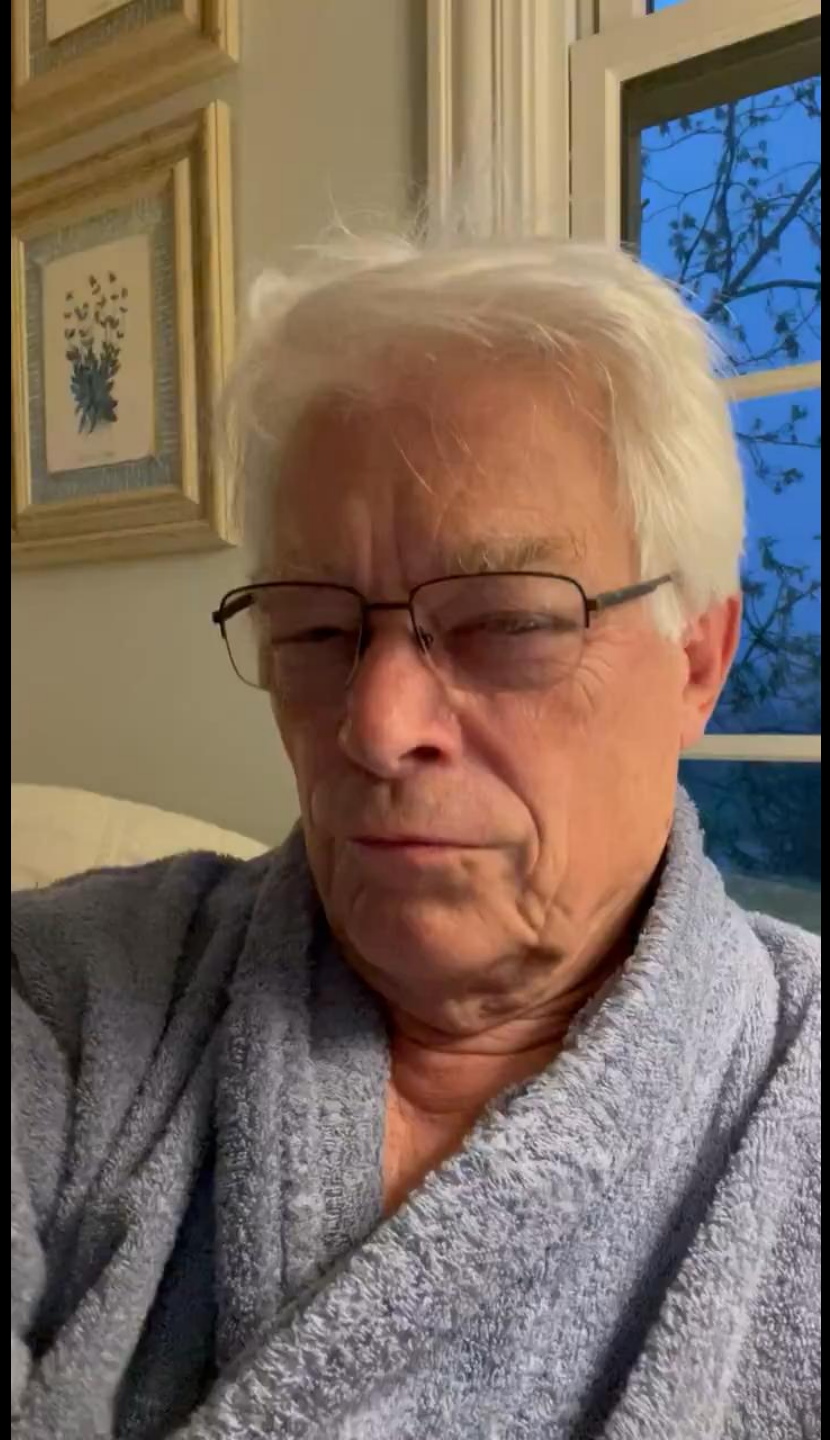
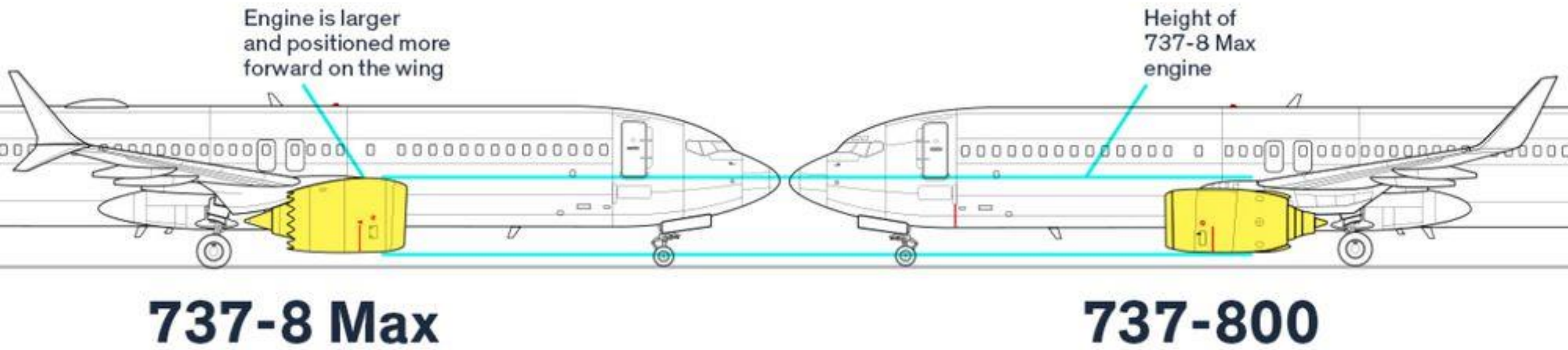
Legacy i refaktoryzacja kodu

Grzegorz Aksamit

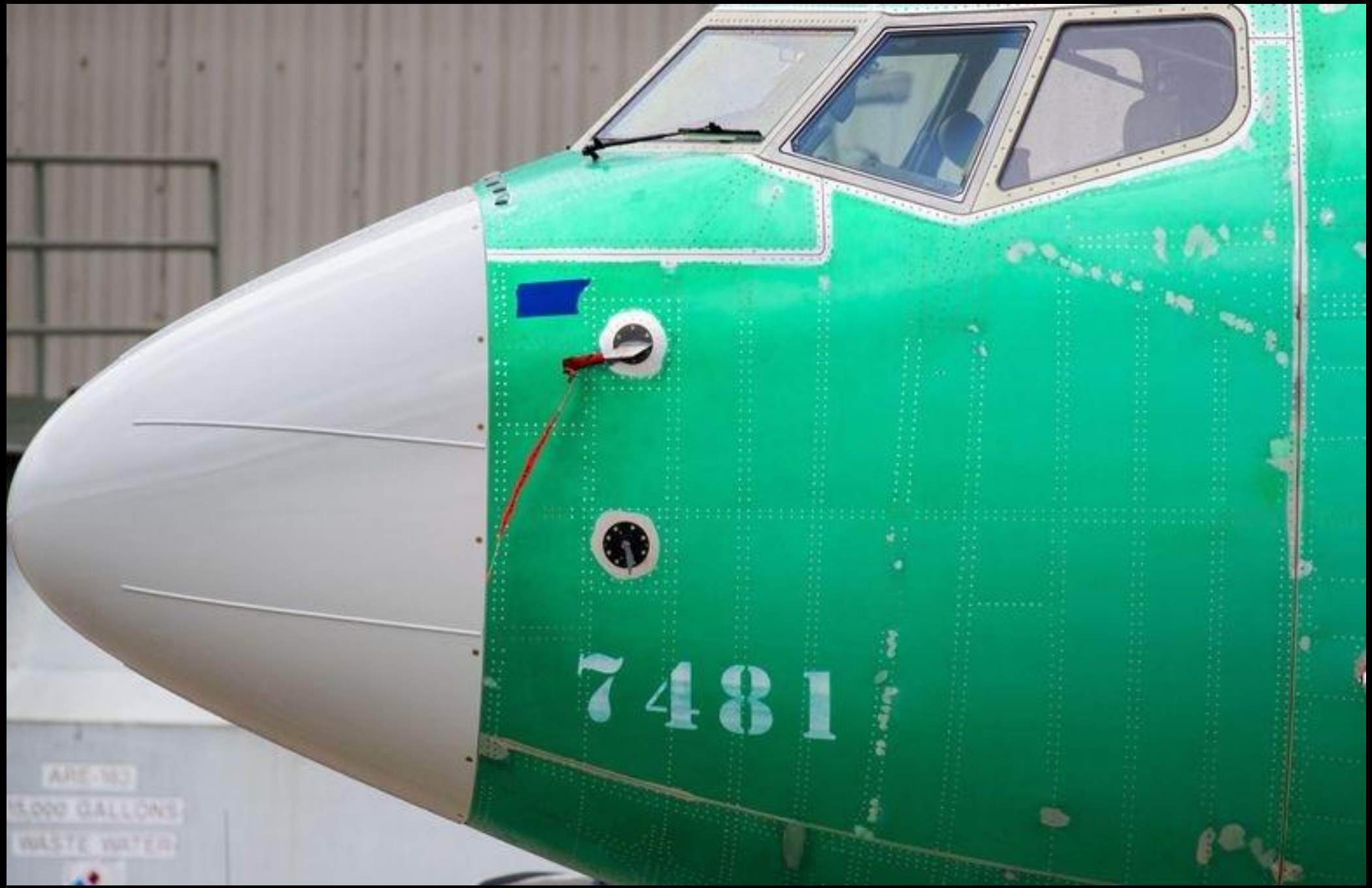


Uncle Bob
(Robert C. Martin)

▶ Zobacz film



https://spectrum.ieee.org/how-the-boeing-737-max-disaster-looks-to-a-software-developer


7481
WASTE WATER

“Boeing, more than 20 years ago, probably got a little too far ahead of itself on the topic of outsourcing”
- Brian West, Boeing CFO

“Did it go too far? Yeah, probably did.”
- Dave Calhoun, Boeing CEO

Jaki język programowania powstał w 1959 roku?

Branża finansowa COBOLem stoi

80%

Of in-person transactions use COBOL

95%

Of card swipes use COBOL

800 billion

Lines of COBOL in use today

112,500

Different COBOL apps run in one NYC bank

https://www.reuters.com/article/technology/banks-scramble-to-fix-old-systems-as-it-cowboys-ride-into-sunset-idUSKBN17C0CN/
https://www.wealthsimple.com/en-ca/magazine/cobol-controls-your-money

WEEKLY WORLD NEWS

THE COMPUTER CRASH OF THE MILLENNIUM!

JANUARY 1, 2000

IS THIS THE END?

ALL BANKS WILL FAIL!

FOOD SUPPLIES WILL BE DEPLETED!

THE STOCK MARKET WILL CRASH!

ELECTRICITY WILL BE CUT OFF!

VEHICLES USING COMPUTER CHIPS WILL STOP DEAD!

TELEPHONES WILL CEASE TO FUNCTION!

DEVASTATING WORLDWIDE DEPRESSION!

THE DAY THE EARTH STANDS STILL!

WEEKLY WORLD NEWS

THE FINAL DAYS

DOOMSDAY COUNTDOWN

ALSO INSIDE...

ARMAGEDDON

YEAR 2000 COMPUTER BUG will turn machine against man!

Hundreds of planes will fall out of the sky!

Cars will stop dead in their tracks!

Nuclear missiles will launch themselves!


COBOL COWBOYS
Professionals for Legacy COBOL Systems
Home
Meet the Company
Services Provided
Environments
COBOL Today
New to COBOL?
Contact
NOT OUR FIRST RODEO
MEET THE COMPANY

"Some of the software I wrote for banks in the 1970s is still being used”
- Bill Hinshaw, founder COBOL Cowboys


BILLETS


(CON) CSC-W32 starting ...
POTWIERDZENIE
KARTA
WINCOR NIXDORF
WPŁATA | WYPŁATA BANKNOTÓW
100
włóż banknoty dłuższym bokiem
CANCEL PORZUĆ
CLEAR POPRAW
ENTER POTWIERDŹ

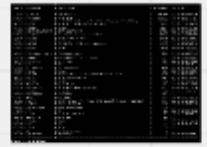
```
-rw-r--r--  1 gaks  staff   1,6G 19 maj  2024 src-backup-20240519.tar.bz2
```

SERWER

BAZA DANYCH

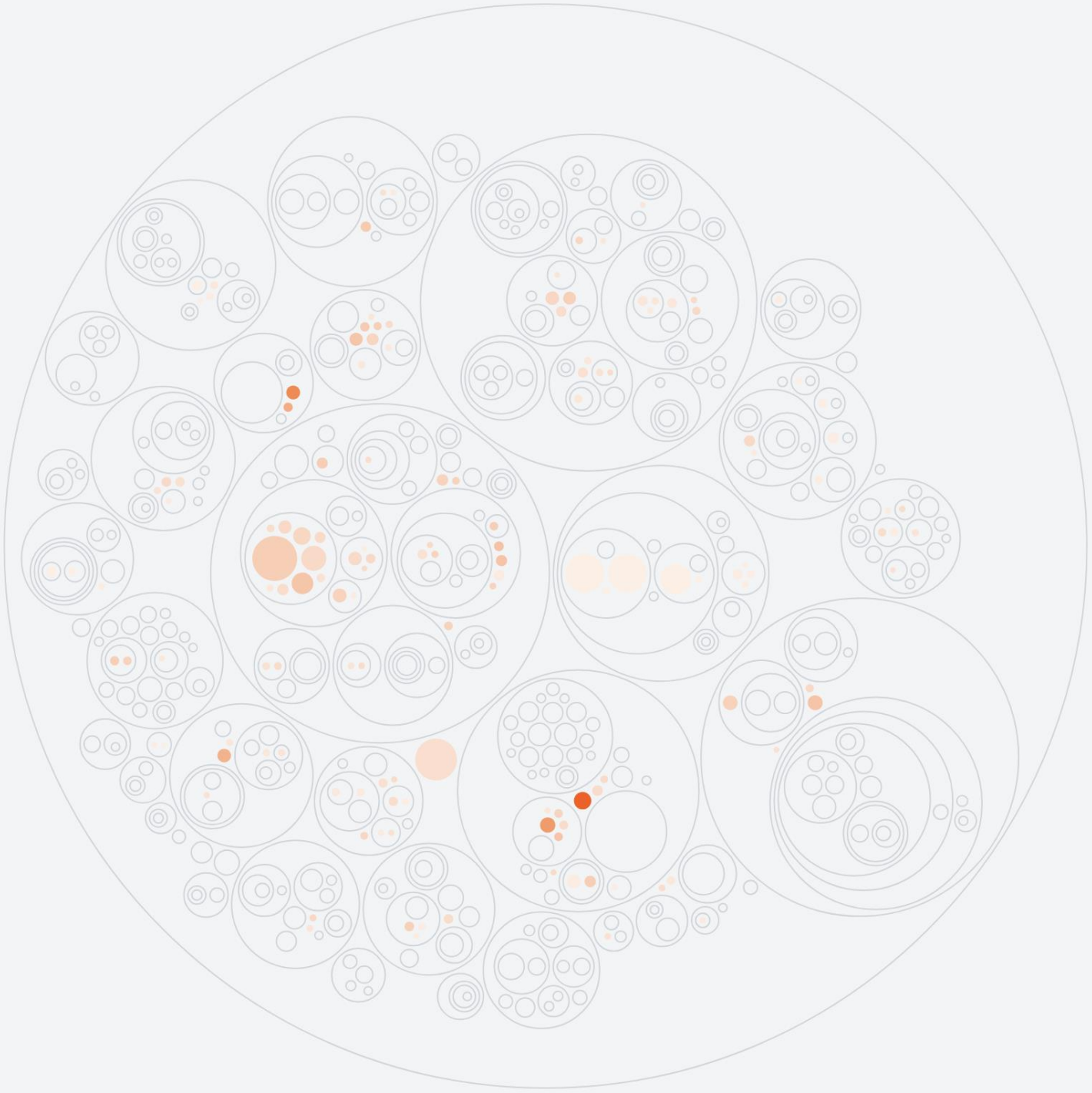
KOLEKTOR

KOD



FAKTUROWANIE

„Dużo więcej czasu spędzamy CZYTAJĄC kod niż go pisząc”
- Grzegorzeo Aksamitehlo



Jak wybrać cel refaktora

Dobry moment, żeby zacząć to DZISIAJ

Zacznij od najmniejszych i najprostszych rzeczy

Zmień głupią nazwę na niegłupią
Rozbij wielką metodę na kilka mniejszych
Usuń niepotrzebny kod
Dopisz komentarz dla future-self
Dopisz jeden unit-test
...

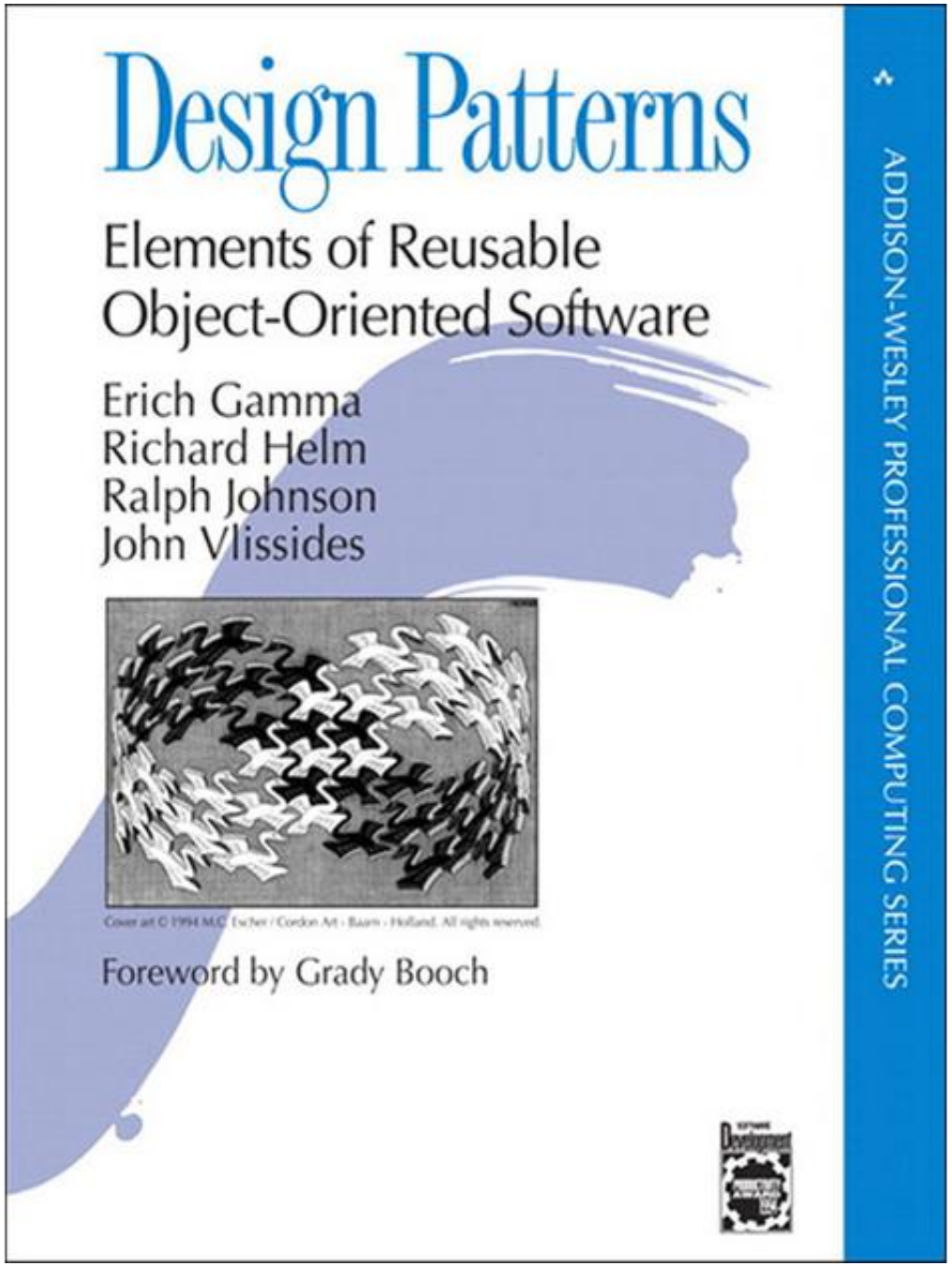
Stosuj metodę „skauta”

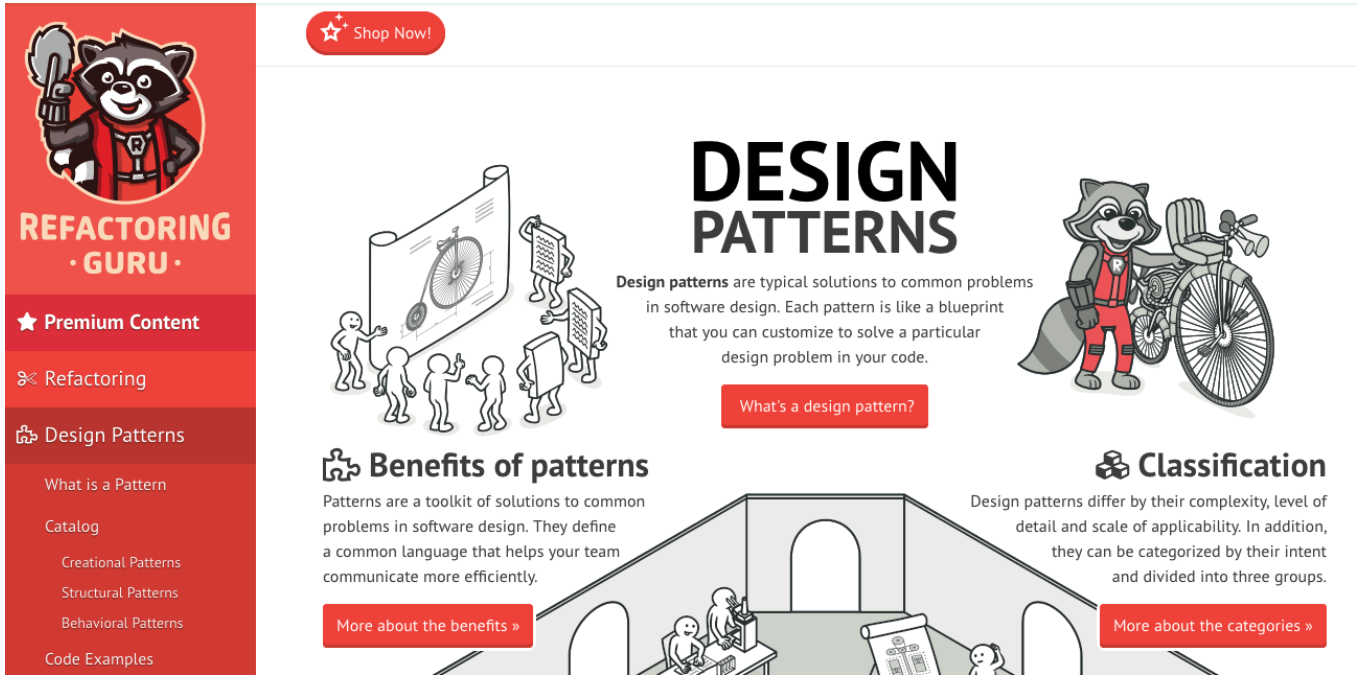
zostaw wokół siebie większy porządek niż zastałeś

Uruchamiaj unit-testy wielokrotnie i często.
Po każdej najdrobniejszej zmianie.

„The key to effective refactoring is recognizing that you go faster when you take tiny steps”
- Martin Fowler

Wzorce projektowe (design patterns)





https://refactoring.guru

Uncle Bob Martin
@unclebobmartin

Design Patterns don't slow anybody down. Being stupid does.

9:35 PM · May 6, 2025 · **197.3K** Views

84 159 1.6K 126

Feature Toggles (Feature Flag)

1 usage Grzegorz Aksamit

```python
def __assign_task_statuses(self, now: datetime = None):
    if CityConfig.get_solo().use_new_task_status_management:
        return self.__assign_task_statuses_new(now)
    else:
        return self.__assign_task_statuses_legacy(now)
```

1 usage Grzegorz Aksamit

```python
def __assign_task_statuses_new(self, now: datetime = None):...
```

1 usage Michał Klimek +2

```python
def __assign_task_statuses_legacy(self, now):...
```

```
if CityConfig.get_solo().use_pickup_estimate_walking_plus_waiting_per_order is True:
    walking_characteristics = characteristics.get_pickup_walking_characteristics()
    walking_time = walking_characteristics.get_walking_time_for_venue(ft.order.venue)
else:
    walking_time = datetime.timedelta(minutes=2)
```

Circuit Breaker

```python
def _cb_factory(name=None):
    return pybreaker.CircuitBreaker(
        fail_max=3,
        reset_timeout=120,
        name=name,
        state_storage=CircuitBreakerSettings.storage(namespace="route"),
        listeners=[RouteBreakerListener()],
        exclude=[lambda e: type(e) == HTTPError and e.status_code < 500]  # error codes 500 up are broken service
    )


path_breaker = _cb_factory("routing-path")
route_breaker = _cb_factory("routing-route")
isochrone_breaker = _cb_factory("routing-isochrone")

@isochrone_breaker
def isochrone(
    self,
    center: Point,
    profile: RoutingProfile,
    timeout: float = TIMEOUT_ISOCHRONE,
    *,
    time_limit: Optional[float] = None,
    distance_limit: Optional[float] = None,
) -> Optional[Polygon]:
```

Dependency Inversion Principle

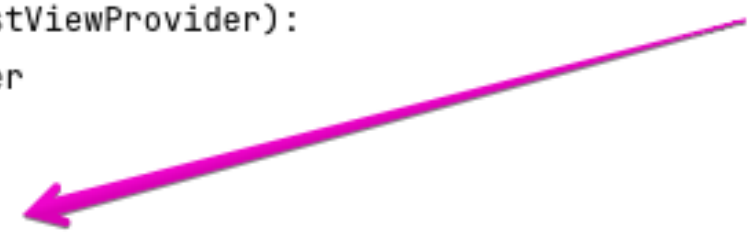
```python
class EmailService:
    def send_email(self, message):
        print(f"Sending email: {message}")

class ReportGenerator:
    def generate_report(self):
        return "Generated Report"

    def send_report(self):
        email_service = EmailService()  # DEPENDENCY do EmailService
        email_service.send_email(self.generate_report())
```

Dependency Inversion Principle

```python
from abc import ABC, abstractmethod

class NotificationService(ABC):
    @abstractmethod
    def send(self, message):
        pass

class EmailService(NotificationService):
    def send(self, message):
        print(f"Sending email: {message}")

class ReportGenerator:
    def __init__(self, notification_service: NotificationService):
        self.notification_service = notification_service

    def generate_report(self):
        return "Generated Report"

    def send_report(self):
        self.notification_service.send(self.generate_report())
```

Dependency Injection

```
class ProductionCourierApiRequestViewProvider(CourierApiRequestViewProvider):
    view_options_provider: CourierApiRequestViewOptionsProvider
    service_locator: ServiceLocator

    def __init__(self, view_options_provider: CourierApiRequestViewOptionsProvider, service_locator: ServiceLocator):
        self.view_options_provider = view_options_provider
        self.service_locator = service_locator
```



Dependency Injection

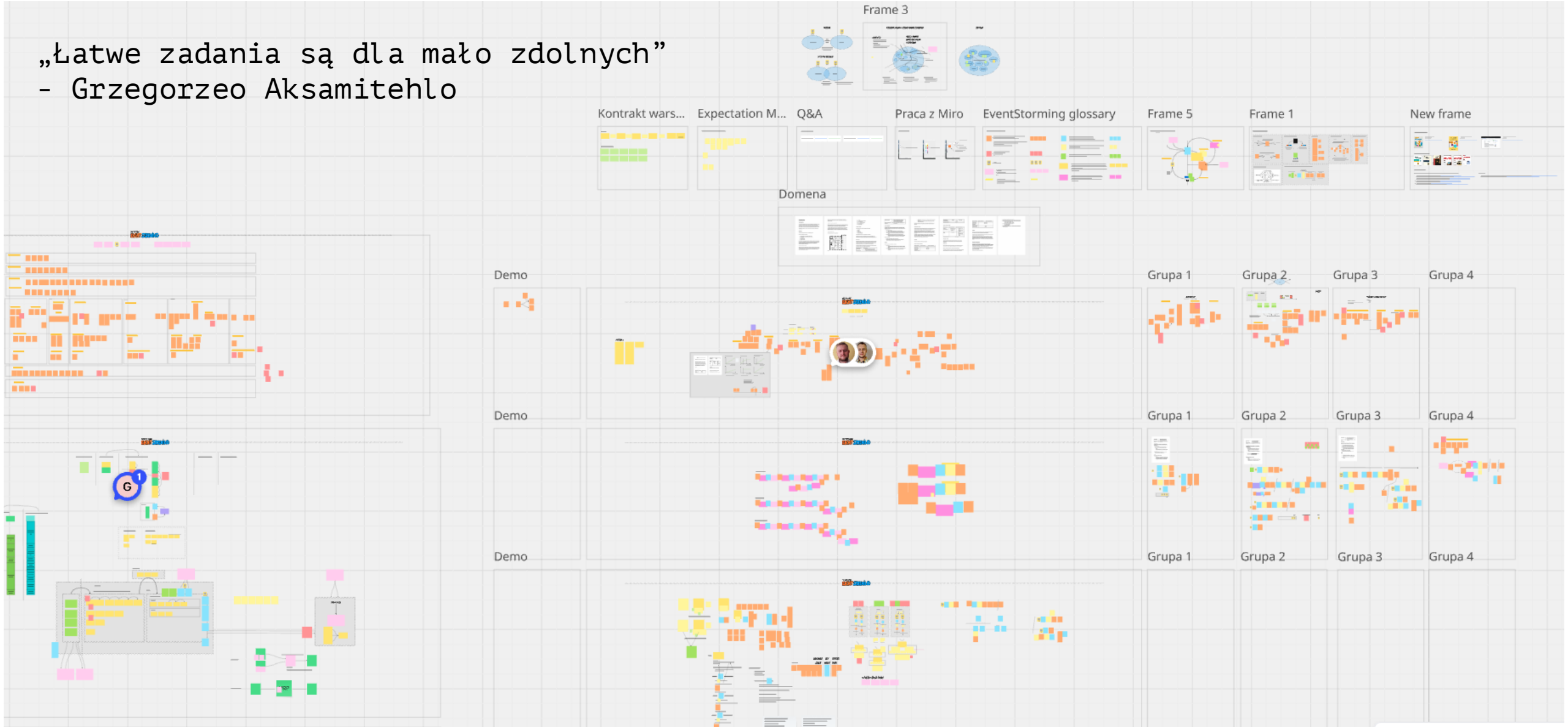
```python
class RequestViewOptions(ABC):
    ...


class CourierApiRequestViewOptionsProvider(ABC):
    @abstractmethod
    def get_request_options(self, context: CourierApiViewContext) -> RequestViewOptions:
        raise NotImplementedError()


class ProductionRequestViewOptionsProvider(CourierApiRequestViewOptionsProvider):
    def get_request_options(self, context: CourierApiViewContext) -> RequestViewOptions:
        return ProductionRequestViewOptions()


class MockRequestViewOptionsProvider(CourierApiRequestViewOptionsProvider):
    def get_request_options(self, context: CourierApiViewContext) -> RequestViewOptions:
        return MockOptions()
```

Dependency Injection

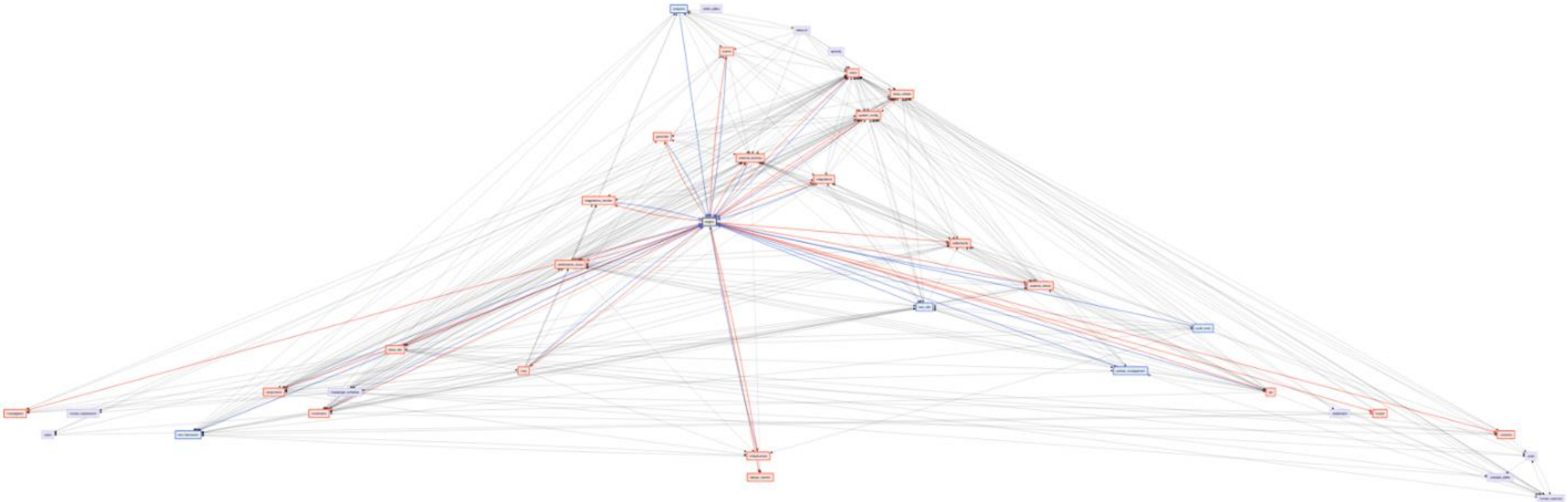
```python
from django_injector import Module, singleton, provider

class CourierApiViewsModule(Module):
    @singleton
    @provider
    def courier_view_options_provider(self) -> CourierApiRequestViewOptionsProvider:
        return ProductionRequestViewOptionsProvider()


class MockCourierApiViewsModule(Module):
    @singleton
    @provider
    def request_view_options_provider(self) -> CourierApiRequestViewOptionsProvider:
        return MockRequestViewOptionsProvider()
```

Co przed nami?

„Łatwe zadania są dla mało zdolnych”
- Grzegorzeo Aksamitehlo



Co przed nami?



DDD - Domain Driven Design
Event Storming
SOLID principles
CQRS
Ports & Adapters architecture
Hexagonal architecture
Repository pattern
Anti-corruption layer
Modular monolith
Code as a crime scene - Adam Tornhill

Martin Fowler
Robert C. Martin (Uncle Bob)
Alberto Brandolini
Mariusz Gil
Sławomir Sobótka
Jakub Pilimon
Łukasz Szydło