COMAQA.BY

QA automation community

Mixing Visitor, Builder, Composite, Decorator and Iterator: building architecture on the cross-cutting example

Anton Semenchenko

About lecture ☺

Anton Semenchenko

Activist of COMAQA.BY and CoreHard.BY communities, co-founder of DPI.Solutions, manager at EPAM Systems. More than 16 years of experience in IT. Specializes in low-level development, QA automation, management, sales.

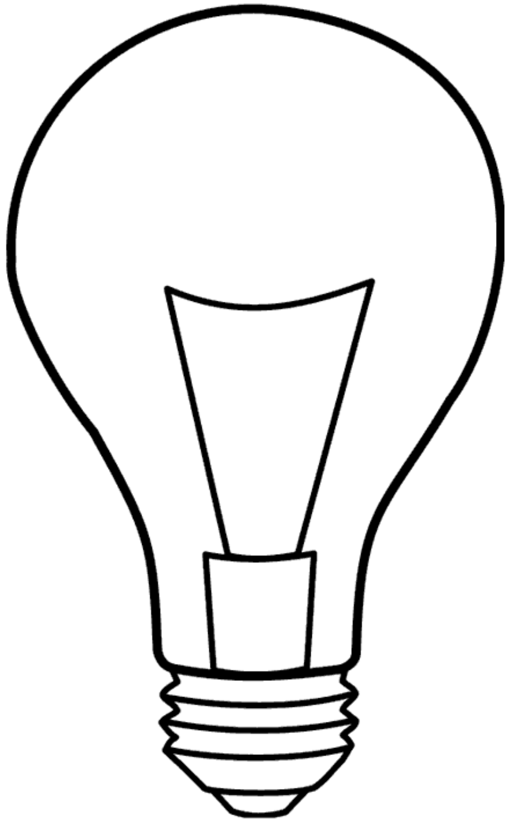
Agenda

1. Issue
2. Solution
3. Detailed context of the cross-cutting example
4. List of necessary DP’s
5. A way to link DP’s
6. Architecture example
7. Pros and Cons
8. Detailed summary
9. High level summary
10. Recommended literature

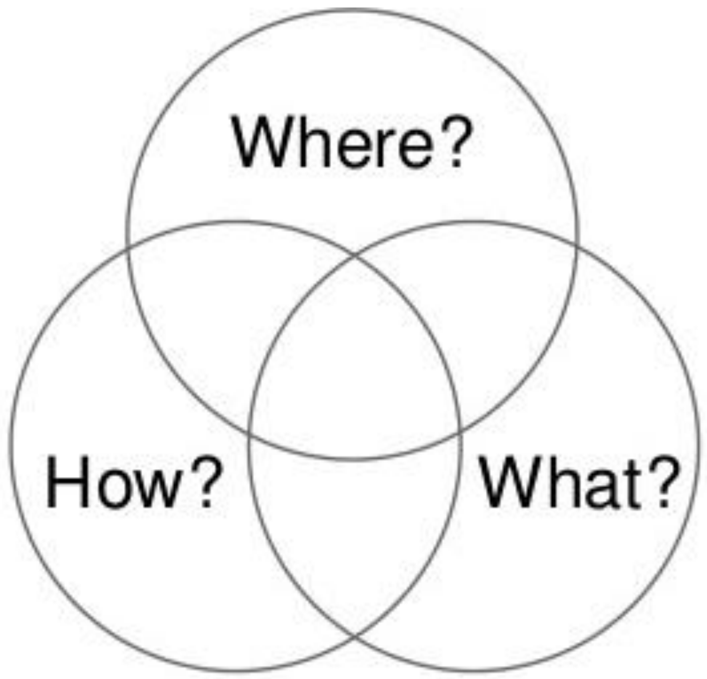
COMAQA.BY

Issue!

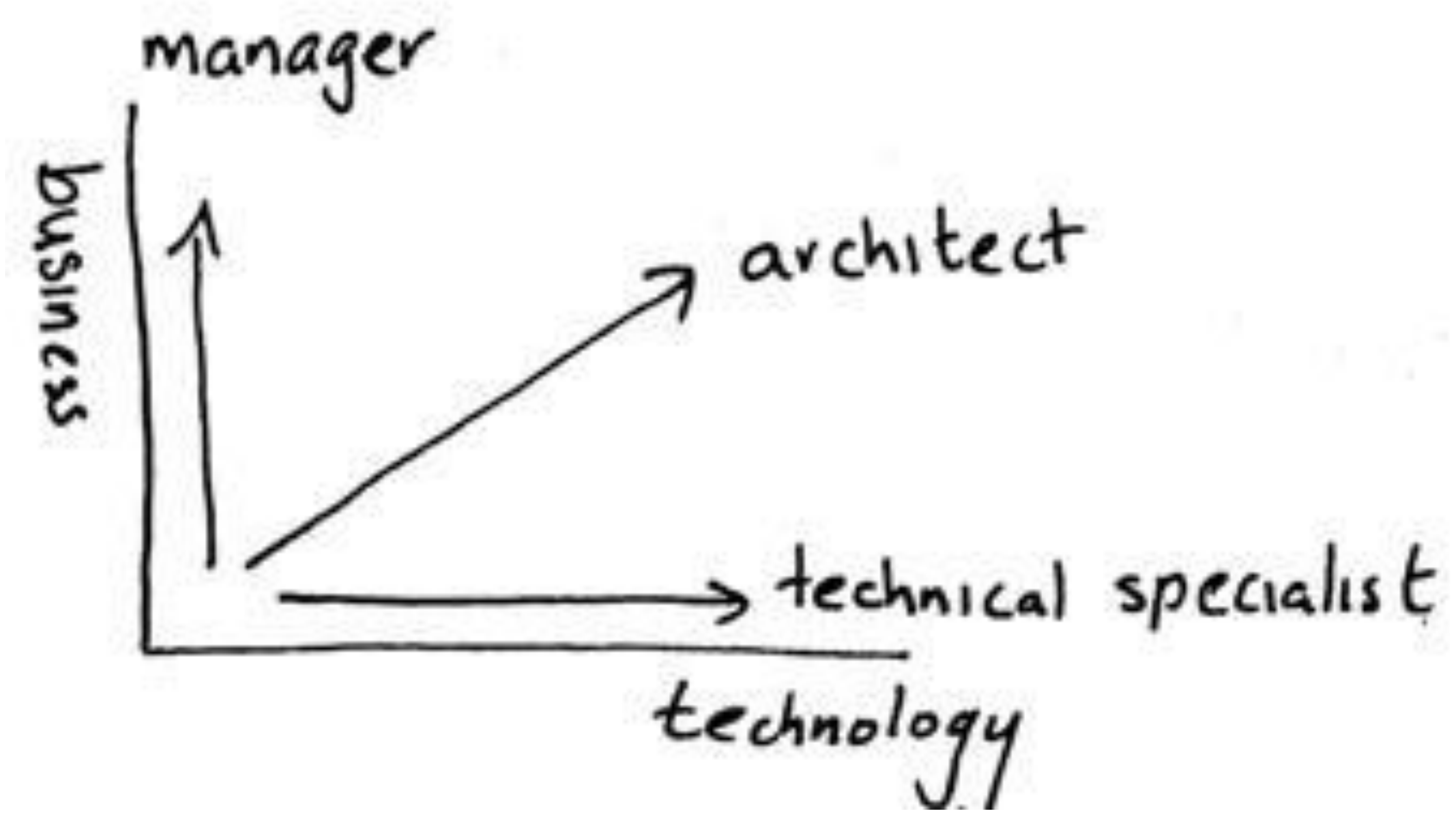
How to solve challenges?



Tech side related solution!

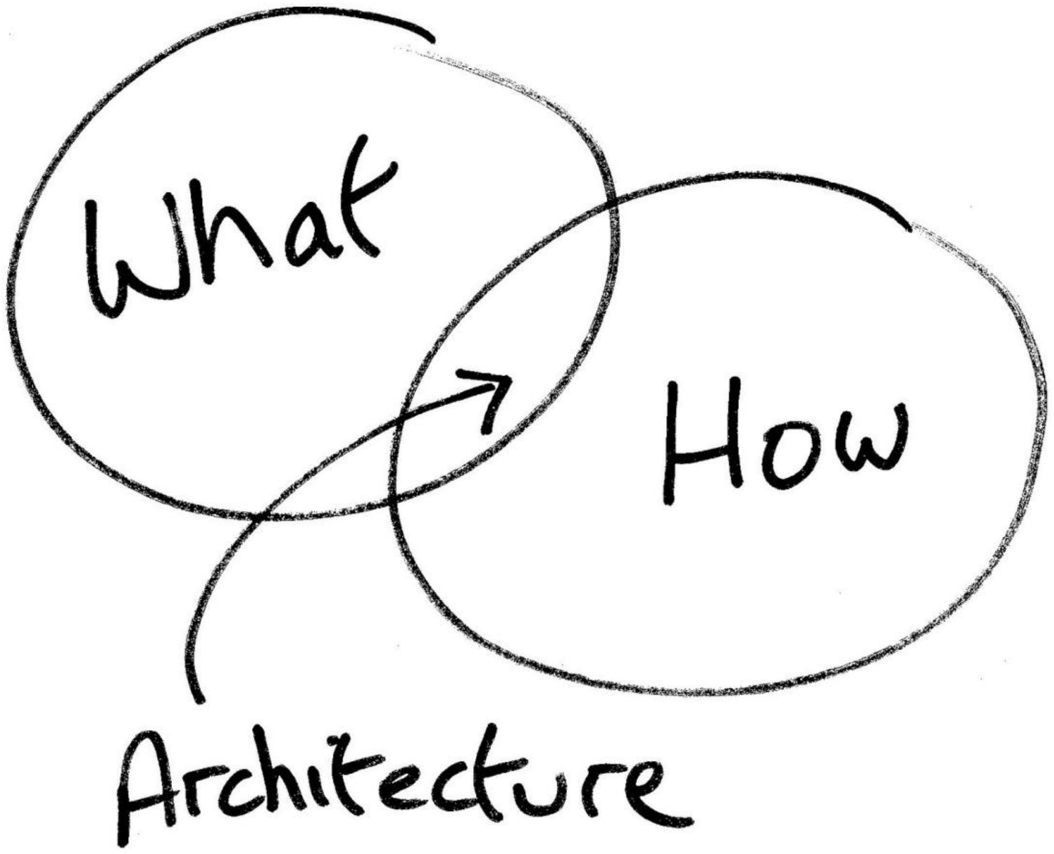


Process related solution!



COMAQA.BY

Linked DP based Architecture



COMAQA.BY

Cross-cutting example context

EXAMPLE

COMAQA.BY

Domain

Granular back-up and restore of Targets:

- MS SQL
- MS Exchange
- MS SharePoint
- file systems
- different variants of virtualization
- and many other things …

Task

Formulating the task:

- Fast
- Consistent
- “Supportable”
- Granular

Current solutions

Current “state”:

- A lot of **manual testing**
- **Not following coding standards** in a strict way
- **From time to time** formal **inspections**
- **Not full Unit-tests** coverage
- **Not using Design Patters systematically**
- Very **skillful C++ engineers**, including soviet physicists, scientists, and really clever people

Requirements: to improve

- Decrease the number of **bugs found by end users**
- **Guarantee the release time** of new features in predictable, and ideally, in a short time
- **Shorten** the **work \ time load** for testing of new versions

Additional limitations

- Release is scheduled not more than in 3 months after Release Back-up Target, constant release date changes
- *Scope is constantly changing*
- *“Low priority” features are almost not exist*
- *Architecture should be created in a way for making the process of adding new features or deleting them from the scope easy and convenient*
- Issues in the department of functional testing

Methodology

- **Iterational** process
- Non-Scrum but **most of the practices** are taken from that methodology
- **PRD** - Project Requirement Document
- **ERD** - Engineering Requirement Document
- **Prototyping**
- **Architecture \ Design Draft Phase**
- Then, **Scrum like iteration**

Ideal Scrum killed company

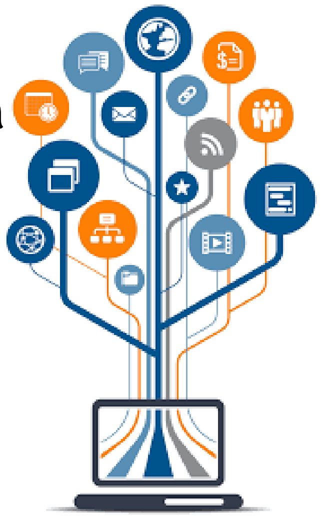
- a. Concept of “**Universal Soldier**”
- b. Blind following of **Scrum** methodology as the **dogma**
- c. **Results**
- d. Details – “it’s completely a different story” ☺

Tasks, part 1

Implement a plug-in for

- Effective
- Consistent
- Granular
- Persistent for update

back-up restore solution for new version of MS SQL Dena



Tasks, part 2

- A big number of features
- Features are **prioritized**
- Most of the **features are very important** for the end-user
- Number of features that are easy to avoid is almost zero
- Development process is build using **Release Candidate**
- Release of the product not more than in **3 months after Release Target**

Tasks, part 3

Complex configuration

- One physical machine one SQL instance
- One physical machine multiple SQL instances of one version
- One physical machine multiple SQL instances of different versions
- The same for virtual machines
- Work in the bounds of network with physical machines
- Work in the bounds of network with virtual macl
- Work in “mixed” network
- Variants of clustering of SQL

Tasks, part 4

A big number of special cases

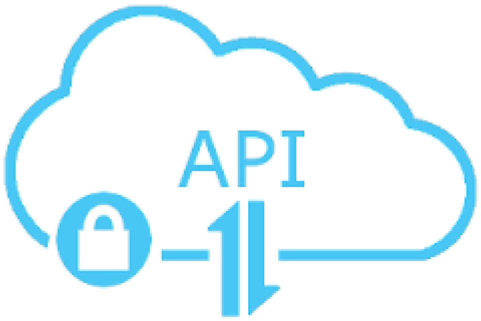
- DB or a separate file are renamed during use process
- DB or a separate file are moved during use process
- Variations of naming conflicts
- Restore to the different folder, with additional variants of naming conflicts
- Continuous operations at the DB
- Other special cases

Tasks, part 5

- Time for functional testing is not more than 3 months, in fact – approximately not more than one month

Tech details, part 1

- Using standardized mechanisms of performing back-up copying (VSS)
- Using C# SMO for ultra granular reserve back-up for some special cases.



Tech details, part 2

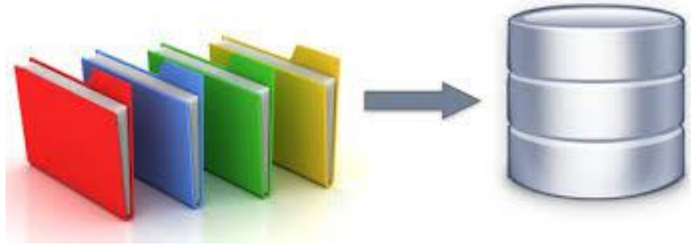
Mechanisms of optimization on VSS level

- System DB
- User DB
- 3 Recovery models
- *Simple*
- *Bulk logged*
- *Full*

Tech details, part 3

Mechanisms of server level optimization

- Storing data
- Restore data
- Smothering the edges between storing and restore speed



Tech details, part 4

- Supporting of **limited back-up window**
- **User-chosen subset** of DBs
- **Optimizing** the order of copying DB
- **Multiple checks**, including consistency checks
- **Reports** for users
- Different level of **report** specification
- **Tracing** for technical specialists
- Wide **range** of tracing specification
- Saving the concept of **less surprise**

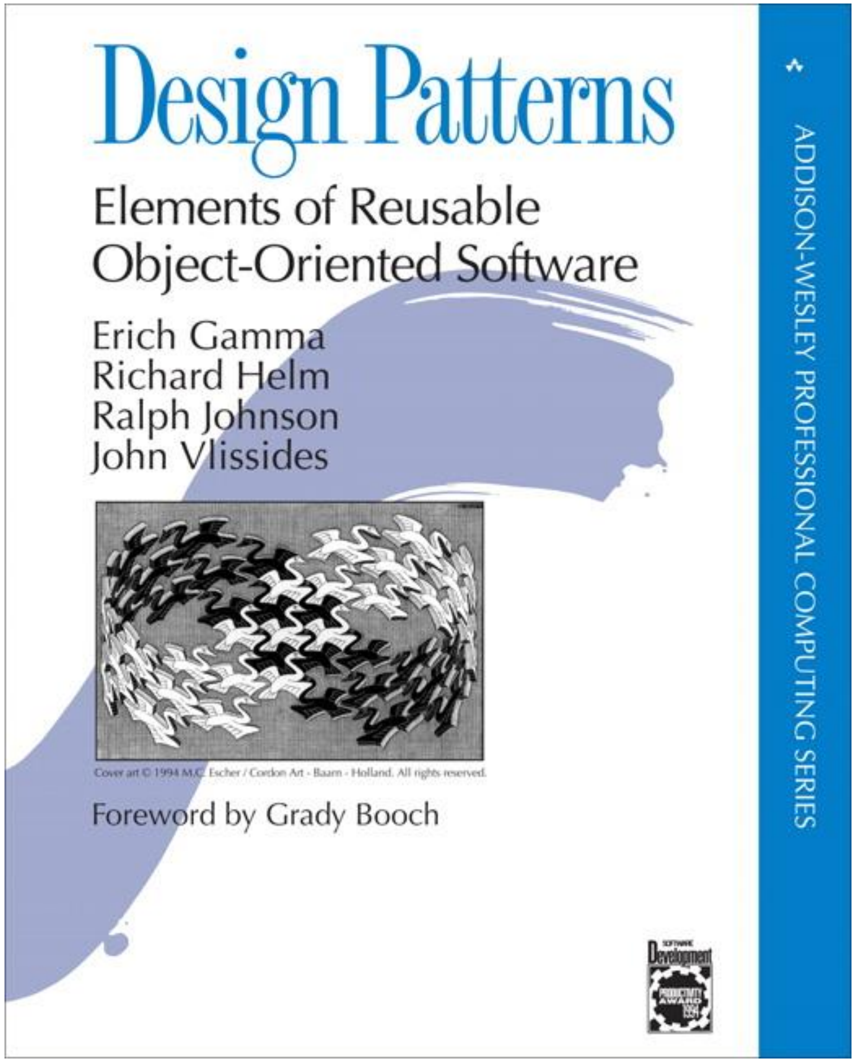
Additional limitations

- **Release is scheduled not more than in 3 months after Release Back-up Target, constant release date changes**
- *Scope is constantly changing*
- *“Low priority” features are almost not exist*
- *Architecture should be created in a way for making the process of adding new features or deleting them from the scope easy and convenient*
- **Issues in the department of functional testing**

Solution variants?

COMAQA.BY

What DPs should be used?


Design Patterns

Elements of Reusable Object-Oriented Software

Erich Gamma
Richard Helm
Ralph Johnson
John Vlissides

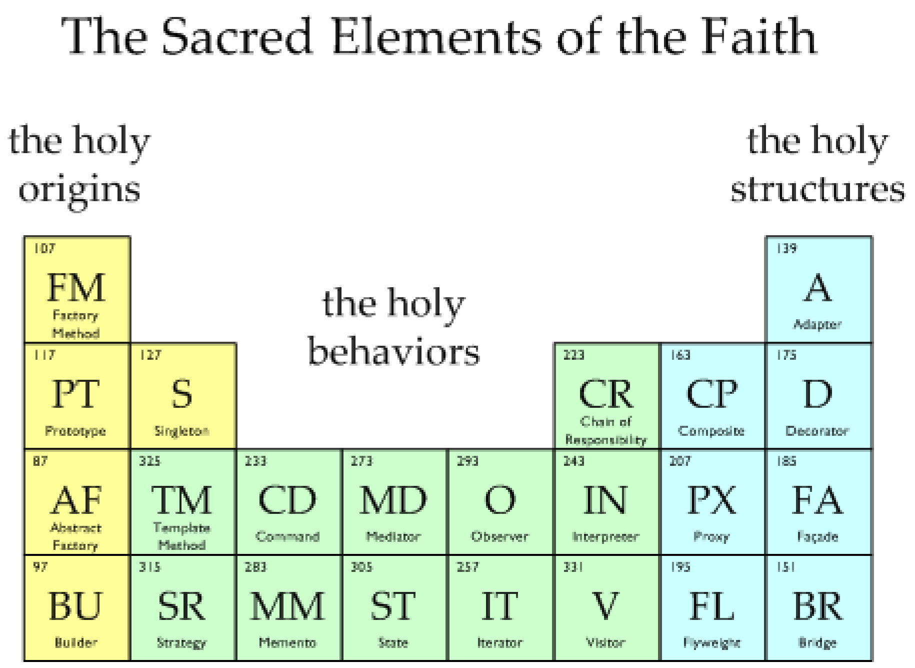
Cover art © 1994 M.C. Escher / Cordon Art - Baarn - Holland. All rights reserved.

Foreword by Grady Booch

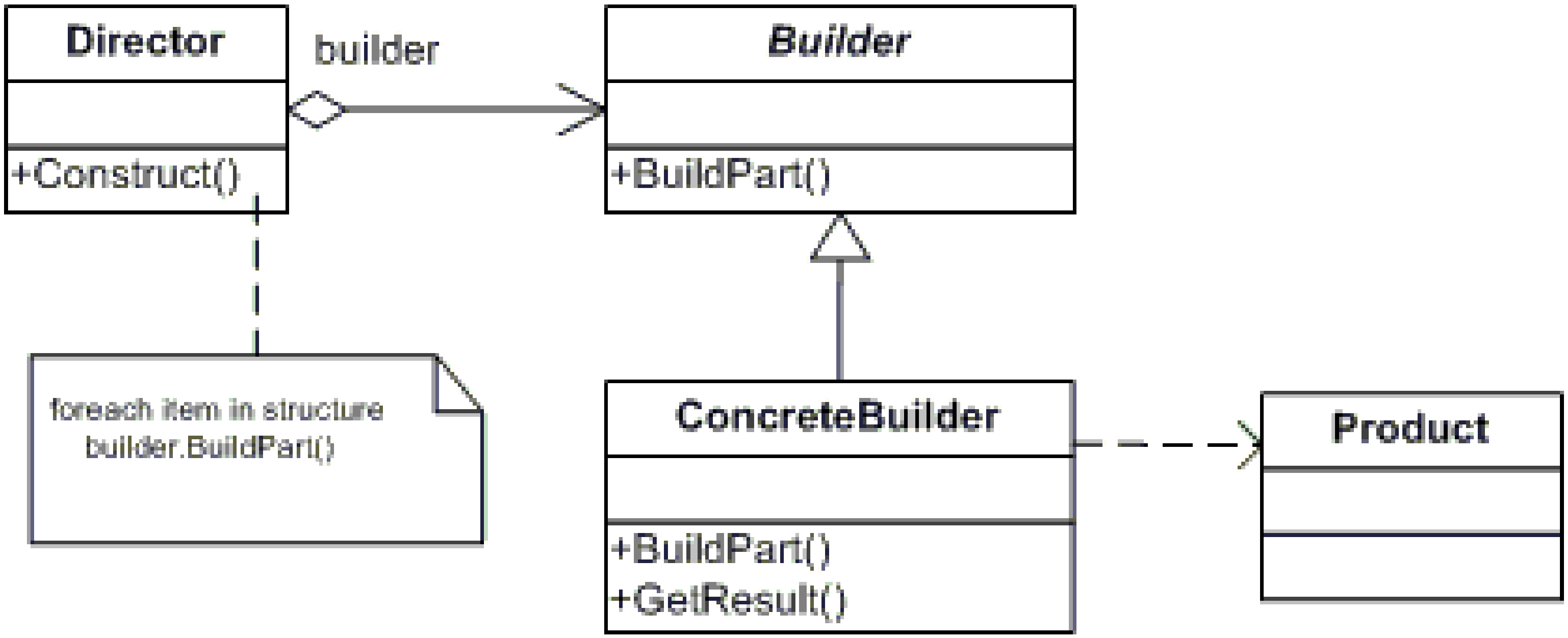
Additional limitations

List of potentially useful patterns:

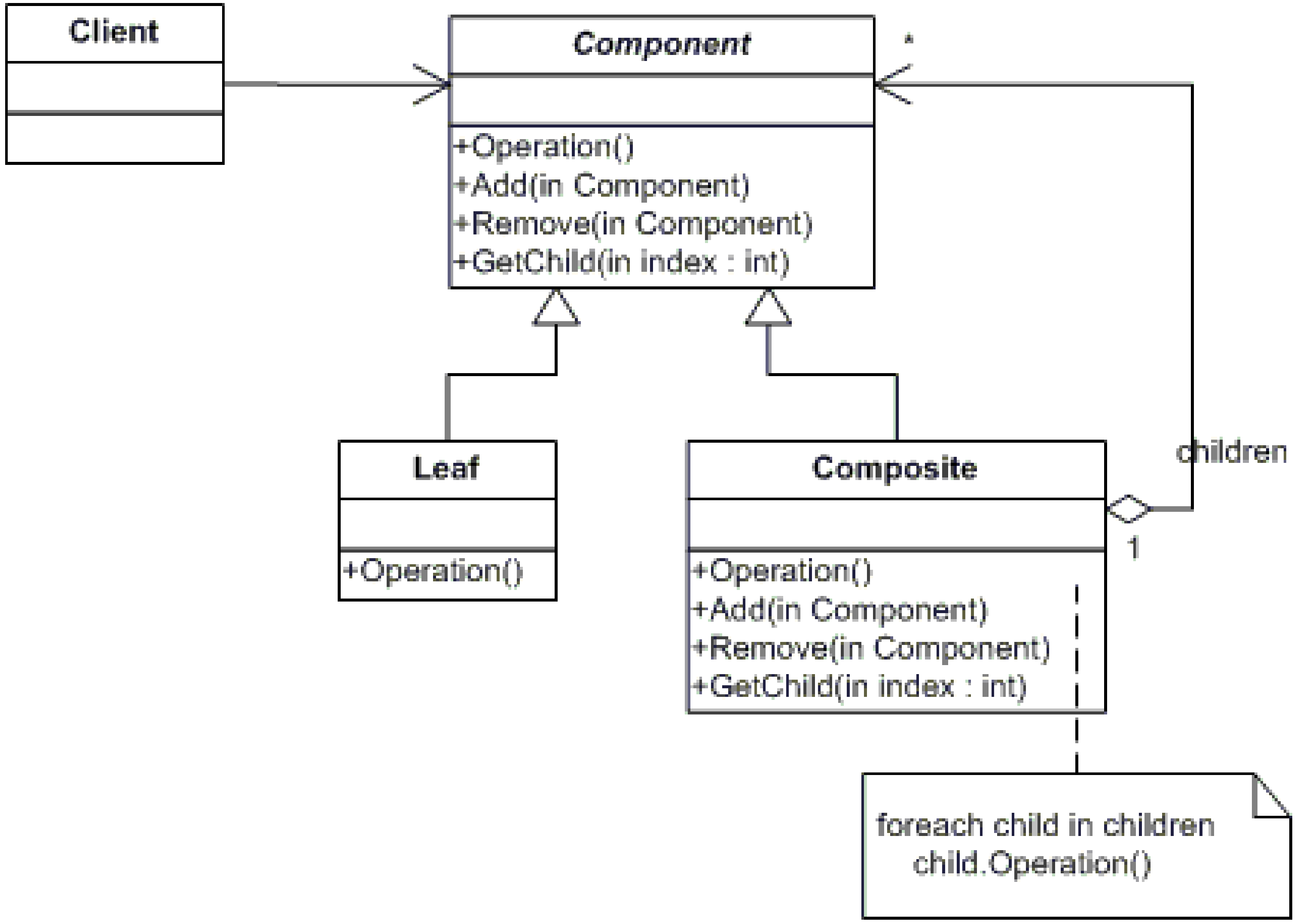
- Builder
- Decorator
- Composite
- Iterator
- Visitor
- Singleton



Builder

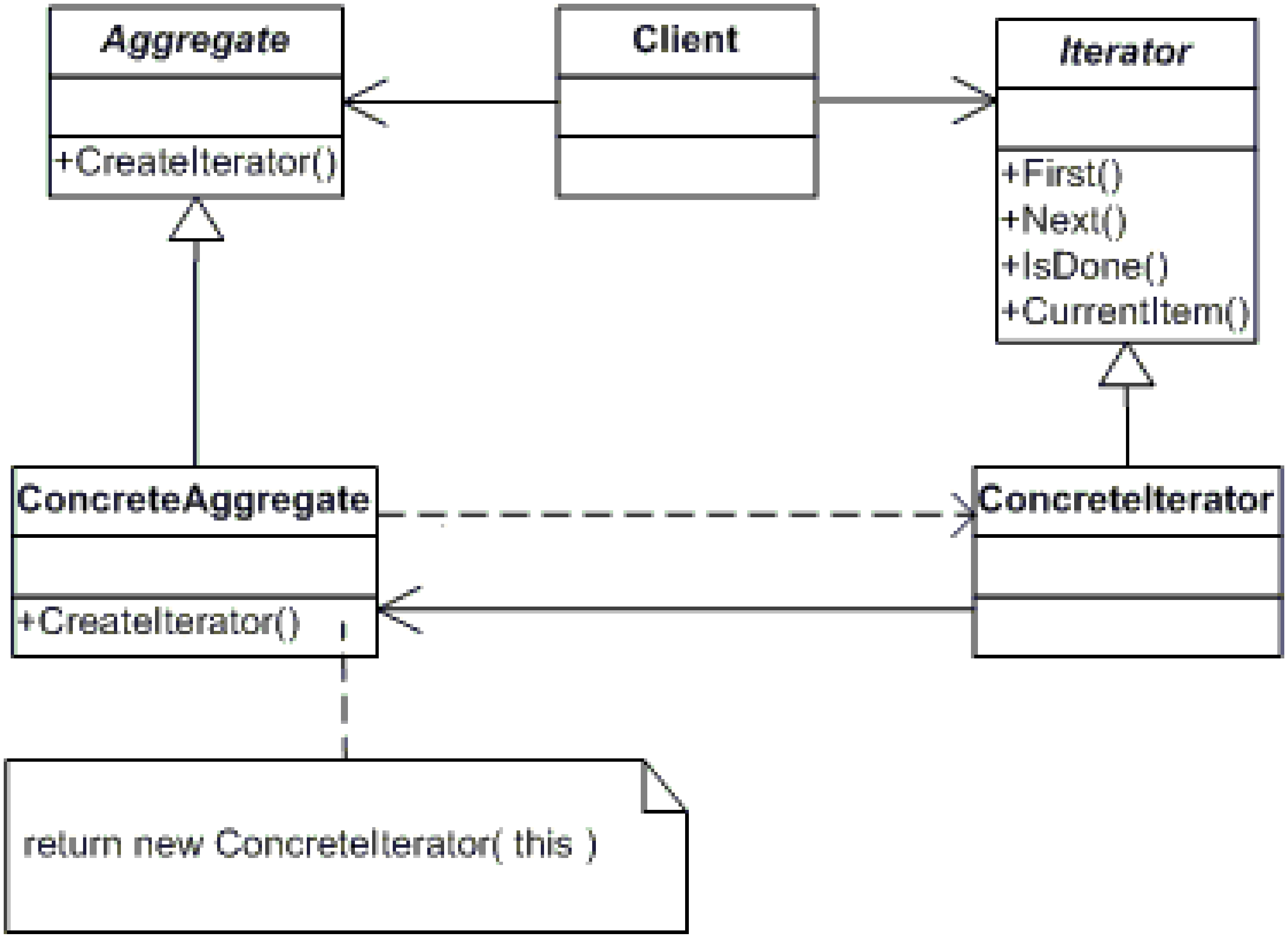

Director
+Construct()
builder
Builder
+BuildPart()
foreach item in structure
builder.BuildPart()
ConcreteBuilder
+BuildPart()
+GetResult()
Product

Composite


Client
Component
+Operation()
+Add(in Component)
+Remove(in Component)
+GetChild(in index : int)
*
Leaf
+Operation()
Composite
+Operation()
+Add(in Component)
+Remove(in Component)
+GetChild(in index : int)
children
1
foreach child in children
child.Operation()

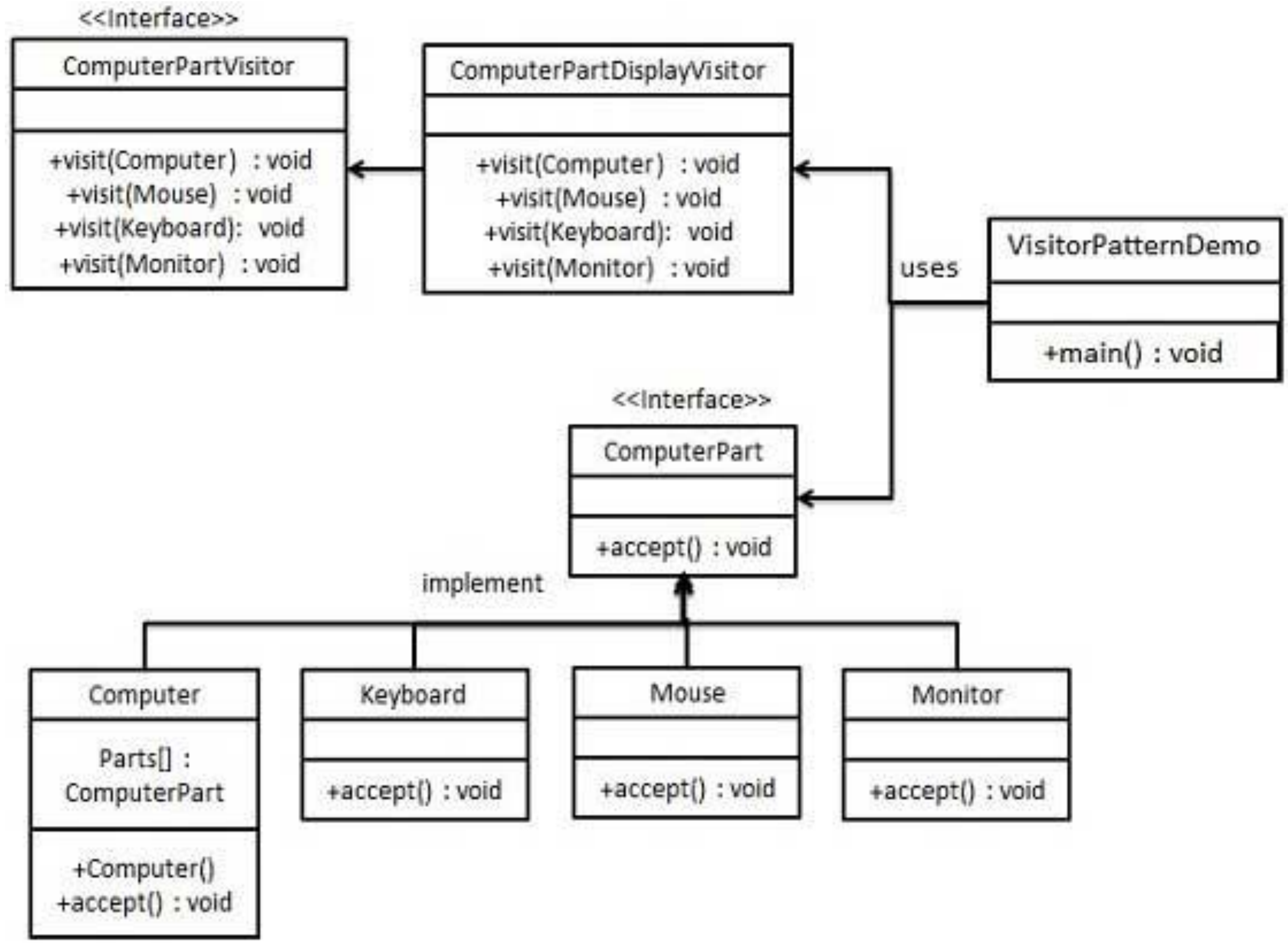
COMAQA.BY

Iterator

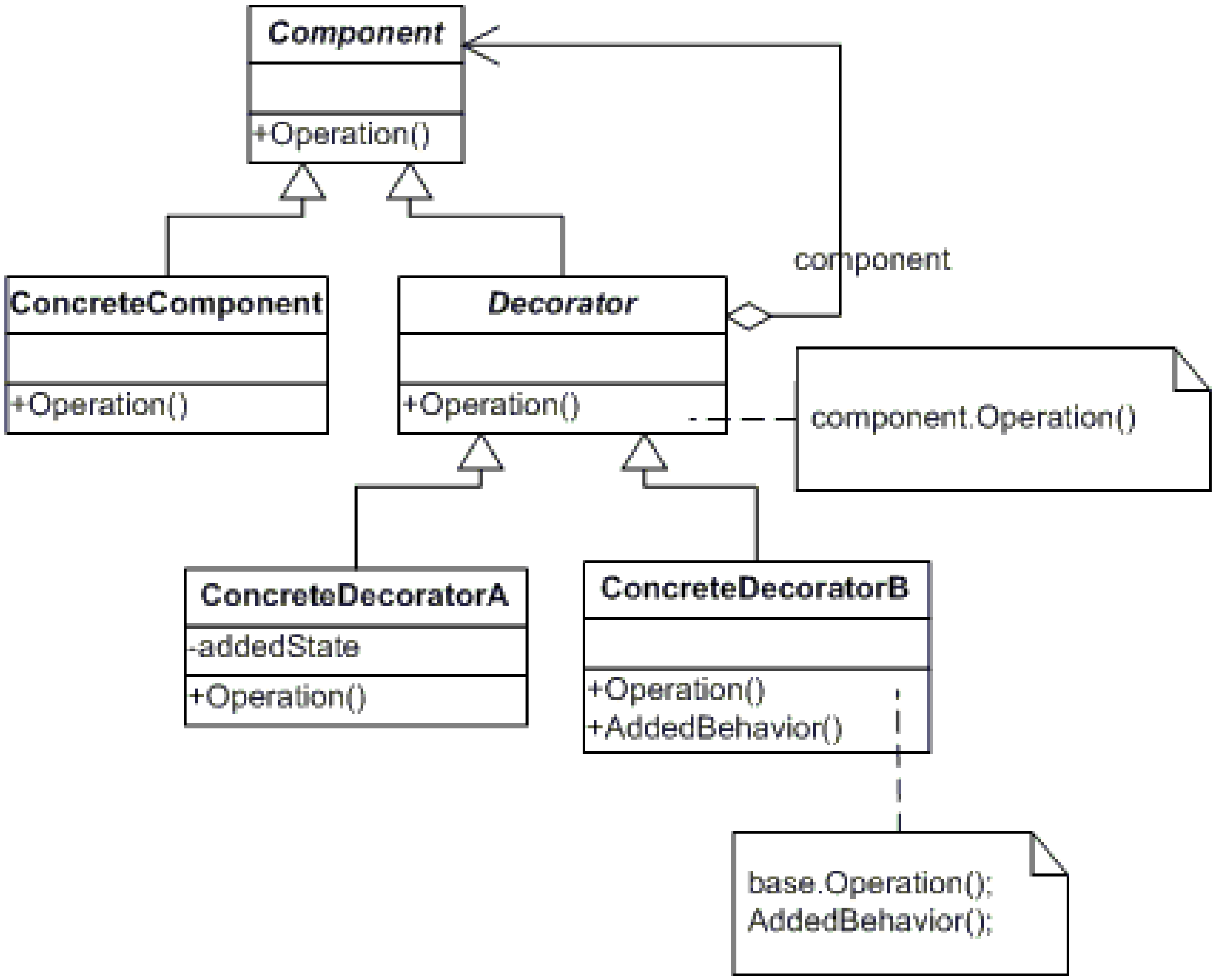


COMAQA.BY

Visitor

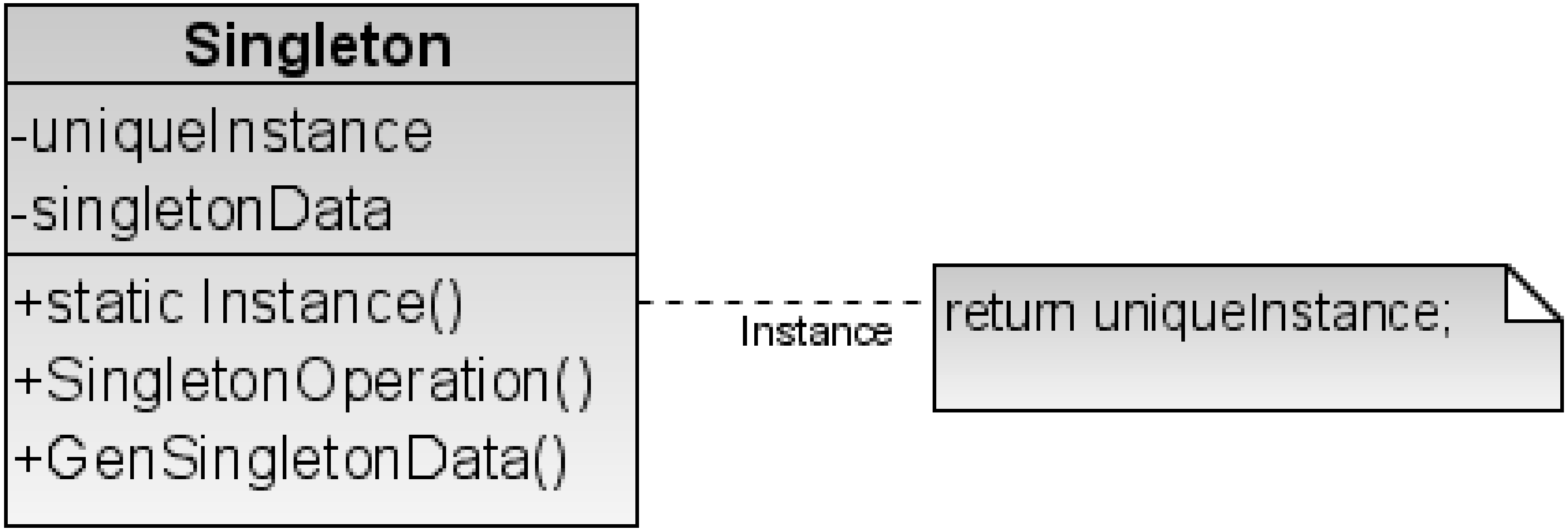


Decorator



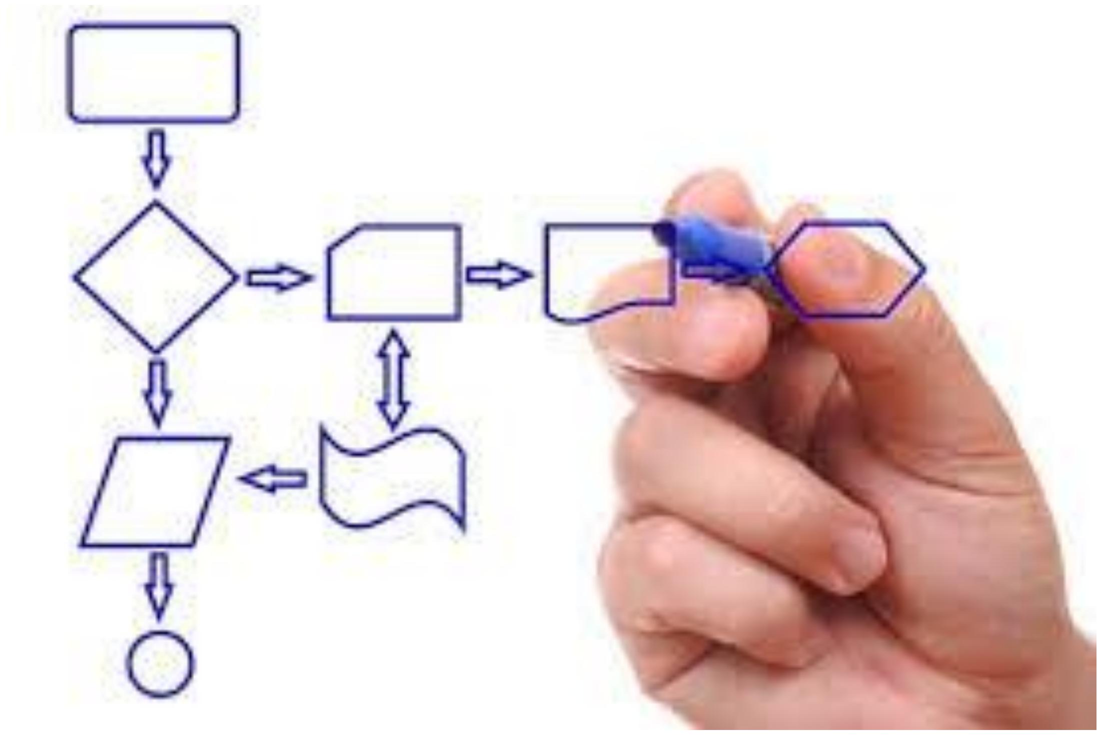
Component
+Operation()
component
ConcreteComponent
+Operation()
Decorator
+Operation()
component.Operation()
ConcreteDecoratorA
-addedState
+Operation()
ConcreteDecoratorB
+Operation()
+AddedBehavior()
base.Operation();
AddedBehavior();

Singleton


Singleton
-uniqueInstance
-singletonData
+static Instance()
+SingletonOperation()
+GenSingletonData()
Instance
return uniqueInstance;

Mapping tasks with patterns

- **Granular Backup \ Restore** (*Builder*, *Composite*)
- Different **source of information** about DB’s (*Builder*, *Composite*)
- Complex **env** (*Builder*, *Composite*, *Decorator*)



Mapping tasks with patterns

Tiny features:

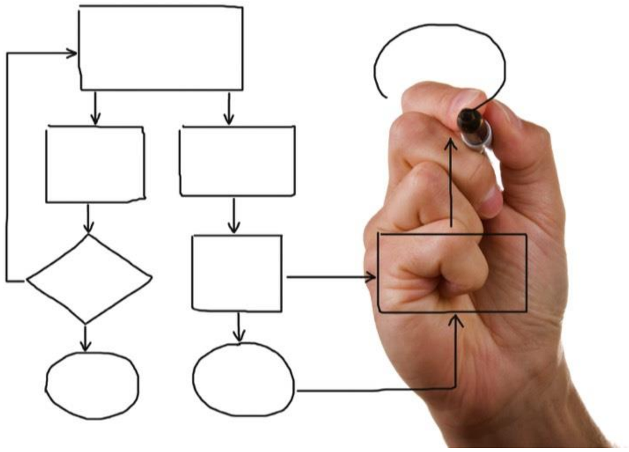
- DB or a separate file are **renamed during use process** (*Iterator*, *Visitor*)
- DB or a separate file are **moved during use process** (*Iterator*, *Visitor*)
- Variations of **naming conflicts** (*Iterator*, *Visitor*)

COMAQA.BY

Mapping tasks with patterns

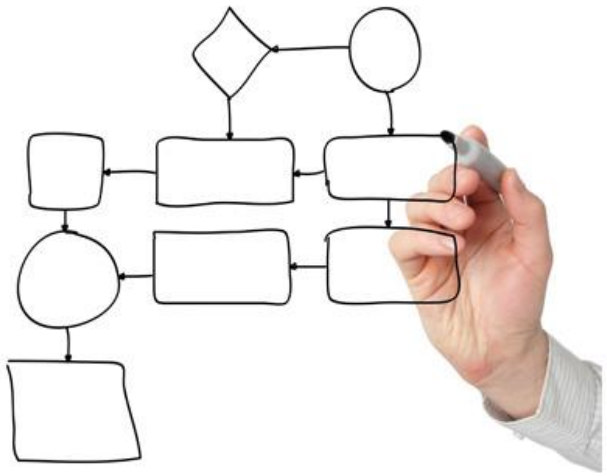
Tiny features:

- **Restore** to the different folder, with additional variants of naming **conflicts** (Iterator, Visitor)
- **Continuous operations** at the DB (Iterator, Visitor, ~Decorator)
- Other **special cases** (Iterator, Visitor, ~Decorator)



Mapping tasks with patterns

- Using **standardized mechanisms** of performing **back-up \ restore copying**, VSS (*Visitor*, *Decorator*)
- Using C# SMO for **ultra-granular back-up \ restore** for some special cases (*Visitor*, *Decorator*)



Mapping tasks with patterns

Mechanisms of optimization on VSS level

- System DB (*Composite*, *Iterator*, *Visitor*)
- User DB (*Composite*, *Iterator*, *Visitor*)
- 3 Recovery models (*Composite*, *Iterator*, *Visitor*)
- *Simple (Visitor)*
- *Bulk logged (Visitor)*
- *Full (Visitor)*

Mapping tasks with patterns

Mechanisms of server level optimization

- Storing data *(Composite, Iterator, Visitor)*
- Restore data *(Composite, Iterator, Visitor)*

Supporting of limited back-up window *(Composite, Iterator, Visitor)*

User-chosen subset of DBs *(Visitor)*

Mapping tasks with patterns

Optimizing the order of copying DB *(Visitor)*

- Multiple checks, including consistency checks *(Visitor)*
- Reports for users *(Visitor)*
- Different level of report specification *(Visitor)*

Mapping tasks with patterns

- Tracing for technical specialists *(Visitor)*
- Wide range of tracing specification *(Visitor)*
- Saving the concept of less surprise *(Iterator, Visitor)*

Linked DP’s

COMAQA.BY

Solution!

COMAQA.BY

Additional advantages 1

- **Universal architecture** for any target with C++ API (e.g. VSS Driver based)
- A **skeleton** of architecture was made, perforating tracing and logging
- Full **Unit-tests coverage**
- Decreasing and **making the testing phase cheaper**
- **Full avoiding of manual testing**
- **No blockers or major bugs, found by end-users**

Additional advantages 2

- **Working on new targets and versions for existing targets by analogue**
- **Generating actual documentation** based on the source code and unit-tests
- Simple **understanding and readability of the code**
- **Simple way of teaching** employees, effective involvement process of new people for speeding up the release process

Adapt for adjacent contexts

- MS Exchange, C++
- MS Share Point, C#

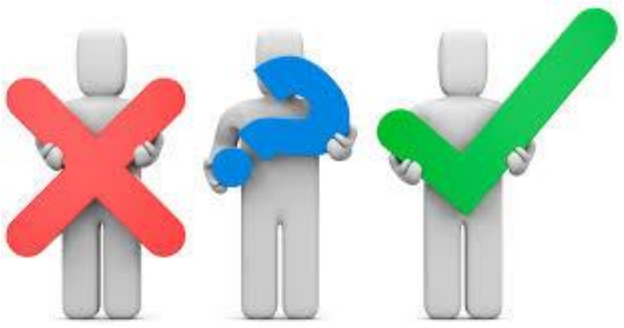
Adapt for "outer" contexts

- **File System**
- What difficulties can be?
- How to solve them?
- *Working with subset of the tree (partial loading into RAM)*
- *Apply Visitor DP not one by one for all of the nodes and than change to next visitor, but all the Visitor DP for one node and them proceeding to the next one.*
- *Updating Iterator DP*

COMAQA.BY

Results

- **Solved** the standard challenges
- **Solved** project-specific challenges
- **Met the budget and time limits**
- **Architecture was awarded as the best in the company**
- It has become **the iconic one** in the company



Results

- **Skeleton** of the **architecture** became a pre-made template
- Product was **awarded by MS**
- **Product was first to go out on the word stage**
- Met almost all the user requirements after the first release
- Second version met all the user requirements and was released several months later because of the Architecture
- ***It couldn't prevent the company for becoming a bankrupt ☺***

Conclusions about DP

- *There is an* **opinion** *especially between super skillful programmers, that DPs are shackling you and are not supposed to be used by a professional programmer*
- You should always take **best examples** of other implementations
- **Learn DP at any loose**
- **Think about architecture beforehand**
- Find the **balance** for your exact project **between flexibility (Agile), Lean** and experience-expertise, preliminary projecting of architecture

Conclusion about processes

- **Prototyping**
- **“Technical” sprints**
- Ways of provision of **high-quality software:**
- *Always a complex of plans*
- *Strict following the coding standards*
- *Effective tool for code reviews*
- *High coverage of Unit tests*
- *Mocking*

Conclusion

Conclusion about processes

- **CI**
- Automated **static code analysis**, running subsets of Unit tests as **pre commit** event, **pre-commit** code-review, review **lead time** as metric of the process, running all Unit tests as **post-commit** event
- Process is not a goal but a tool
- **Iterative process** of development of non-classic scrum that is adapted for your needs
- **Balance** between specialization and concept of universal soldier

What’s next?

- Read books
- Read source code
- Practice, Practice and Practice

Recommended literature

1. Grady Butch “**Object oriented analysis and design with examples of apps on C++**”

- *Notes: you should not be scared of C++ examples ☺, 95% of the material is conceptual, no strict attached to the exact language. In my opinion it might look too simple, and because of that it’s far better to read at before going to bed.*

Recommended literature

2. Martin Fowler “**Refactoring**”

- *Notes: IMHO is should be totally read from end to end, twice, in order to make the contents of your book as your professional luggage (was using the “contents of that book the same way”).*

Recommended literature

3. David Thomas, Andrew Hunt “**The Pragmatic Programmer: From Journeyman to Master**”

- *Notes: Amazing book that consists of a ton of advices. IMHO strongly recommend to read from cover to cover, twice, in order to have contents of the book - you active professional luggage. And then look through different chapters before talking to a customer.*

Recommended literature

4. Gang of four “**Design patterns**”

- *Notes: IMHO strongly recommend to read from cover to cover, twice, in order to have contents of the book - you active professional luggage.*

5. Steve McConnel “**Code complete**”

- *Notes: IMHO No need to be afraid of the size of the book ... it should be read or before “going to bed”, or from any place, of separate chapters, just to fresh things in the memory in the chosen field of problem.*

Recommended literature

6. **“Pattern-Oriented Software Architecture”** Volume 1-3

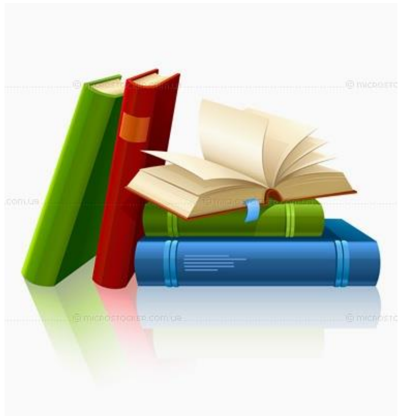
- *Notes: IMHO should be read from start to the end.*

7. **“Domain Specific Languages”**, Martin Fowler

- *Notes: IMHO should be read from start to the end.*

8. **“Patterns of Enterprise Application Architecture”**, Martin Fowler

- *Notes: IMHO should be read from start to the end.*



Thanks for your attention

Anton Semenchenko
DPI.Solutions
EPAM Systems

Skype: dpi.Semenchenko

+375 33 33 46 120
+375 44 74 00 385

www.comaqa.by
www.corehard.by

Scan me

https://t.me/corehard_by

COREHARD.BY

C++ COMMUNITY

COREHARD

AUTUMN 2018

NOVEMBER
2-3
MINSK

The conference is organized by C++ CoreHard Community with kind support of leading Belarusian and Russian IT companies in order to discuss best practices in low-level development in C/C++, programming of controllers, Internet of Things, high-load server solutions and other kind of hardcore development

http://conference.corehard.by

Scan me

Scan me

https://t.me/comaqa

October 5-6

Minsk

COMAQA Conference

http://conference.comaqa.by

Scan me