Software System Architectures (NSWI130)

Modifiability

Martin Nečaský

Faculty of Mathematics and Physics

Charles University in Prague

Change is the only constant in the universe.
It is ubiquitous in the software lifecycle.

Our interest in modifiability centers on the cost and risk of making changes.

4 modifiability questions

- What can change?
  - functions, platform, external systems, qualities, information model, …

4 modifiability questions

- What can change?
  - functions, platform, external systems, qualities, information model, …
- What is the likelihood of the change?
  - estimate the likelihood and prioritize

4 modifiability questions

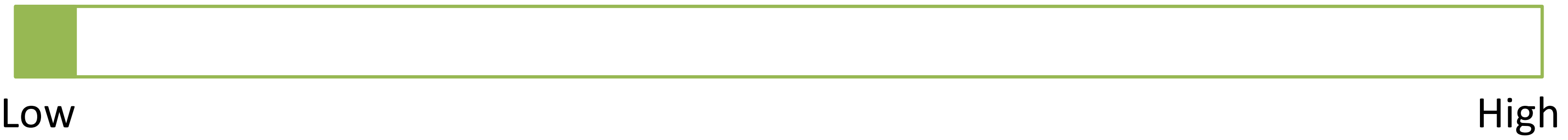
- What can change?
  - functions, platform, external systems, qualities, information model, …
- What is the likelihood of the change?
  - estimate the likelihood and prioritize
- When is the change made and who makes it?
  - design-time, developer
  - run-time, admin|user

4 modifiability questions

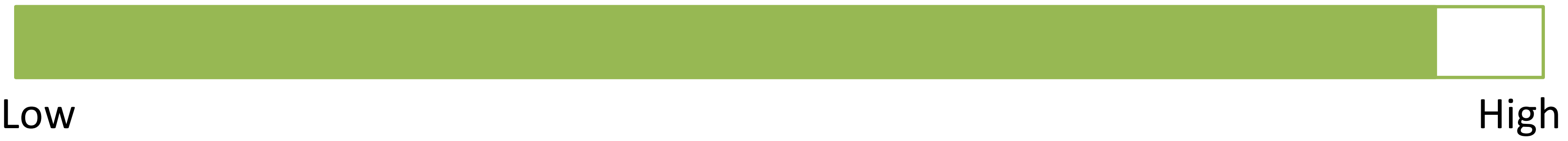
- What can change?
  - functions, platform, external systems, qualities, information model, …
- What is the likelihood of the change?
  - estimate the likelihood and prioritize
- When is the change made and who makes it?
  - design-time, developer
  - run-time, admin|user
- What is the cost of the change?
  - cost of introducing mechanism for changing system
  - cost of changing system with mechanism

Cost of changes with pure coding

cost of introducing mechanism for changing system

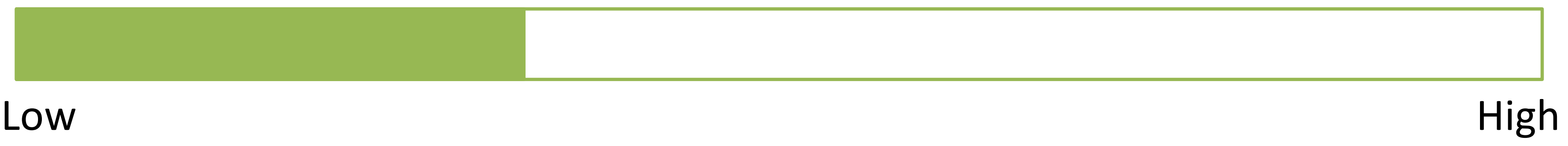


cost of changing system with mechanism

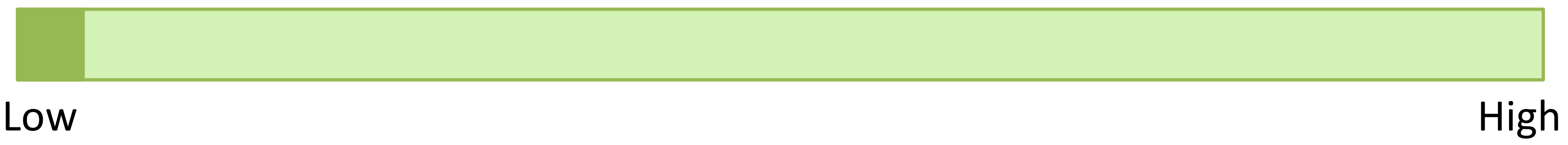


Cost of changes with application generator

cost of introducing mechanism for changing system

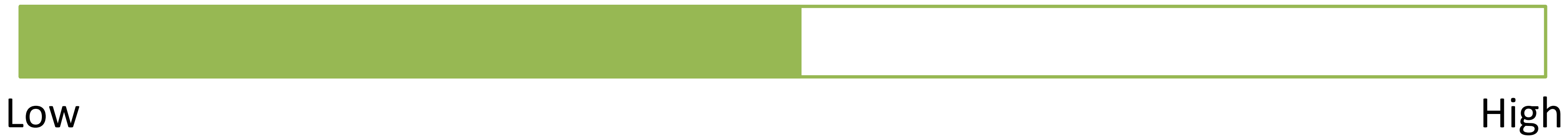


cost of changing system with mechanism

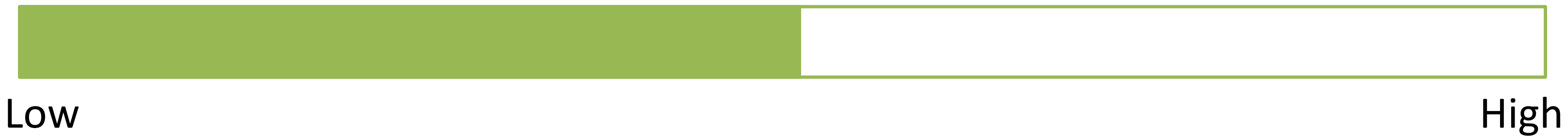


Cost of changes with devops or similar

cost of introducing mechanism for changing system

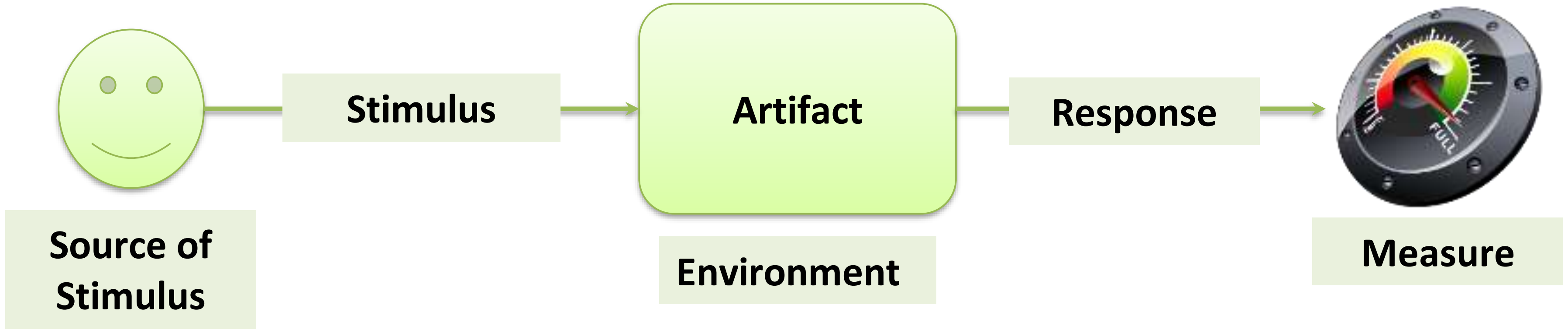


cost of changing system with mechanism



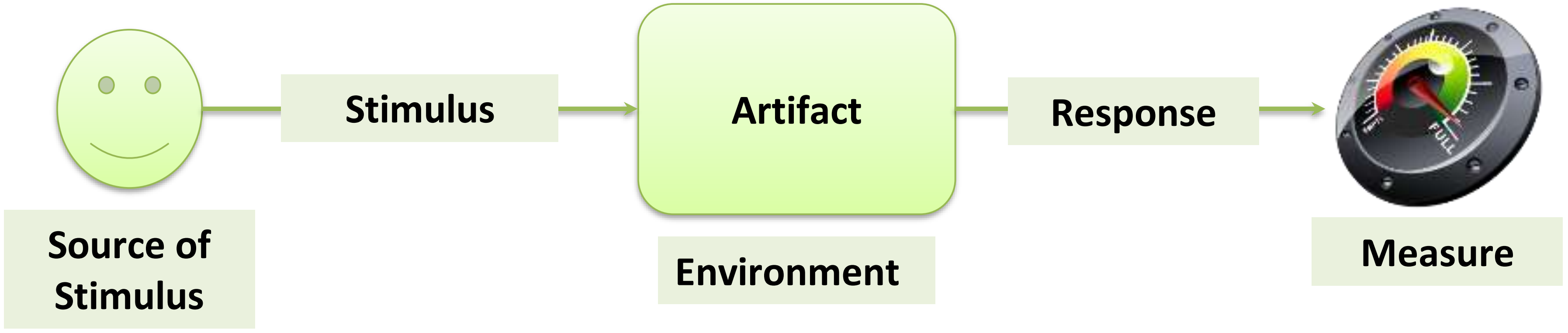
Modifiability Requirement Scenario

- Source of stimulus needs to change artifact.
- Change to be made is the stimulus.
- System or the team ensures that the change is made and the result is tested and deployed.
- Required parameters of the change are measured and meet given constraints.



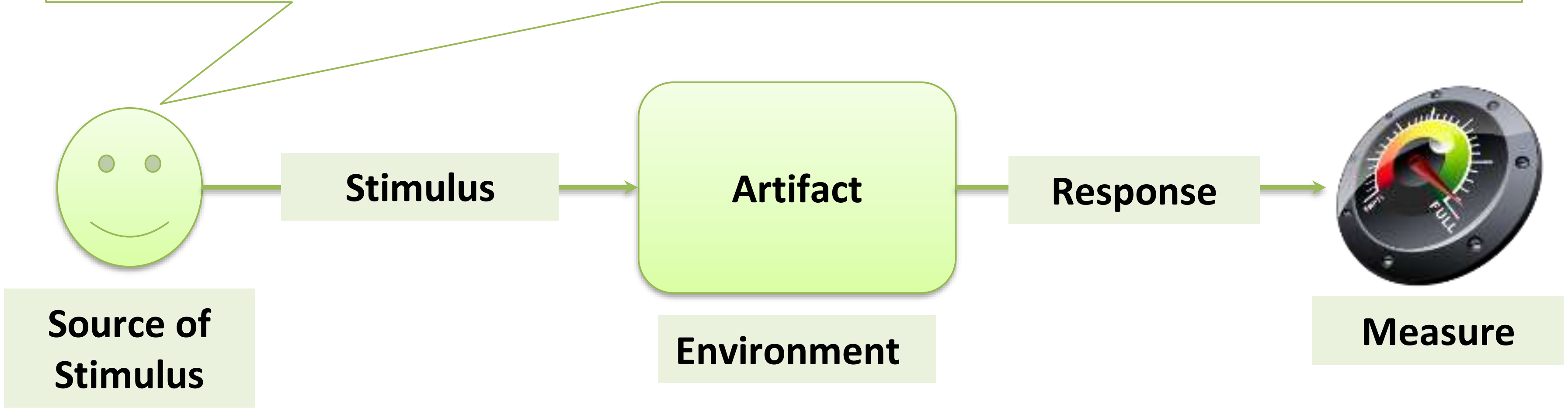
Modifiability Requirement Scenario

- module or component which needs to be changed



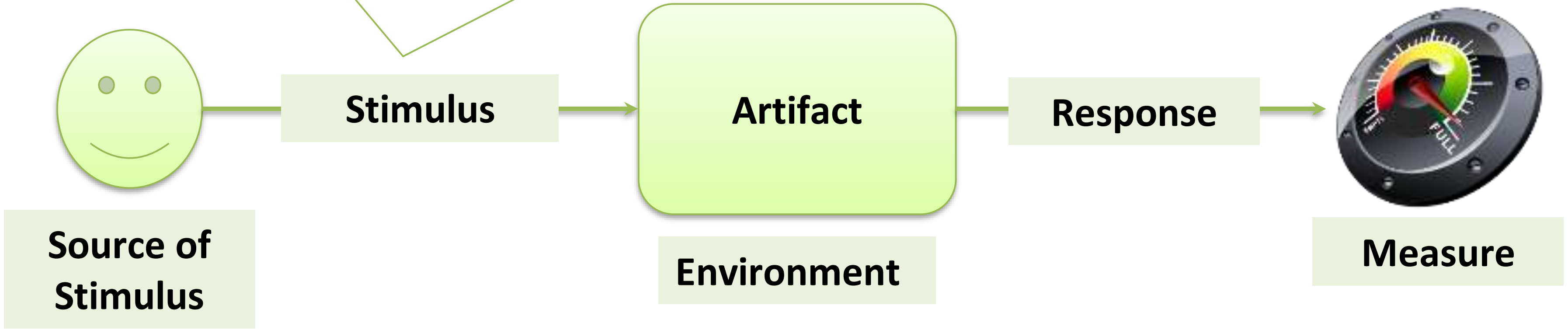
Modifiability Requirement Scenario

- who makes the change



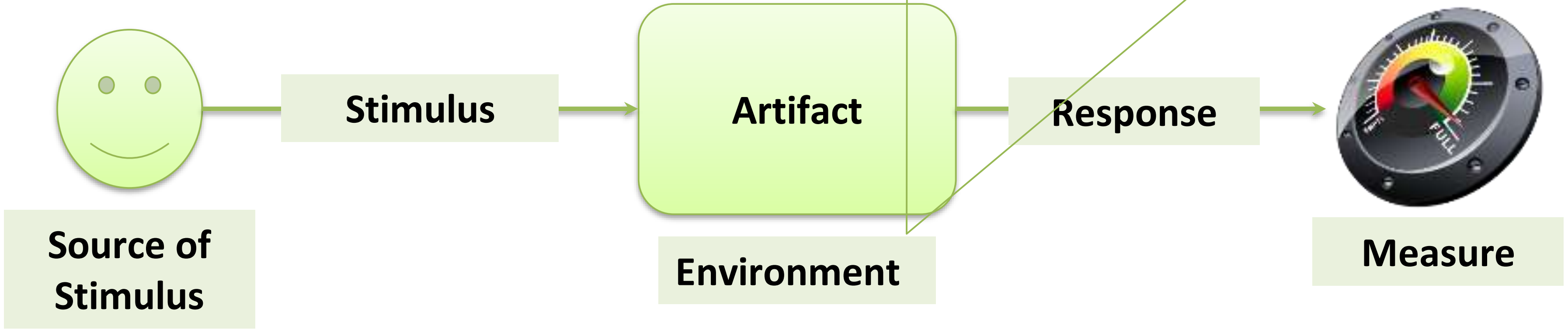
Modifiability Requirement Scenario

- the change



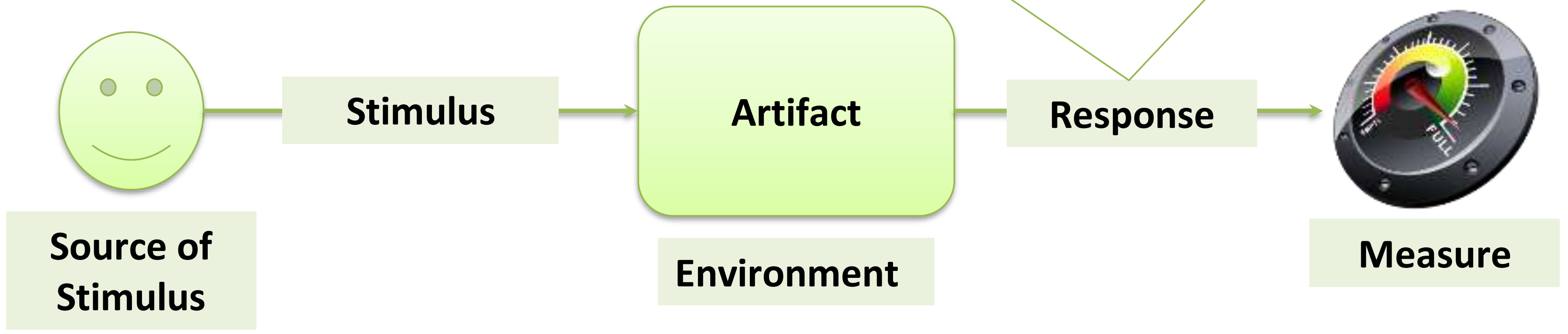
Modifiability Requirement Scenario

- when the change can be made



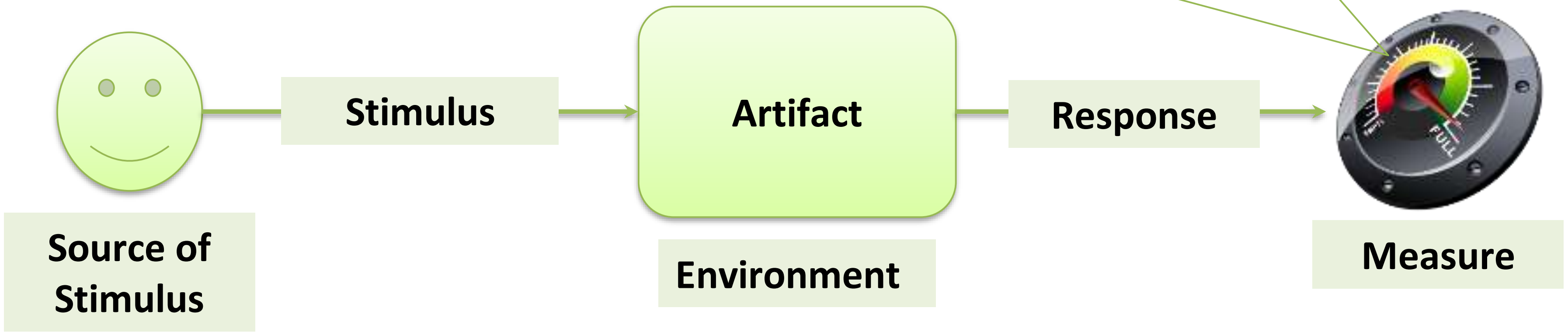
Modifiability Requirement Scenario

- make, test and deploy the change

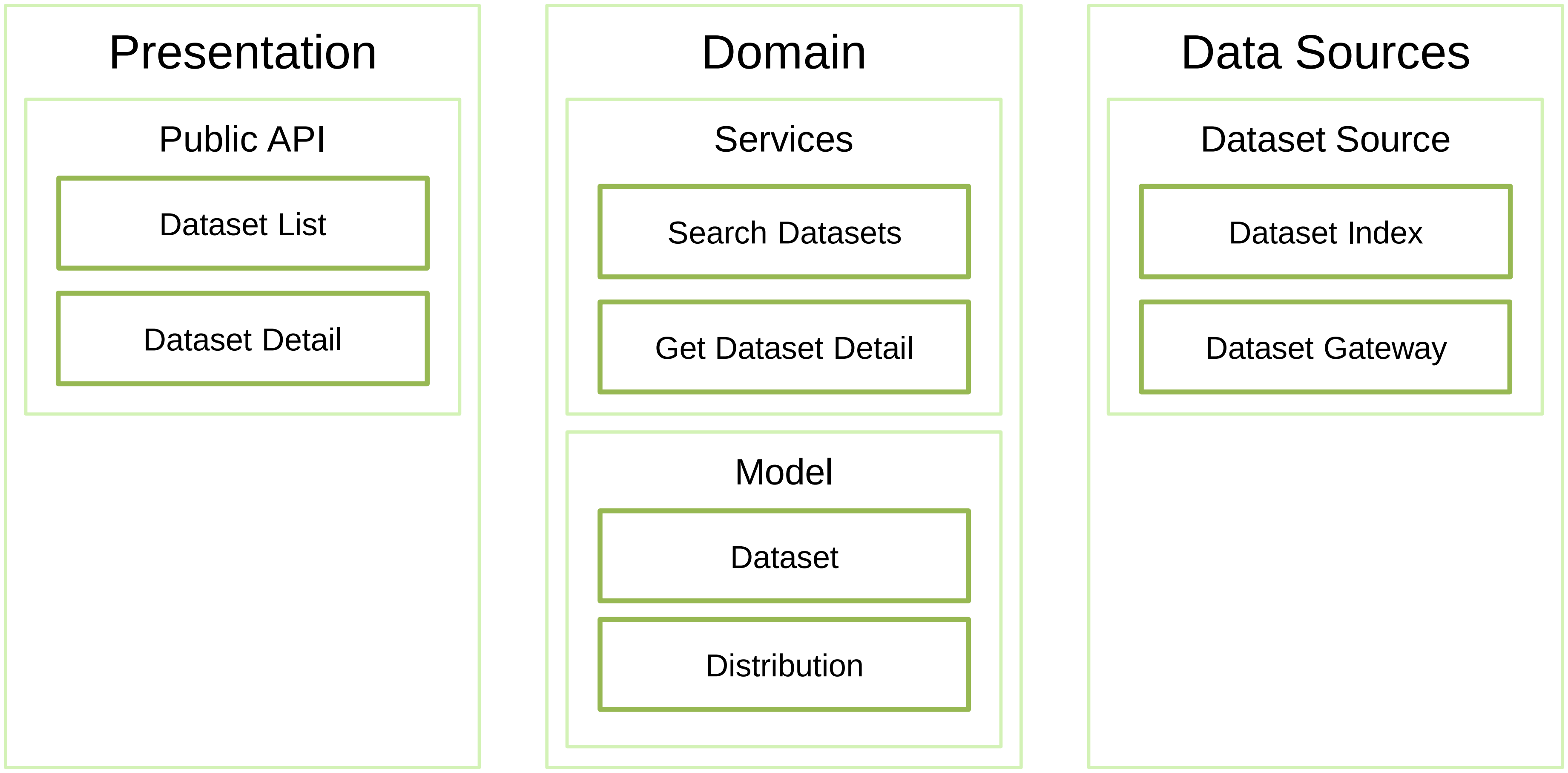


Modifiability Requirement Scenario

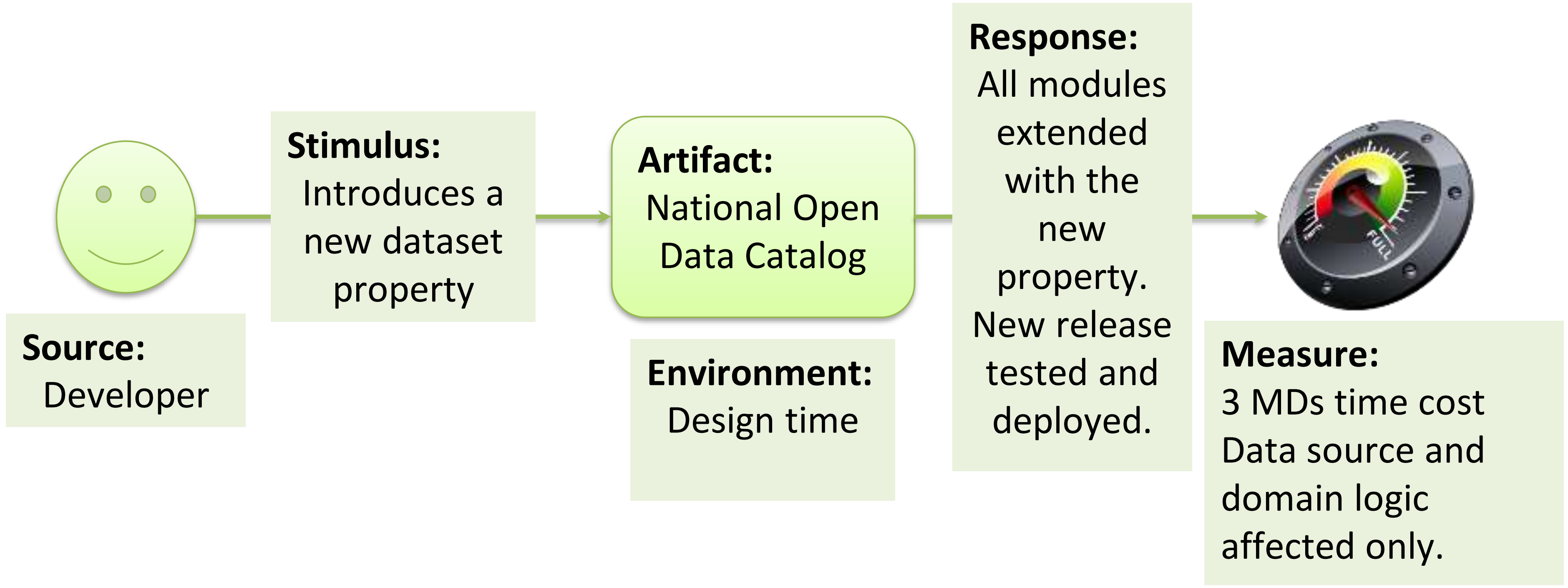
- cost in terms of time and money
- supplementary costs in terms of
  - number of affected places
  - new defects introduced



National Open Data Catalog



Modifiability Quality Attribute



Source:
Developer

Stimulus:
Introduces a new dataset property

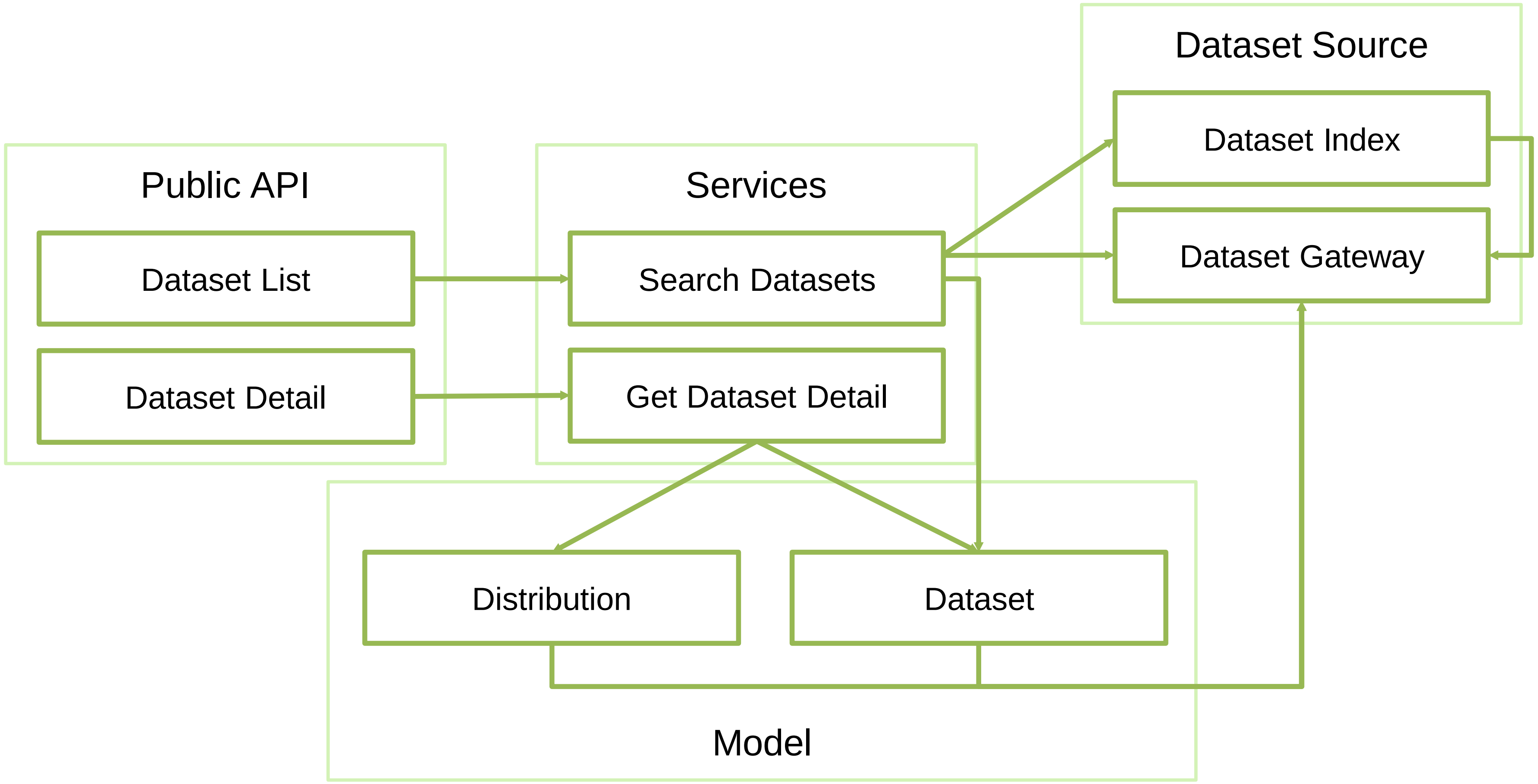
Artifact:
National Open Data Catalog

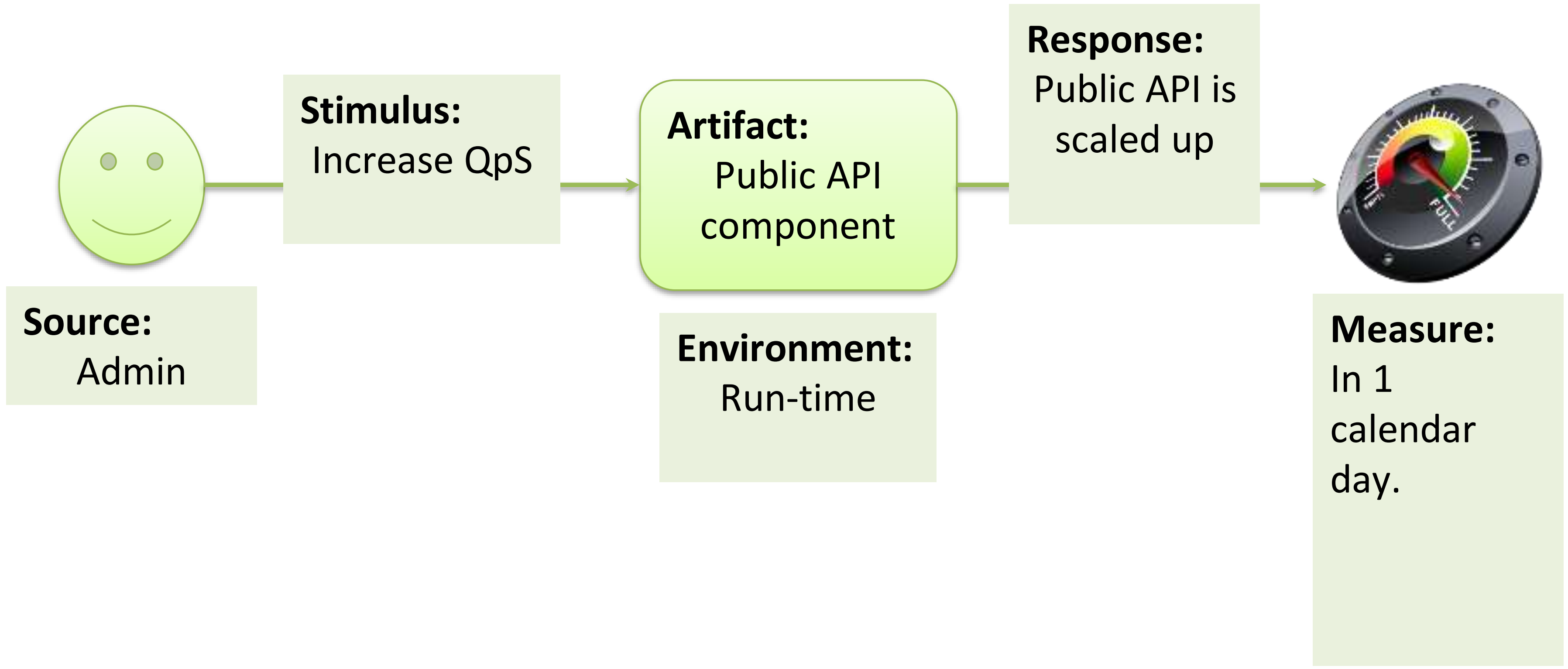
Environment:
Design time

Response:
All modules extended with the new property.
New release tested and deployed.

Measure:
3 MDs time cost
Data source and domain logic affected only.

uses →


Public API
Dataset List
Dataset Detail
Services
Search Datasets
Get Dataset Detail
Dataset Source
Dataset Index
Dataset Gateway
Distribution
Dataset
Model


Source: Admin
Stimulus: Increase QpS
Artifact: Public API component
Environment: Run-time
Response: Public API is scaled up
Measure: In 1 calendar day.

Module responsibility

Module coupling

Module cohesion

Responsibility

- each module in software architecture is responsible for some functionality, part of functionality or quality

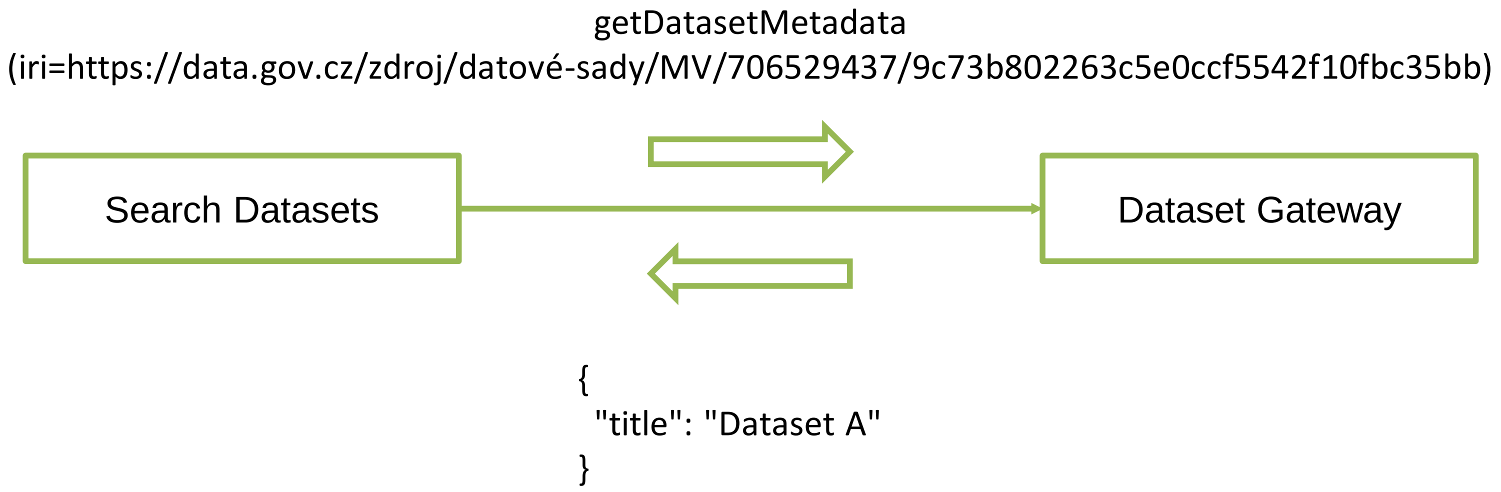
Coupling

- coupling is measure of how modules overlap in their responsibilities
  - more overlap => more coupling => more dependencies

loose coupling vs. tight coupling

Kinds of dependencies

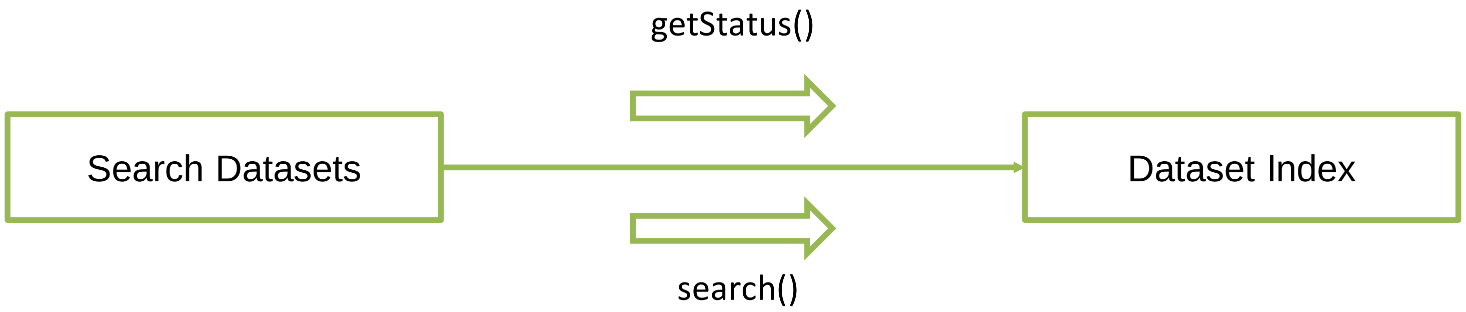
- data dependencies


getDatasetMetadata
(iri=https://data.gov.cz/zdroj/datové-sady/MV/706529437/9c73b802263c5e0ccf5542f10fbc35bb)
Search Datasets
Dataset Gateway

```
{
  "title": "Dataset A"
}
```

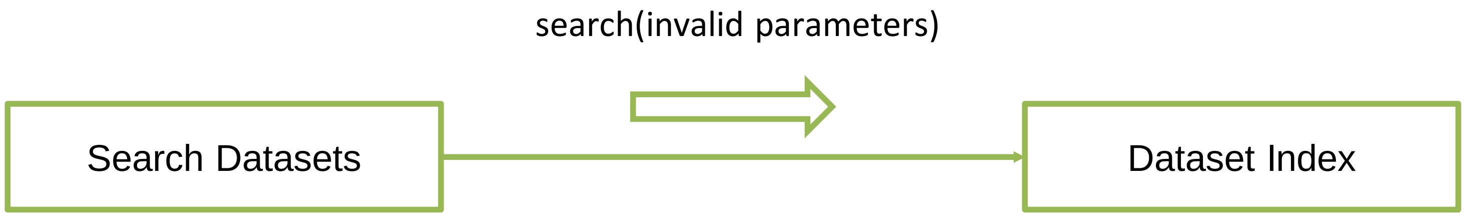
Kinds of dependencies

- control dependencies



Kinds of dependencies

- content dependency



Kinds of dependencies

- quality dependency
- existence dependency

Cohesion

- cohesion is measure of how responsibilities of a given module belong together

low cohesion vs. high cohesion

Kinds of Cohesion

- coincidental cohesion

Kinds of Cohesion

- logical cohesion

Kinds of Cohesion

- temporal cohesion

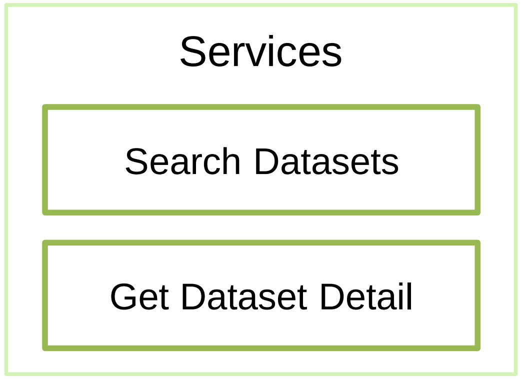
Local Open Data
Catalog Harvestor

Kinds of Cohesion

- procedural cohesion
- informational cohesion
- sequential cohesion

Kinds of Cohesion

- functional cohesion



Cohesion

Low High

+

Coupling

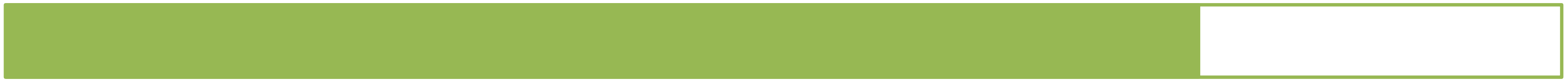
Tight Loose

⍰

Modifiability

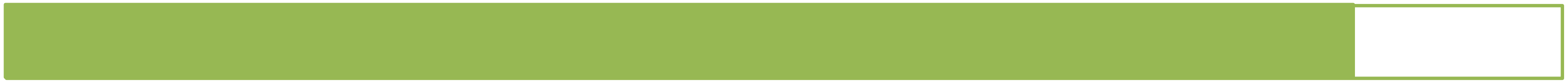
Low High

Cohesion



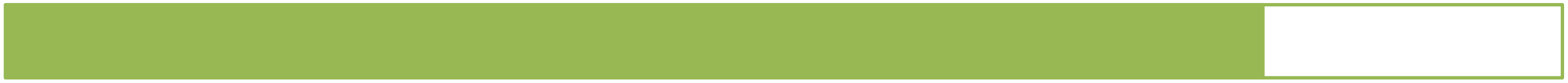
+

Coupling



?

Modifiability



space

⍰

time

⍰

entropy

⍰

software entropy

⍰

low cohesion & tight coupling

Modifiability Tactics

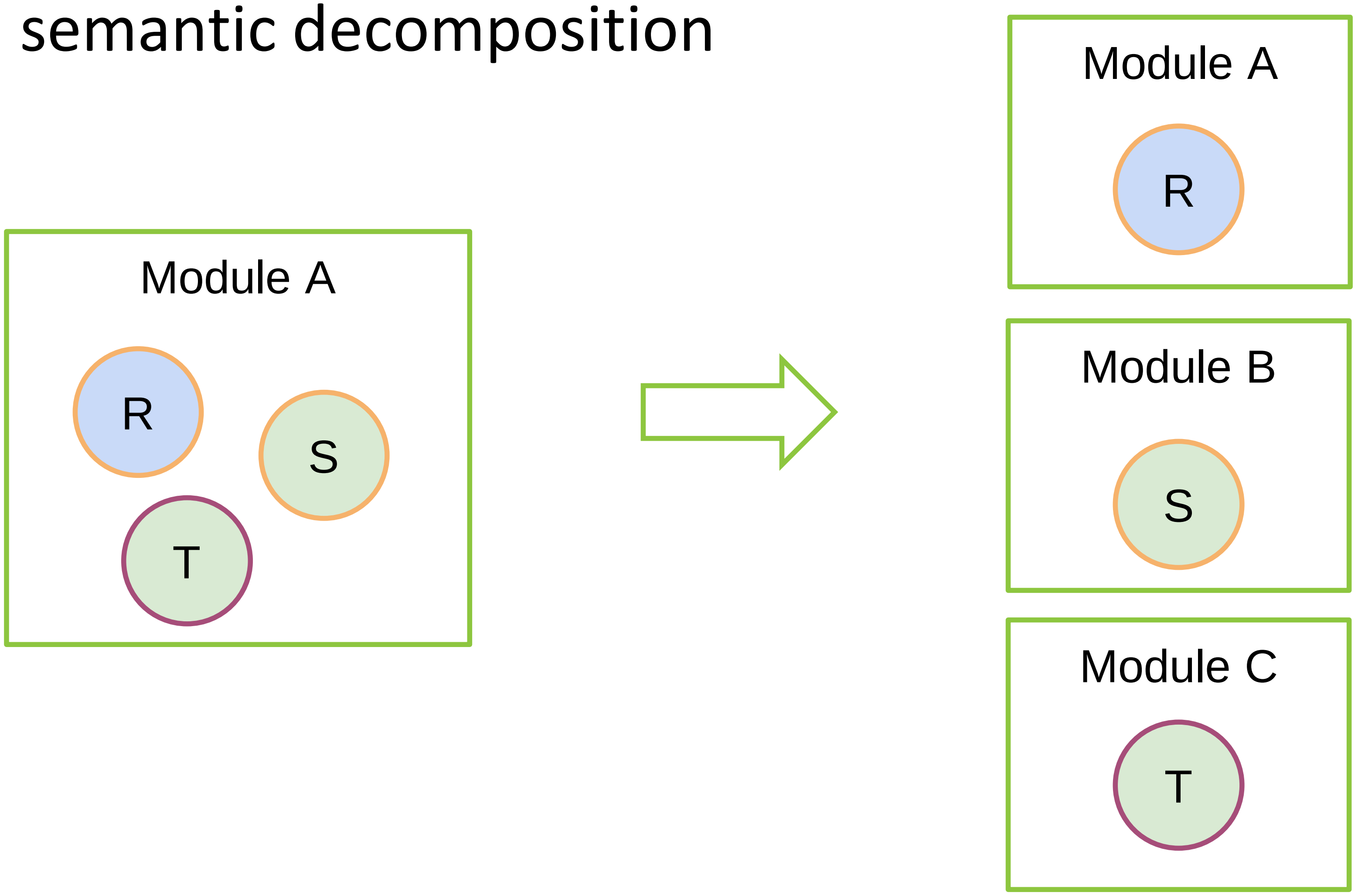
- increase cohesion
- reduce coupling
- defer binding

Increase cohesion

- moving responsibilities from one module to another
- easier identification of modules to be changed
- likelihood of side effects inside changed modules

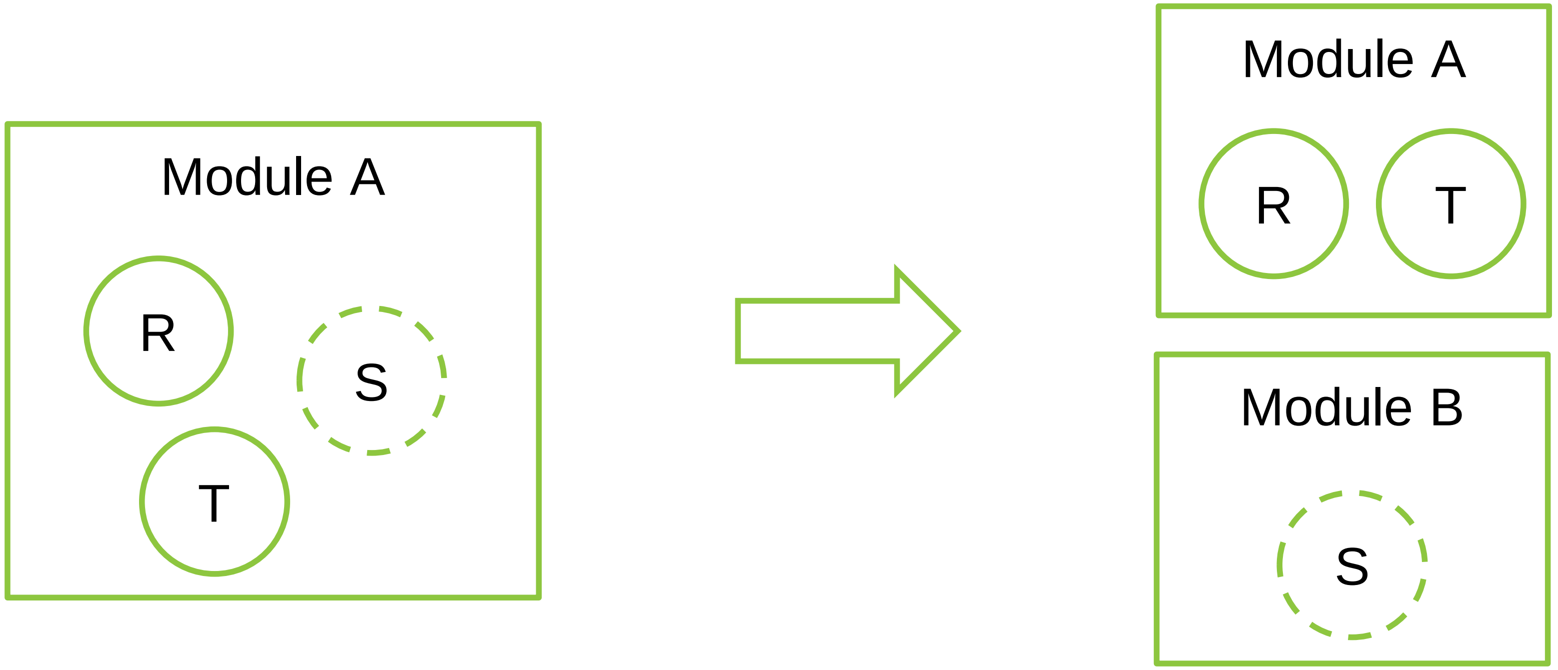
Increase cohesion

- semantic decomposition



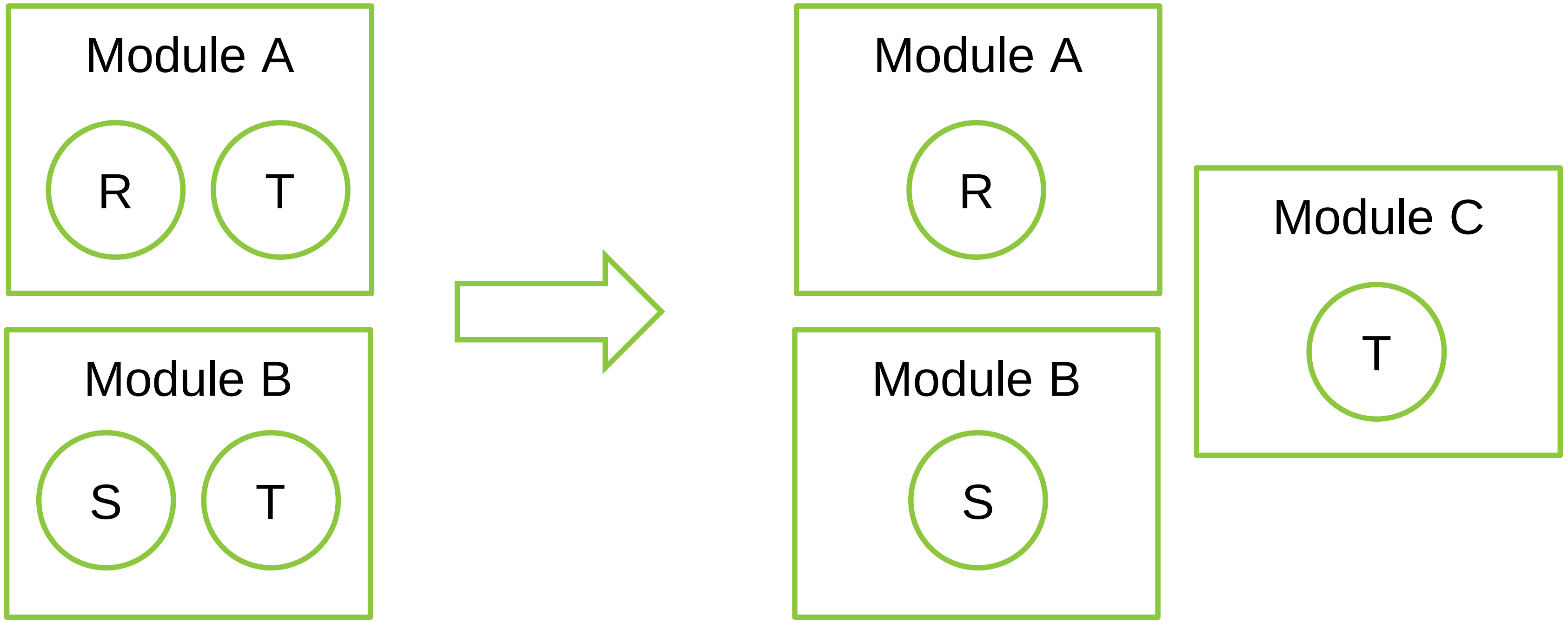
Increase cohesion

- decomposition based on anticipated changes



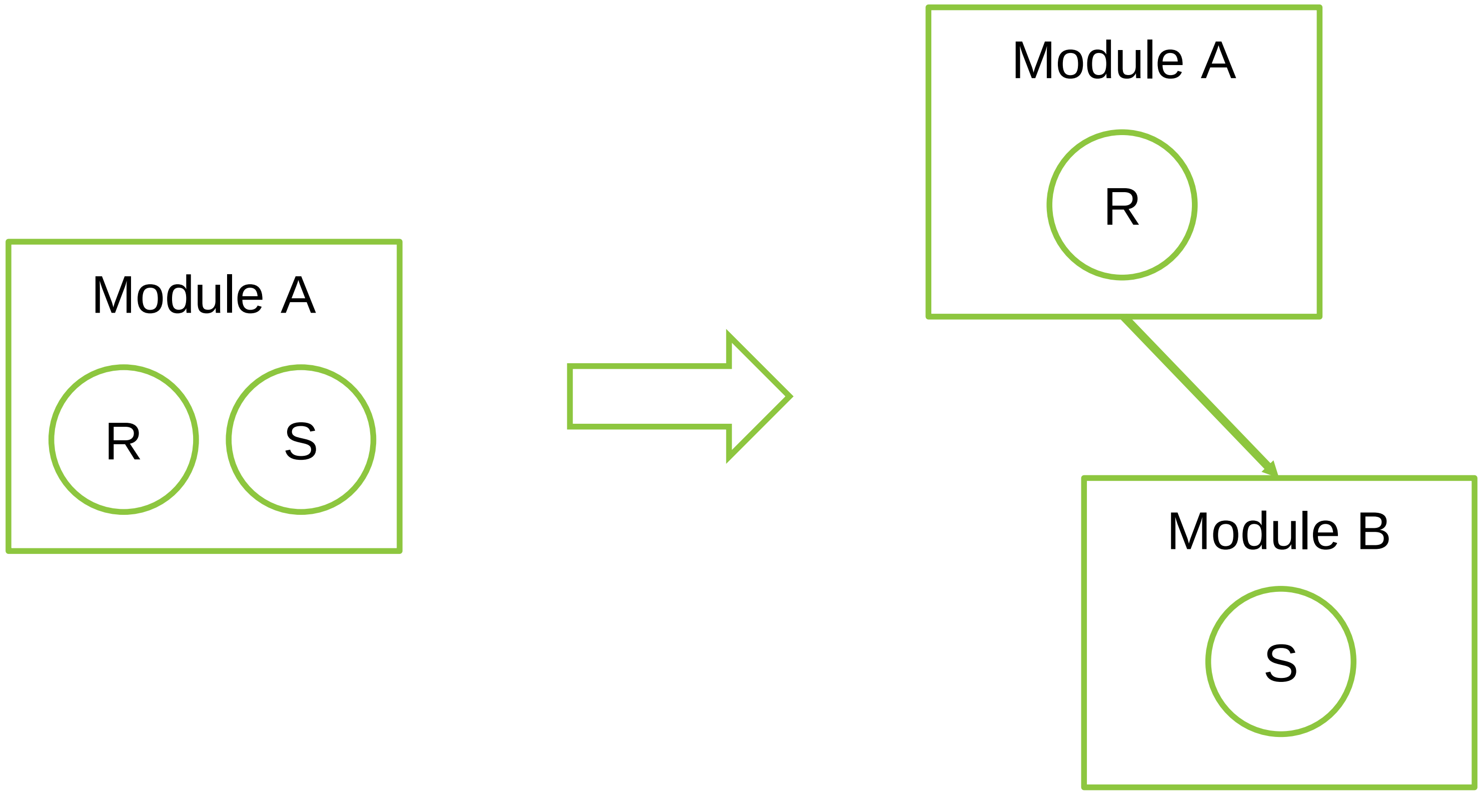
Increase cohesion

- decomposition based on shared responsibilities



Impact of increasing cohesion

- increasing cohesion may create dependencies



Reduce coupling

- goal is to prevent from ripple effect
  - module is modified only because of the modification of another module
  - prevent from big-bowl-of-mud or spaghetti

Reduce coupling

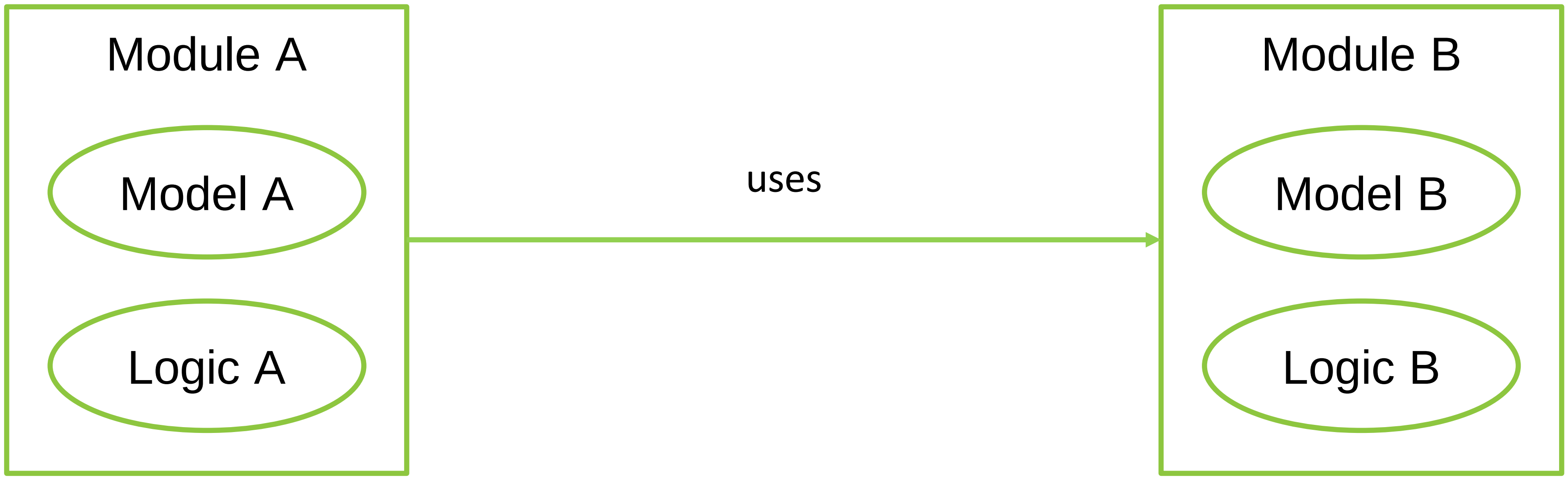
- restrict dependencies
  - restricts modules a given module can use
  - e.g., layered architecture
- information hiding / encapsulation
  - module has public interface which exposes only public responsibilities and hides private ones

Reduce coupling

- intermediary translator
  - module with translation responsibility
  - breaks dependency

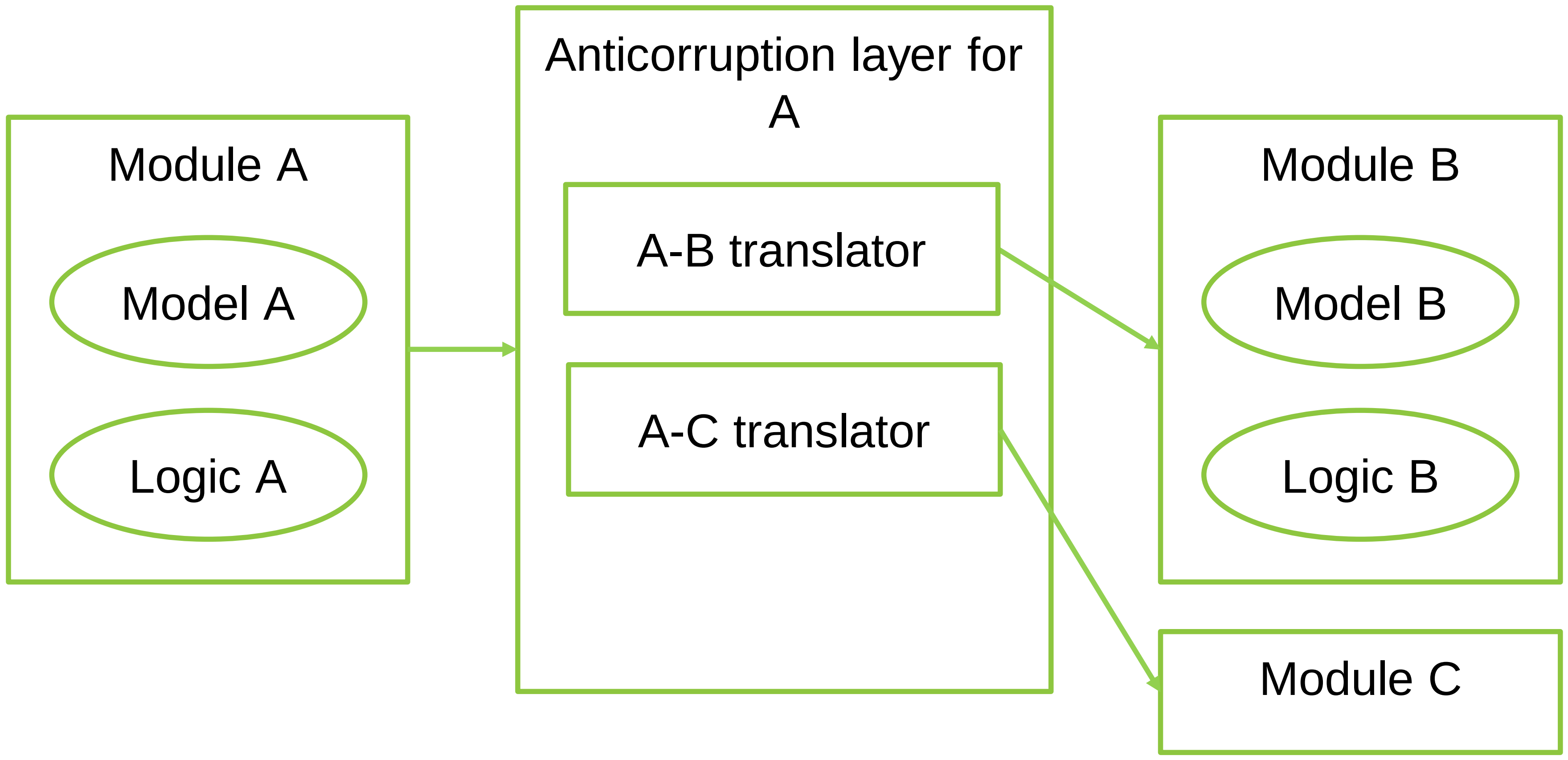
Reduce coupling

- intermediary translator - anticorruption layer



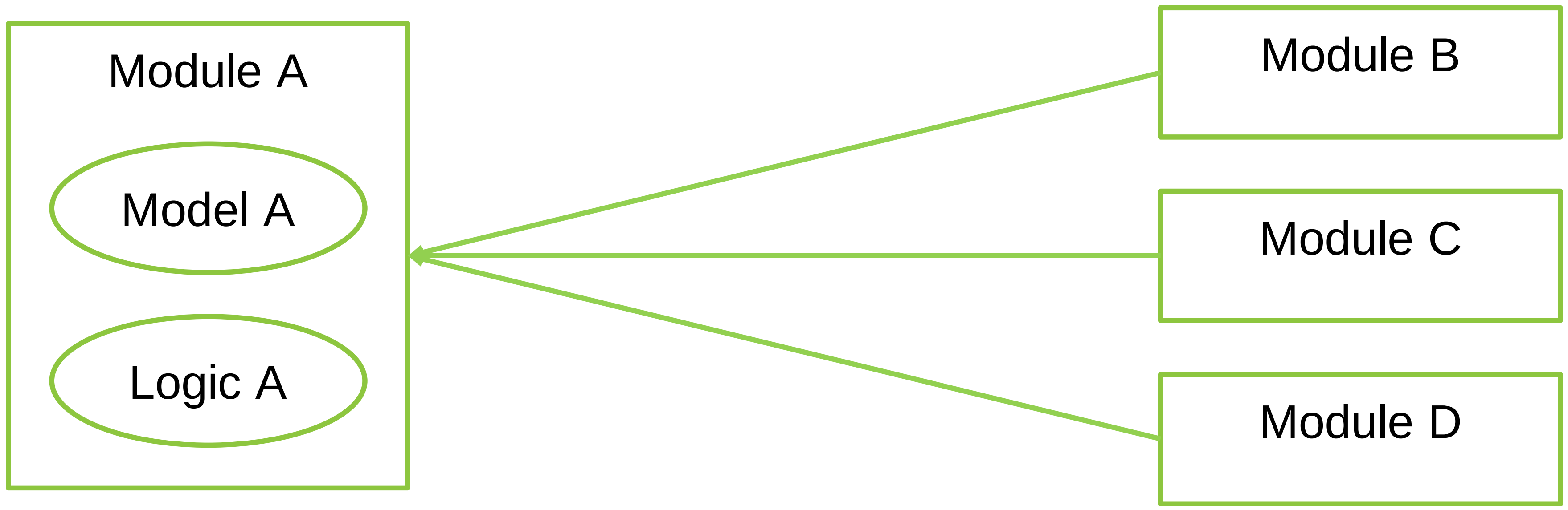
Reduce coupling

- intermediary translator - anticorruption layer



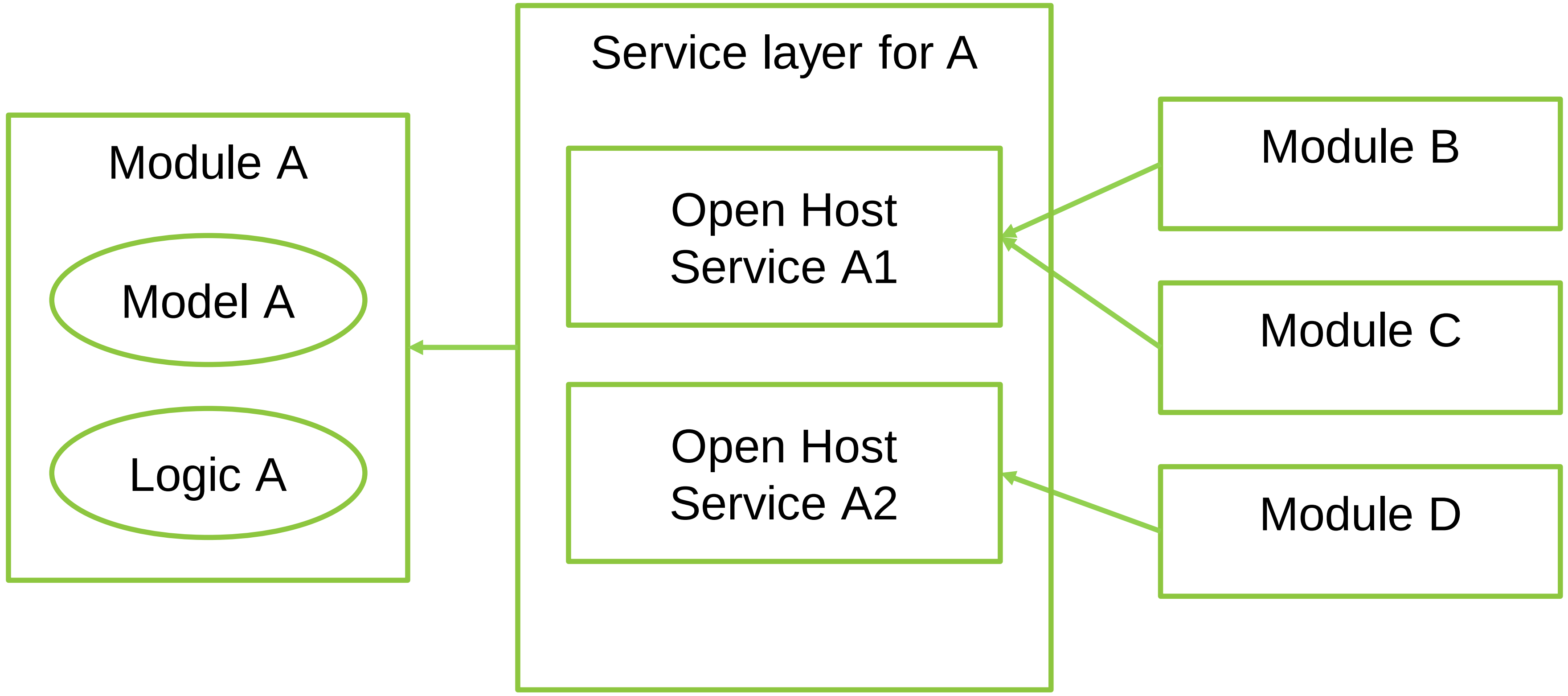
Reduce coupling

- intermediary translator - open host service



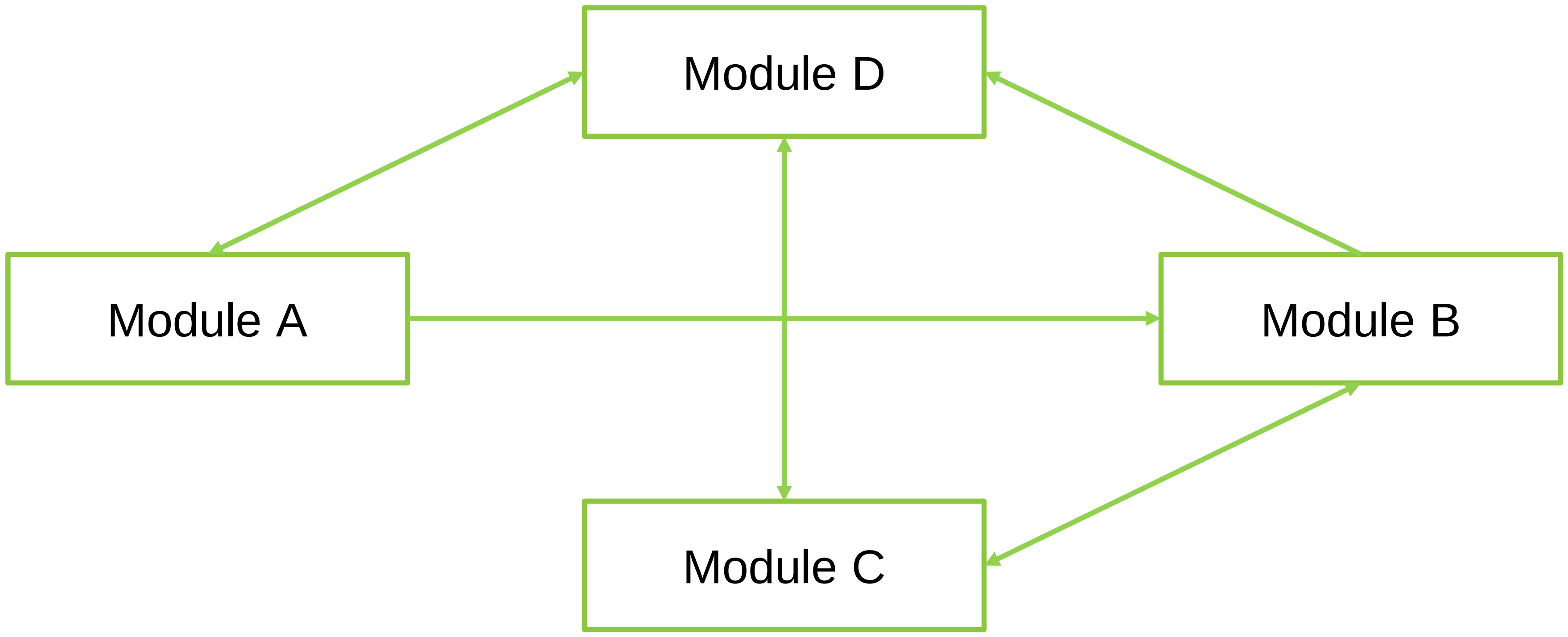
Reduce coupling

- intermediary translator - open host service



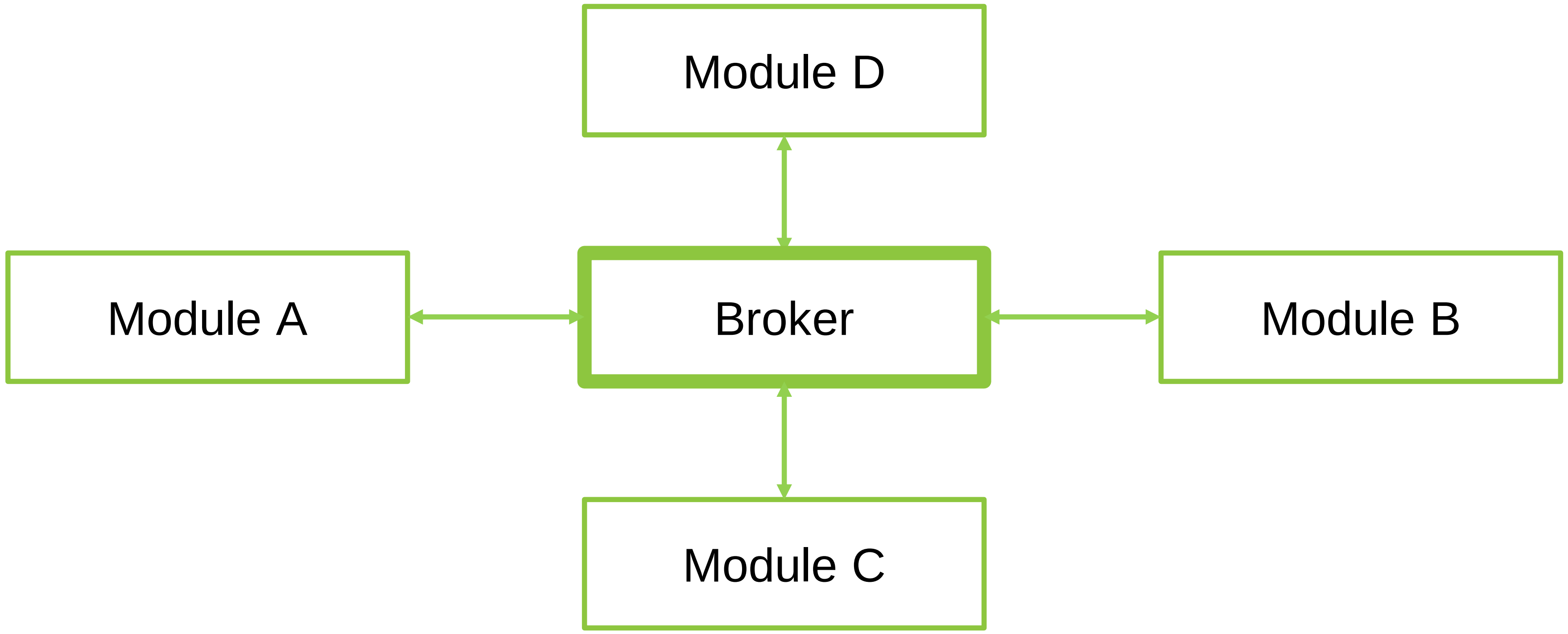
Reduce coupling

- intermediary translator - broker



Reduce coupling

- intermediary translator - broker

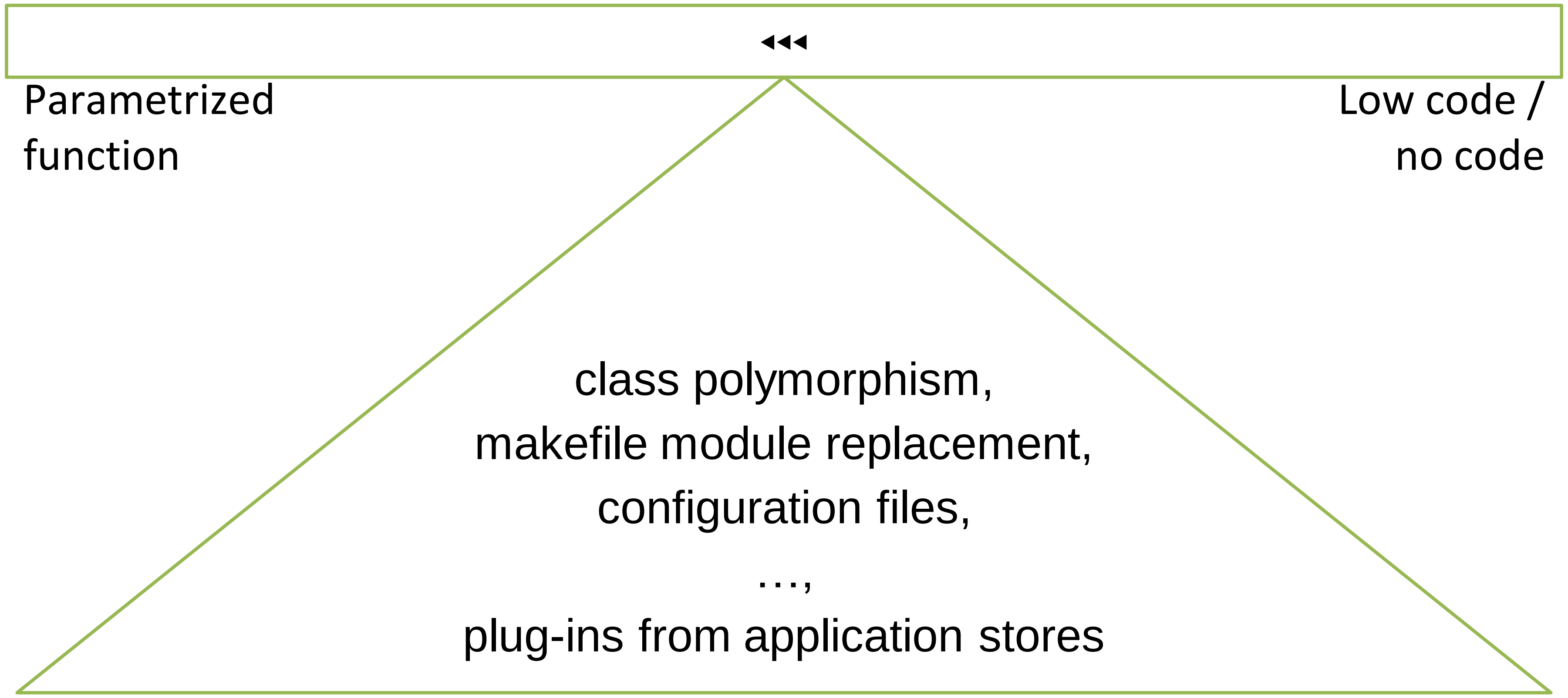


Reduce coupling

- preservation
  - versioning, preserving old version
  - stubs
- refactoring

Defer binding

- let computers handle changes as much as possible


◂◂◂
Parametrized function
Low code / no code
class polymorphism,
makefile module replacement,
configuration files,
…,
plug-ins from application stores