;
}
```

https://blog.feabhas.com/2021/09/c20-coroutines/

С минусом.

No-dynamic-objects

ISO26262:6-6-1b “No dynamic objects or variables, or else online test during their creation"

С минусом.

No-recursion

ISO26262:6-6-1j: “No recursion”

«Итерация от человека,
рекурсия от Бога»

(c) Питер Дойч

Компромисс

ISO26262:6: “An appropriate compromise .. can be necessary since the principles are not mutually exclusive.”

ISO26262:6: “_restricted_ means to minimize in balance with other design considerations.”

ВЫВОДЫ

1. Функциональная безопасность - интересный ракурс чтобы посмотреть на разработку
2. В функциональной безопасности есть наборы юного безопасника (aka стандарты), которые все объясняют
3. Функциональная безопасность - это не только код, но архитектура и процессы
4. Стандартные линтеры помогут проверить код на лучшие практики
5. Стандарты по функциональной безопасности достаточно разумны и призывают к поиску компромиссов

Спасибо!

KasperskyOS

Ссылки на почитать

https://wiki.sei.cmu.edu/confluence/pages/viewpage.action?pageId=88046682

https://developers.redhat.com/articles/defensive-coding-guide

https://resources.sei.cmu.edu/asset_files/TechnicalReport/2009_005_001_15101.pdf

https://www.autosar.org/fileadmin/user_upload/standards/adaptive/17-03/AUTOSAR_RS_CPP14Guidelines.pdf

https://www.perforce.com/resources/qac/high-integrity-cpp-coding-standard

https://google.github.io/styleguide/cppguide.html

https://habr.com/ru/company/yandex/blog/471012/

Книги
на почитать