Synergy of Organizational Patterns and Aspect-Oriented Programming

Peter Berta and Valentino Vranić

Institute of Informatics, Information Systems, and Software Engineering

vranic@stuba.sk  fiit.sk/~vranic
```java
interface Subject {
    attach(observer: Observer)
    detach(observer: Observer)
    notify()
}

interface Observer {
    update()
}

class Subject1 {
    state
    +getState()
    +setState()
}

class Observer1 {
    state
    +update()
}
```
Observer

observing objects should be notified of the change in the state of the subject of their observation,

But they should be attachable to the subject without having to modify it
The Scrum Team consists of a Product Owner, the Development Team, and a Scrum Master. Scrum Teams are self-organizing and cross-functional. Self-organizing teams choose how best to accomplish their work, rather than being directed by others outside the team. Cross-functional teams have all competencies needed to accomplish the work without depending on others not part of the team. The team model in Scrum is designed to optimize flexibility, creativity, and productivity. The Scrum Team has proven itself to be increasingly effective for all the earlier stated uses, and any complex work.

Developer Controls Process

People don't like being ordered what to do, but the work needs to be organized.

Make the developers as a team decide how to organize development.

The Scrum Team consists of a Product Owner, the Development Team, and a Scrum Master. Scrum Teams are self-organizing and cross-functional. Self-organizing teams choose how best to accomplish their work, rather than being directed by others outside the team. Cross-functional teams have all competencies
A conflict of contradicting forces

Developer Controls Process

People don't like being ordered what to do, *But* the work needs to be organized.

Make the developers as a team decide how to organize development.
The Scrum Team consists of a Product Owner, the Development Team, and a Scrum Master. Scrum Teams are self-organizing and cross-functional. Self-organizing teams choose how best to accomplish their work, rather than being directed by others outside the team. Cross-functional teams have all competencies needed to accomplish the work without depending on others not part of the team. The team model in Scrum is designed to optimize flexibility, creativity, and productivity. The Scrum Team has proven itself to be increasingly effective for all the earlier stated uses, and any complex work.

team. Cross-functional teams have all competencies needed to accomplish the work without depending on others not part of the team. The team model in Scrum is designed to optimize flexibility, creativity, and productivity. The Scrum Team has proven itself to be increasingly effective for all the earlier stated uses, and any complex work.


**Architect Also Implements**

Architects need to focus on the overall structure, *But* they should not lose contact with the development reality.

Let the (software) architect participate in actual programming.
Community of Trust

People are naturally cautious and suspicious, which is often enforced by rules and practices, but for to really do the work, they need to trust each other.

Those "in charge" should make obvious they trust others by giving up the watch-over activities and letting people decide about their own work. Good and sincere communication is essential to overcoming fear.
Scrum

Organizational patterns form pattern languages

**Architect Controls Product** – establishes an architect role

**Architect Also Implements** – elaborates on that role making it also implement (develop)

**Developing In Pairs** – further precises how the architect collaboration with developers may be realized
Organizational patterns form pattern languages

**Architect Controls Product** – establishes an architect role

**Architect Also Implements** – elaborates on that role making it also implement (develop)

**Developing In Pairs** – further precises how the architect collaboration with developers may be realized
Conway's Law

Any organization that designs a system (defined broadly) will produce a design whose structure is a copy of the organization's communication structure.

How organizational patterns correspond to particular programming paradigms?

Organizational Patterns in Aspect-Oriented Programming (AOP)
Conway's Law

There is a need for a system to follow a particular architecture

But its structure is constrained by the structure of the organization that produces the system

Adapt the organizational structure to the needs of the system architecture
Customer

- Place an Order
- Cancel an Order
Symmetric aspect-oriented modularization
UC Place an Order

Basic Flow: Place an Order

1. Customer selects to place an order.
2. UC Search Products is being activated.
3. Customer confirms the product selection and adjusts its quantity.
4. If the product is available, System includes it in the order.
5. Customer continues in ordering further products.
6. Customer chooses the payment method, enters the payment data, and confirms the order.
7. Customer can cancel ordering at any time.
8. The use case ends.

Extension points:

- Checking Product Availability: Step 4
UC Place an Order

Basic Flow: Place an Order

1. Customer selects to place an order.
2. UC Search Products is being activated.
3. Customer confirms the product selection and adjusts its quantity.
4. If the product is available, System includes it in the order.
5. Customer continues in ordering further products.
6. Customer chooses the payment method, enters the payment data, and confirms the order.
7. Customer can cancel ordering at any time.
8. The use case ends.

Extension points:
- Checking Product Availability: Step 4

UC Modify the Restock Plan

Alternate Flow: Modify the Restock Plan

After the Checking Product Availability extension point of the Place an Order use case:

1. System checks the available quantity of the product being ordered.
2. If the quantity is below the limit, System adds the quantity under demand to the restock plan.
3. The flow continues with the step that follows the triggering extension point.
public class Ordering {

    ...

    public void order() {

        ...

        new ProductSearch().search(product);

        ...

        if (productAvailable(product)) {
        ...
        }
        else...

    }

    ...

}
public class Ordering {
    ...
    public void order() {
        ...
        new ProductSearch().search(product);
        ...
        if (productAvailable(product)) {
            ...
        } else...
    }
    ...
}

public aspect RestockPlan {
    ...
    void around(Product product):
        call(* Ordering.productAvailable(..) && args(tovar) {;

        // increase the quantity in the restock plan
        ...
    }
    ...
}
AOP provides a means of decoupling cross-cutting concerns from other modules, thereby enhancing the flexibility and reusability of the organization's systems. This includes activities such as peer-to-peer (P2P) communication.

By creating group-level task assignments and Peer-to-Peer activities and enhancing AOP (peer-use and AOP)
directly may be easier with symmetric AOP (peer use cases)

AOP provides for a greater level of decoupling between the software modules developed by different parts of the organization, including reversing dependencies (use case extension)

By creating groups with similar
Dividing people into independent teams and assigning them software modules to be developed directly may be easier with symmetric AOP (peer use cases).

Few Roles

Divide and Conquer

AOP provides for a greater level of decoupling between the software...
Organizational Style pattern language

Form Follows Function

Shaping Circulation Realms

Hallway Chatter

By creating groups with similar activities and employing symmetric AOP (peer use cases), code becomes more modular and the development becomes easier.
pointcut calculations(): call(* Stock.calc*(..));

Much of pointcut fragility can be avoided by encouraging communication among developers.

Aspects are about obliviousness, but the developers or teams whose code is affected by the aspects others have developed may be oblivious of them, too.
Project Management pattern language

- Size the Schedule
  - Surrogate Customer
  - Work Split
  - Team per Task
    - Sacrifice One Person
> Conway's law: people <-> code

> Organizational patterns and aspect-oriented programming can support each other

> 9 organizational patterns examined more closely

> Organizational patterns referred by the examined ones are natural candidates for further examination

> Currently examining organizational patterns close to design patterns (People and Code pattern language)