AGILE GURGAON 2016

27-28 May 2016
The Leela Ambience Hotel,
Gurgaon
www.agilegurgaon.com

Change Vector Tracking

A reflective approach towards designing large scale software

BY
Ranjith Tharayil
Senior Agile Consultant & Technical Coach
rtharayil@soultionsiq.com

Expectations and take always

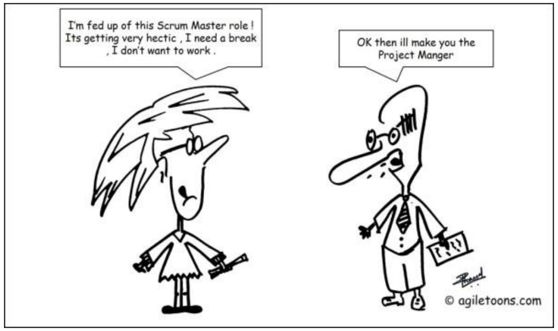
- Understand the problem wrt design debt in large scale software systems.
- Exposure to a novel technique called Change Vector Tracking (CVT)
  - Can start applying it right away
- Knowledge of how change can be modelled as a weighted vector
- Using CVT to make informed decision on architecture refactoring.

About Me

Ranjith Tharayil

SolutionsIQ®
Advancing Enterprise Agility

Printer
ComputerGraphics
HumanComputerInteraction
SoftwareArchitecture
Multimedia MachineLearning
Gui ImageProcessing
Healthcare
Android
EmbeddedSystems

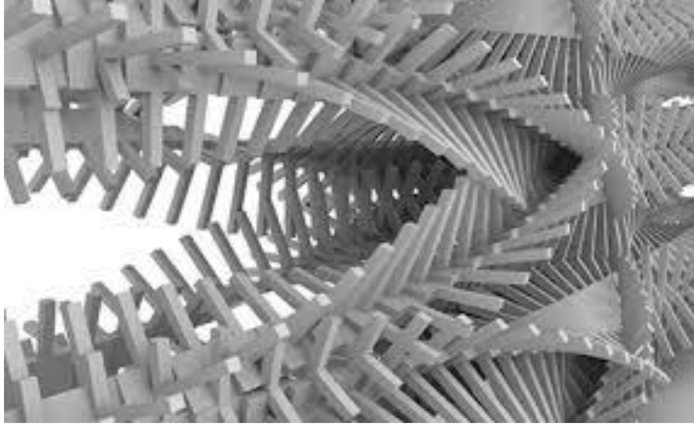

I'm fed up of this Scrum Master role ! Its getting very hectic , I need a break , I don't want to work .
OK then ill make you the Project Manger
© agiletoons.com

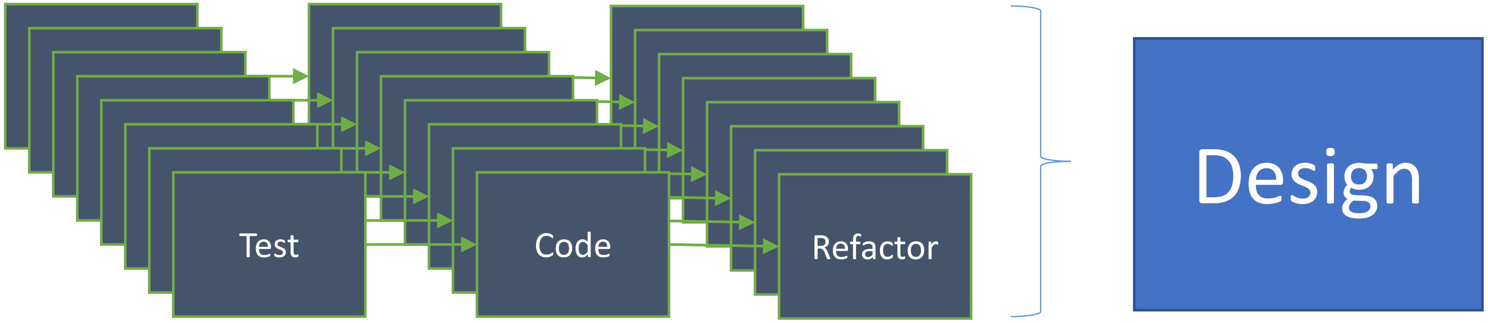
in https://in.linkedin.com/in/ranjiththarayil

“Design Is a Wicked Problem , a “wicked” problem is one that could be clearly defined only by solving it, or by solving part of it .This paradox implies, essentially, that you have to “solve” the problem once in order to clearly define it and then solve it again to create a solution that works”

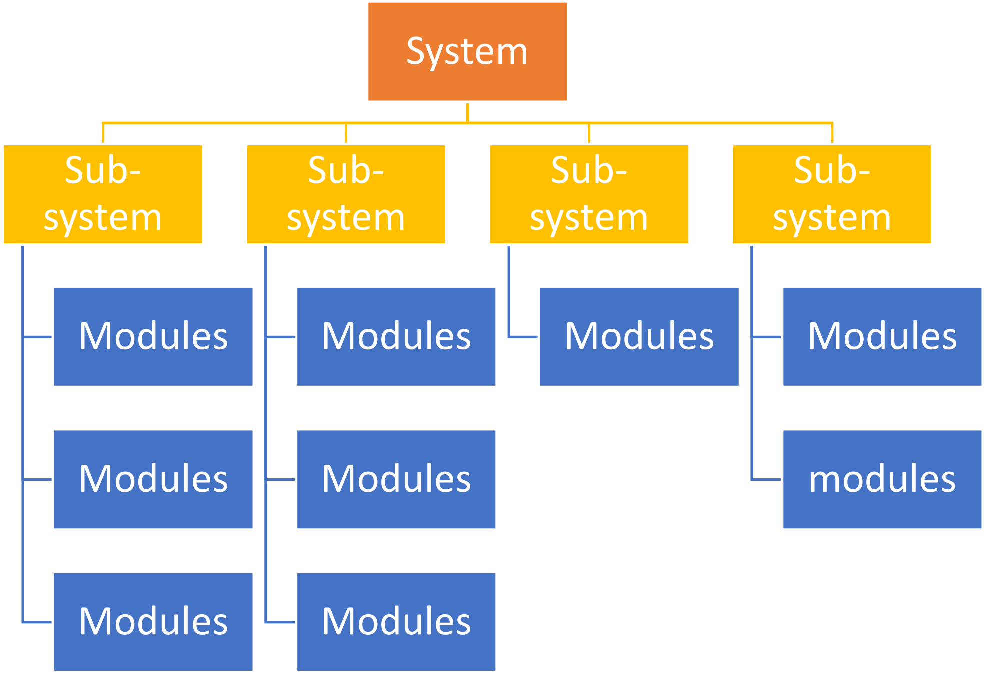
- McConnell, Steve. *Code complete*. Pearson Education, 2004.

Emergent design





Problem : Refactoring less focus on architecture



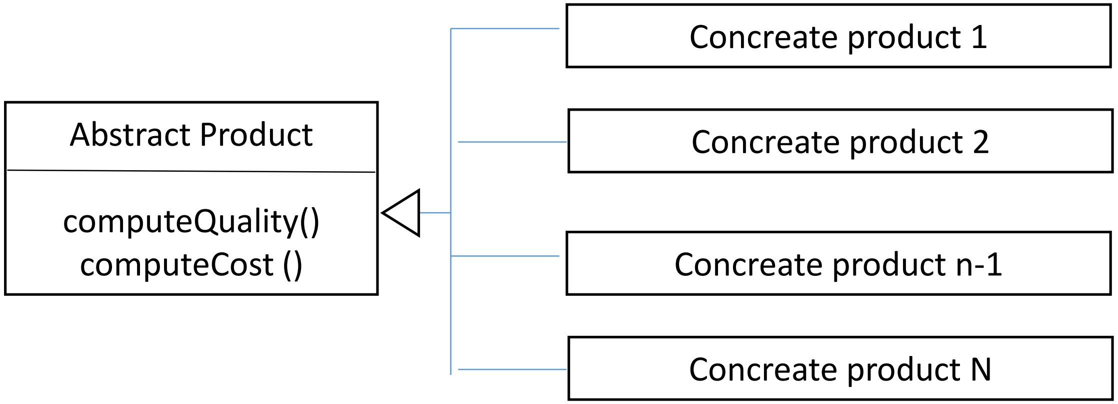
Refactoring , **observed trends**

- » More focus on classes inside modules
  - » Less focus on Modules ,sub-Subsystem and system
- » Inexperience team members
- » Accumulate design debt
  - » Technical debt

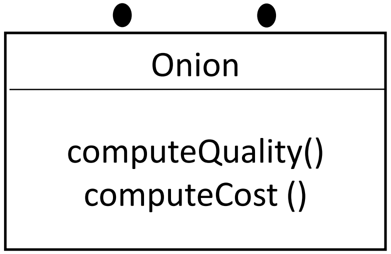
Example : Super Market

I 'am using a simple example to explain the problem , same is applicable for bigger complex systems

- The super market has many products to sell
- Each product has a
  - quality
  - cost
- The rules for computation of Quality and Cost are different for each product

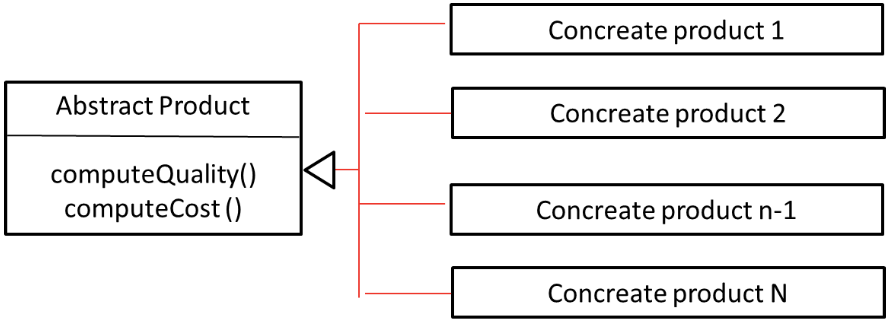


Is this the best design ?



Onion

computeQuality()
computeCost ()



Abstract Product

computeQuality()
computeCost ()

Concreate product 1

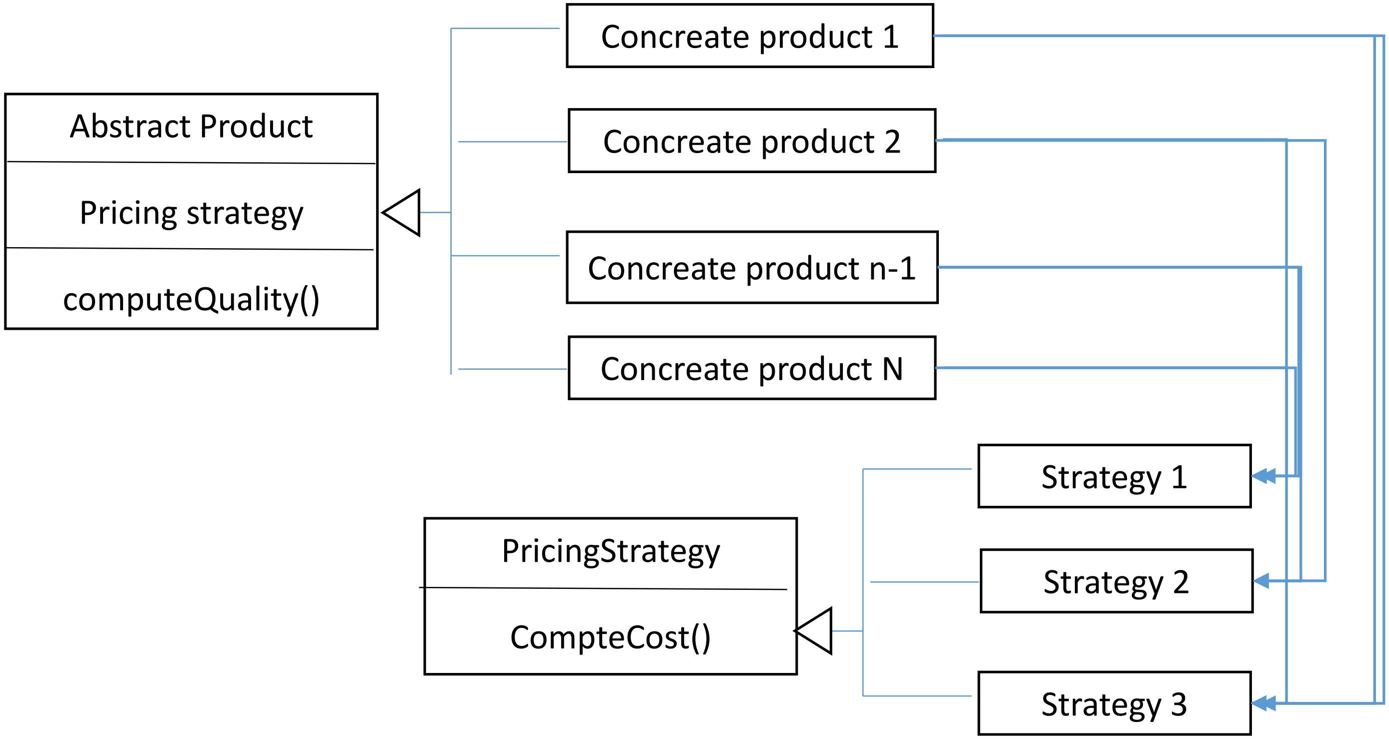
Concreate product 2

Concreate product n-1

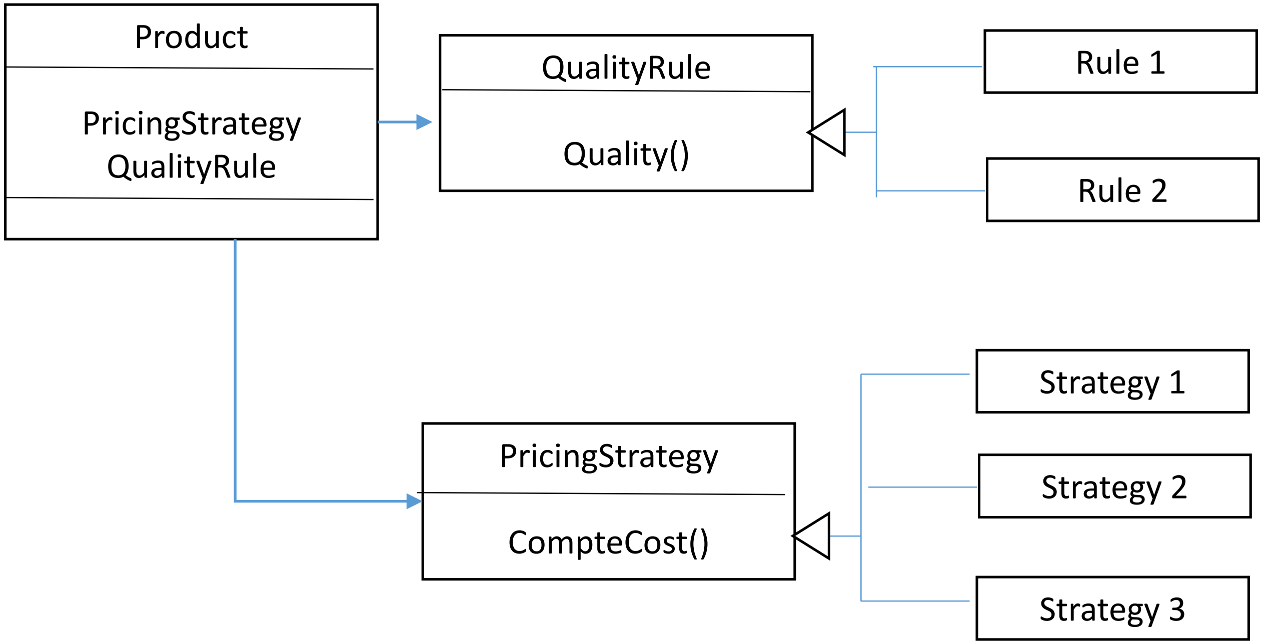
Concreate product N

- Is this following SRP ? Single responsibility principle
- What if the pricing strategy of a product varies from time to time ?

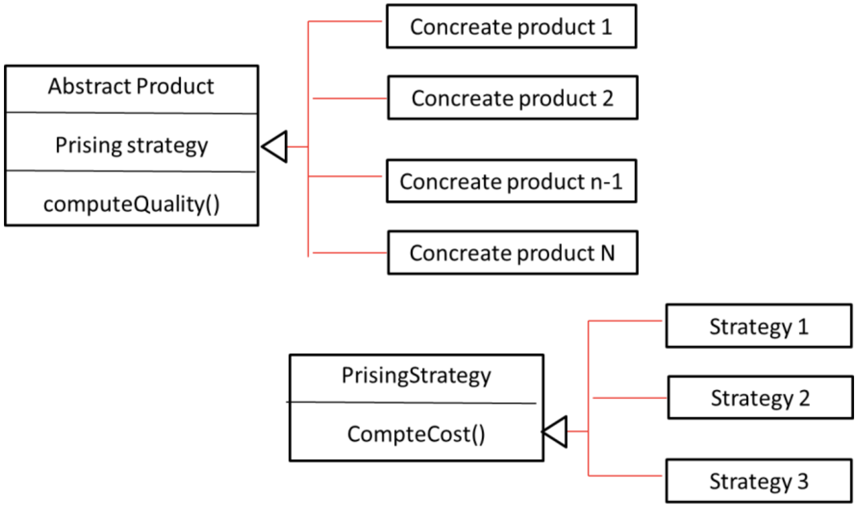
How about a different design



Another design



Which design is better ?



Abstract Product

Prising strategy

computeQuality()

Concreate product 1

Concreate product 2

Concreate product n-1

Concreate product N

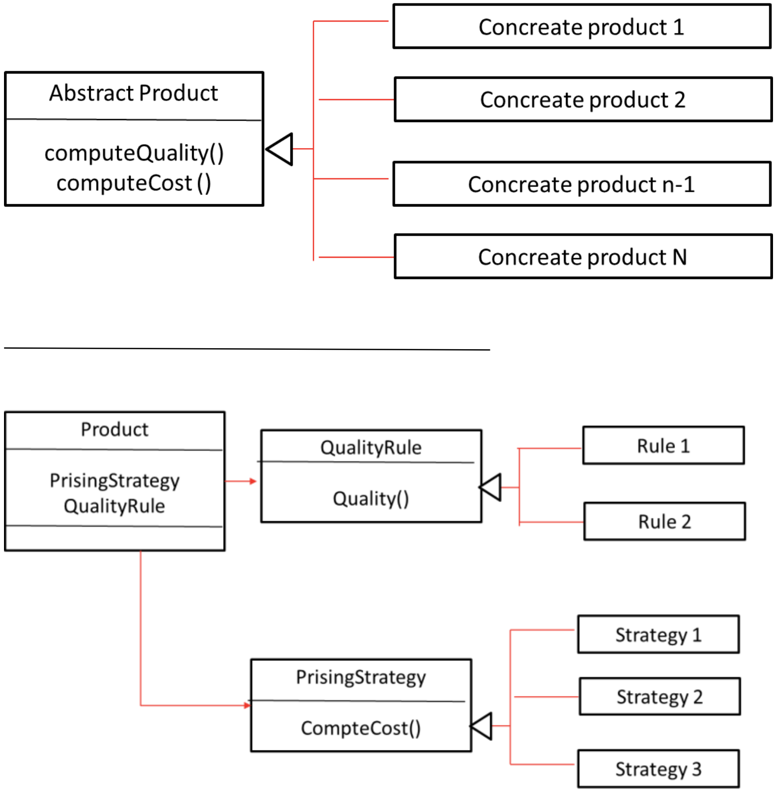
PrisingStrategy

CompteCost()

Strategy 1

Strategy 2

Strategy 3



Abstract Product

computeQuality()
computeCost ()

Concreate product 1

Concreate product 2

Concreate product n-1

Concreate product N

Product

PrisingStrategy
QualityRule

QualityRule

Quality()

Rule 1

Rule 2

PrisingStrategy

CompteCost()

Strategy 1

Strategy 2

Strategy 3

Which design is better ?

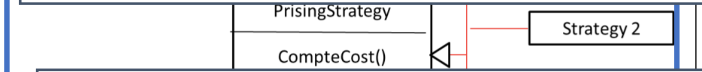
- Answer is : it depends
- Dependents on what ?

Context

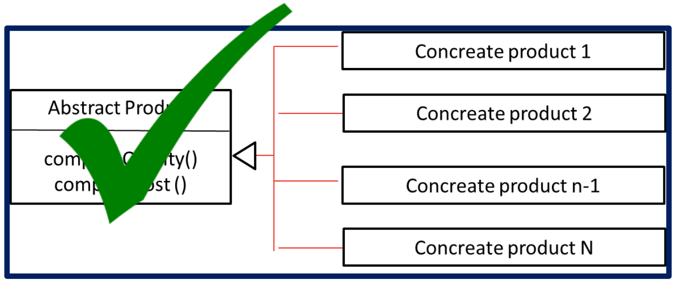
- Lest try to define this context

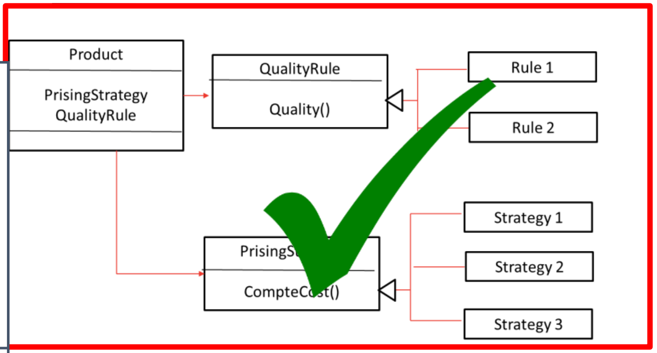
Which design is better ?

- We will be only adding new products



- We need to often modify pricing strategy & Quality rule for existing products

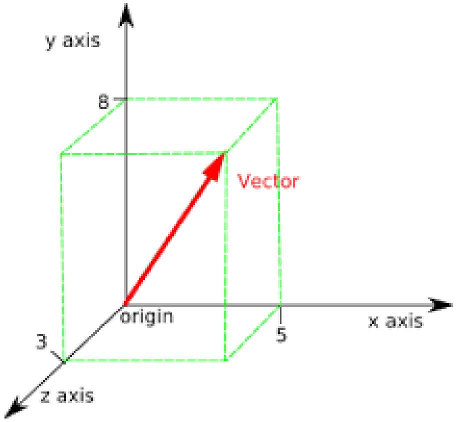




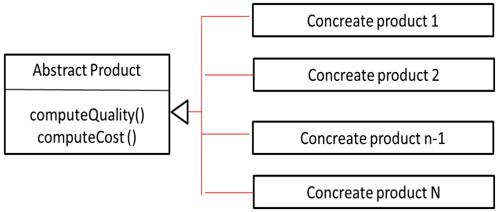
Design should depend on how the system is behaving currently

- The behaviour of a system changes with time
- Due to
  - new requirements
  - enhancements
- Change is constant and can disrupt design & architecture
- Need to track change and its effect on design/architecture

Modelling change as a **vector**



- Change has multiple dimensions
- Hence it can be quantified using a vector like ax+by+cz
- Example:



Abstract Product

computeQuality()
computeCost ()

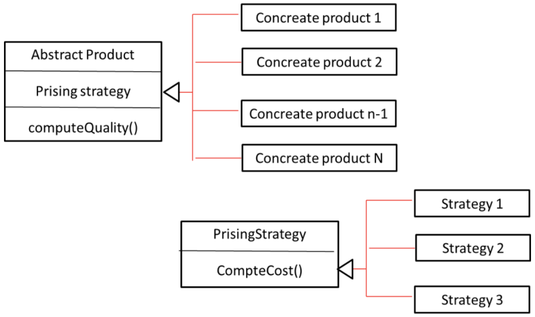
Concreate product 1

Concreate product 2

Concreate product n-1

Concreate product N

4(New Products) +
1(Miscellaneous)



Abstract Product

Prising strategy

computeQuality()

Concreate product 1

Concreate product 2

Concreate product n-1

Concreate product N

PrisingStrategy

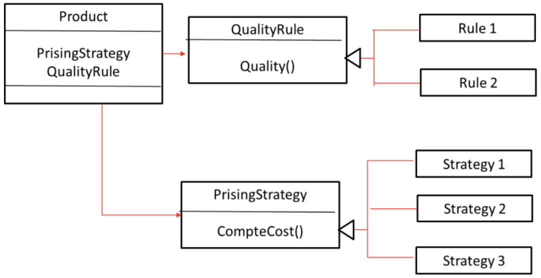
CompteCost()

Strategy 1

Strategy 2

Strategy 3

3(New Products) +
2(Modify Pricing strategy)



Product

PrisingStrategy
QualityRule

QualityRule

Quality()

Rule 1

Rule 2

PrisingStrategy

CompteCost()

Strategy 1

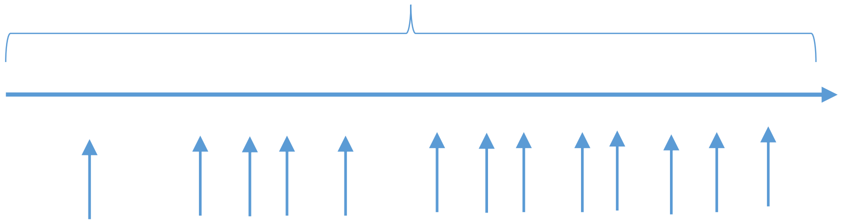
Strategy 2

Strategy 3

2(Modify Pricing strategy) +
2(Modify Quality rule) +
1(Miscellaneous)

How change can be modelled as a weighted vector ?

- We need to periodically review & document change vector for our system

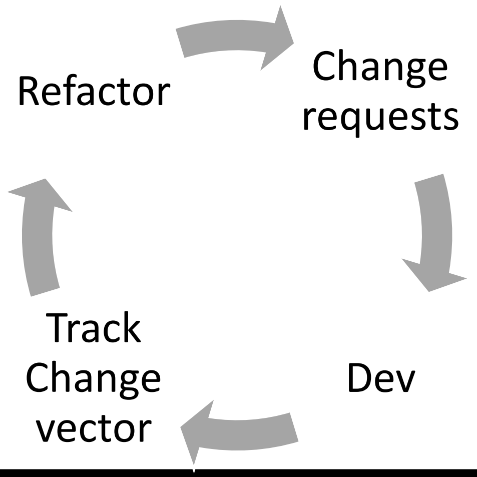


Few Months

4(New Products)
+
1(Miscellaneous)

3(New Products)
+
2(Modify Pricing strategy)

Change: new requirement , Enhancement



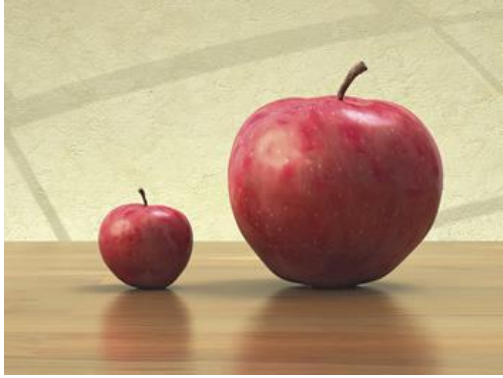
Refactor

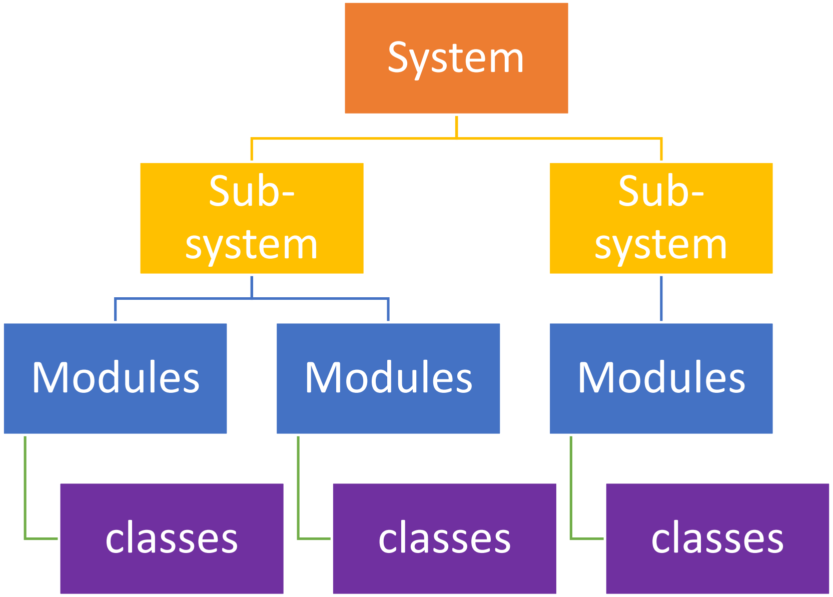
Change requests

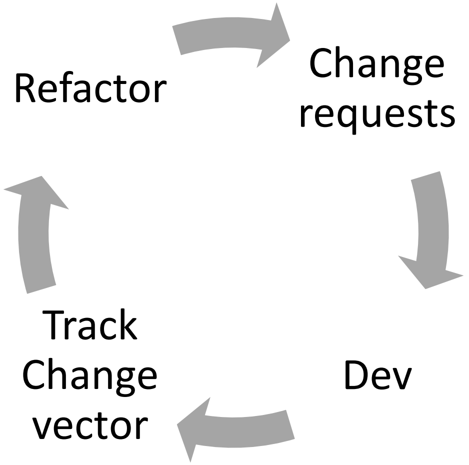
Dev

Track Change vector

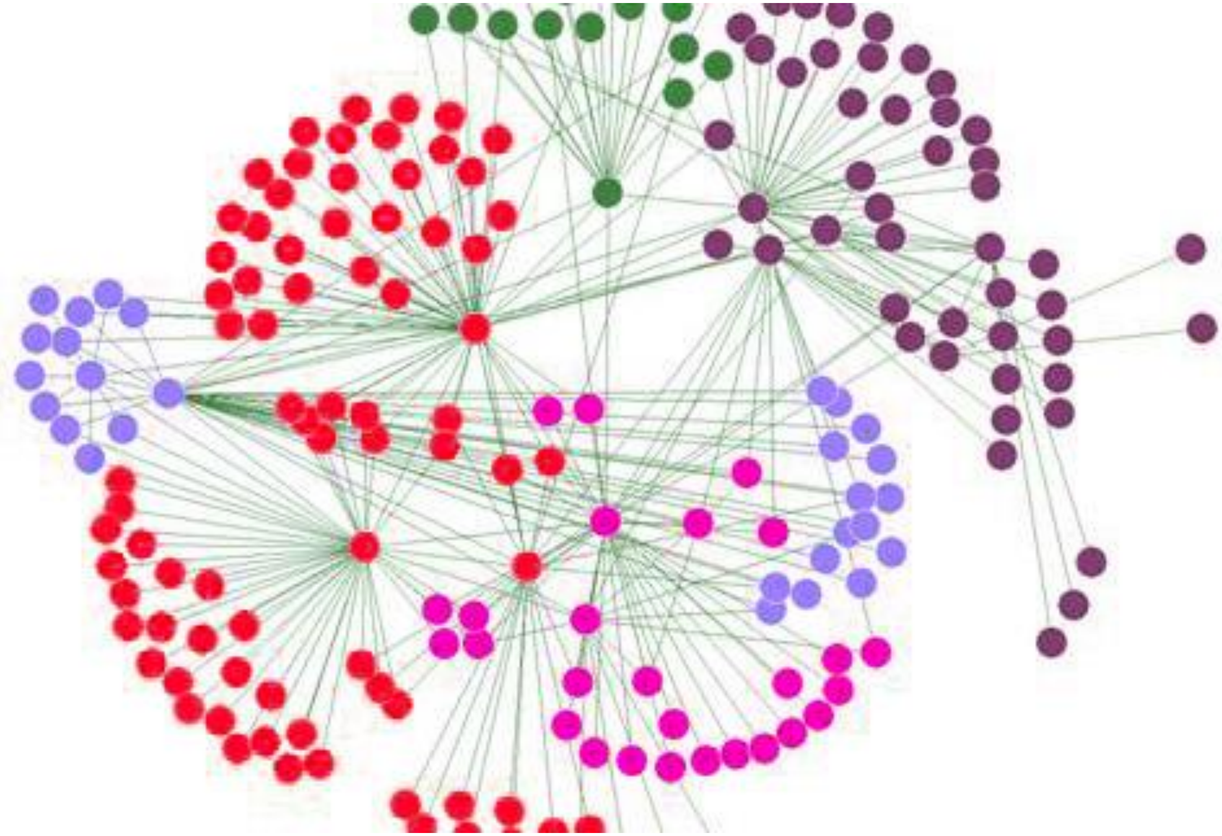
Change vector tracking from Architecture to class







Change vector tracking in context of micro services

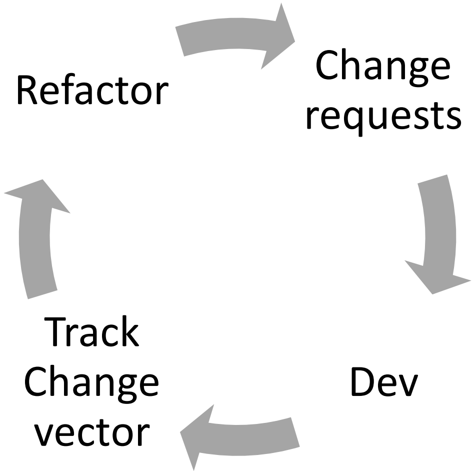


Process and Ceremonies

- Change vector tracking meeting
  - with TL , Architects
  - Every month initially
  - Cadence decide by need
  - Time boxed

- Analyse change request from
  - Source control
  - Analyse product backlog

- Document Change vector
- Identify if any design debt
- Prioritize (design debt) and push a story to your back log



Change requests

Dev

Track Change vector

Refactor

Re-visiting expectations

- Understand the problem wrt design debt in large scale software systems.
- Exposure to a novel technique called Change Vector Tracking (CVT)
  - Can start applying it right away
    - Add this jargon to your resume :P
- Knowledge of how change can be modelled as a weighted vector
- Using CVT to make informed decision on architecture refactoring.

Questions

Acknowledgments :
After hearing out this technique Sharad Julka coined the term reflective design .