Automating Architectural Governance

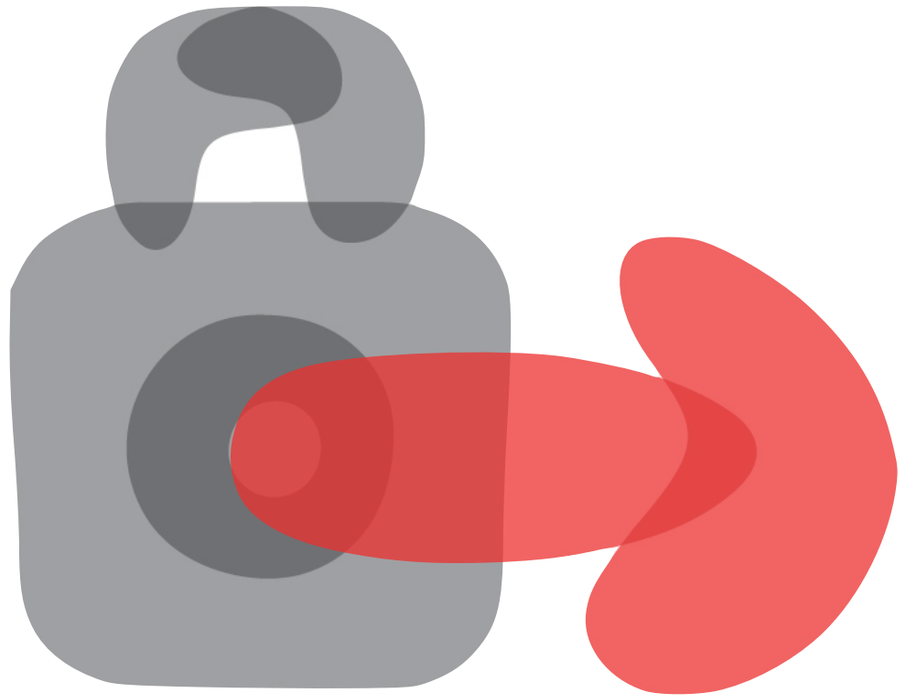
ThoughtWorks®

NEAL FORD

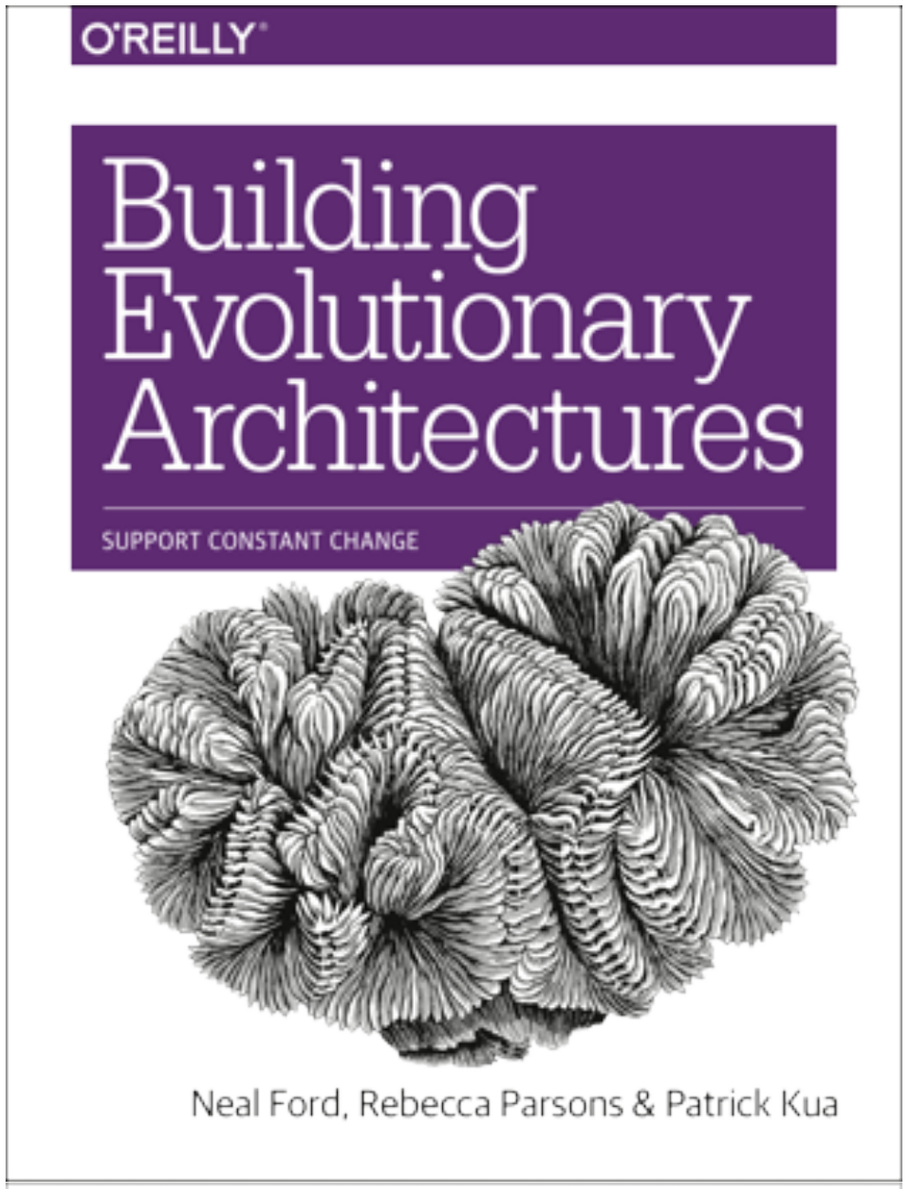
Director / Software Architect / Meme Wrangler

@neal4d

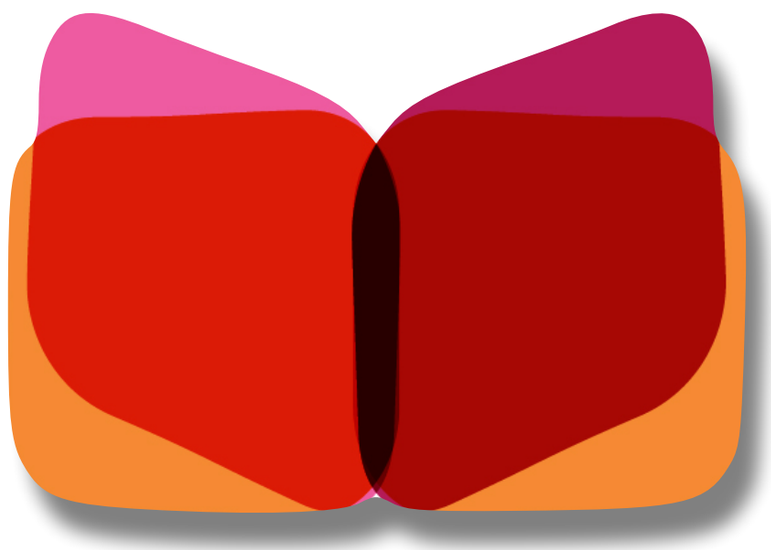
http://nealford.com



2019-11-21


O'REILLY
Building Evolutionary Architectures
SUPPORT CONSTANT CHANGE
Neal Ford, Rebecca Parsons & Patrick Kua

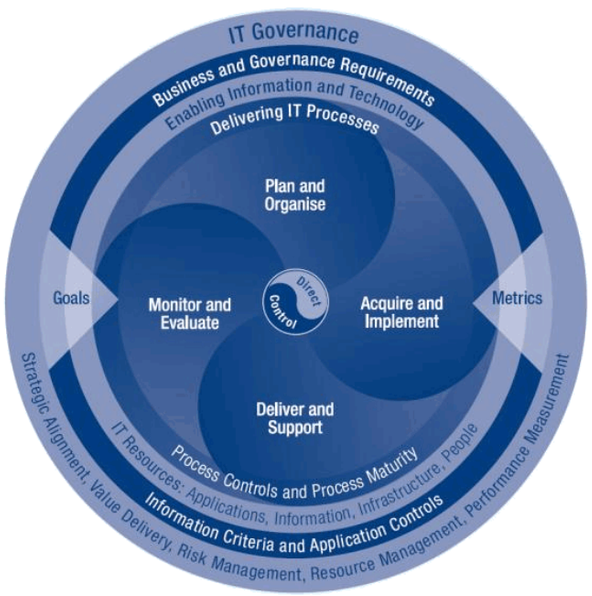
Defining Governance

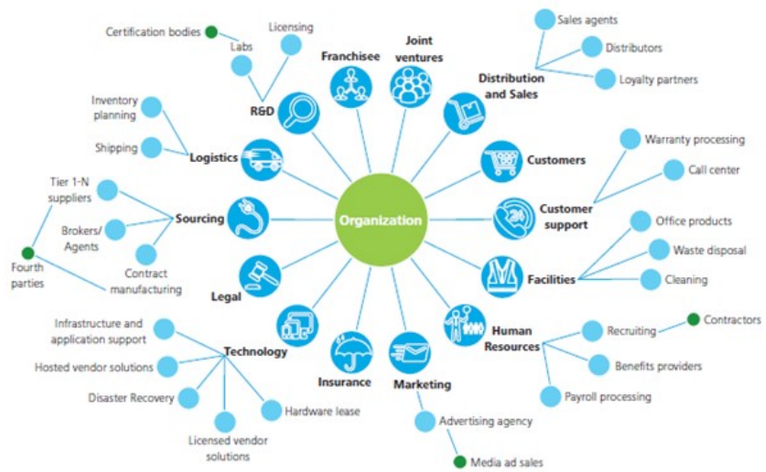


Defining Governance

www.isaca.org/Knowledge-Center/COBIT/Pages/Overview.aspx

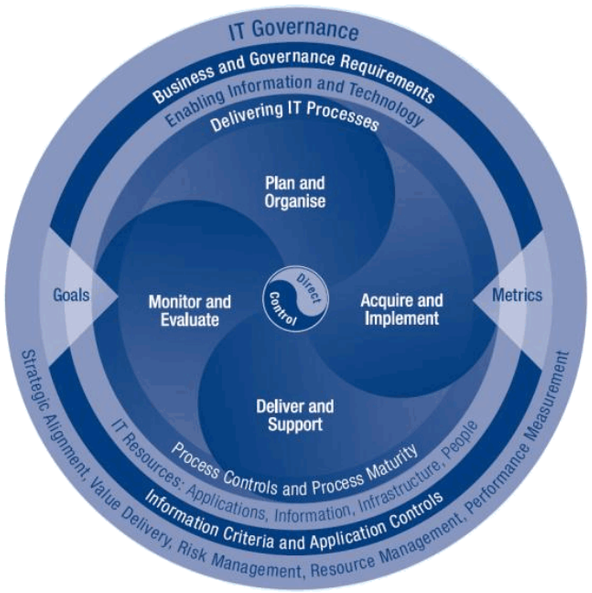
Defining Governance

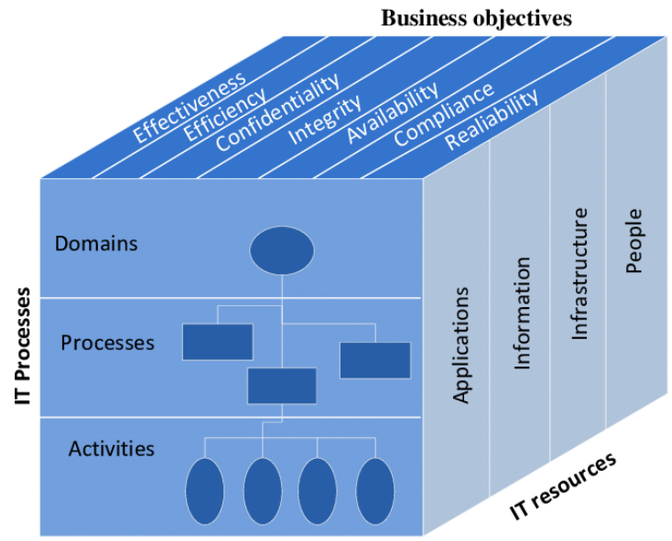




www.isaca.org/Knowledge-Center/COBIT/Pages/Overview.aspx

Defining Governance

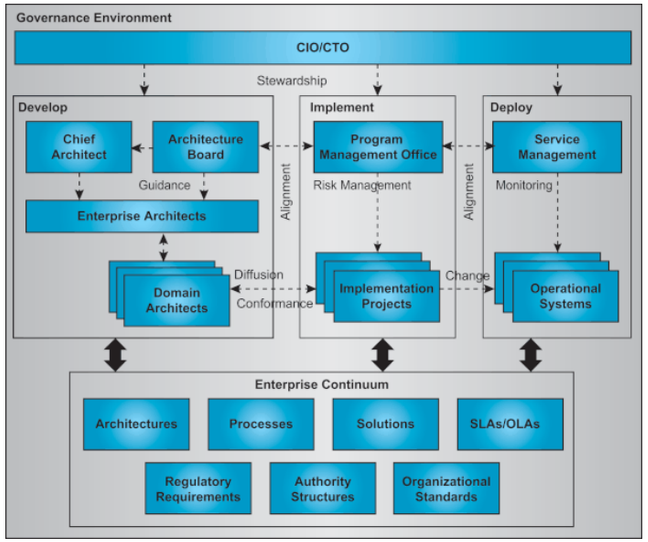


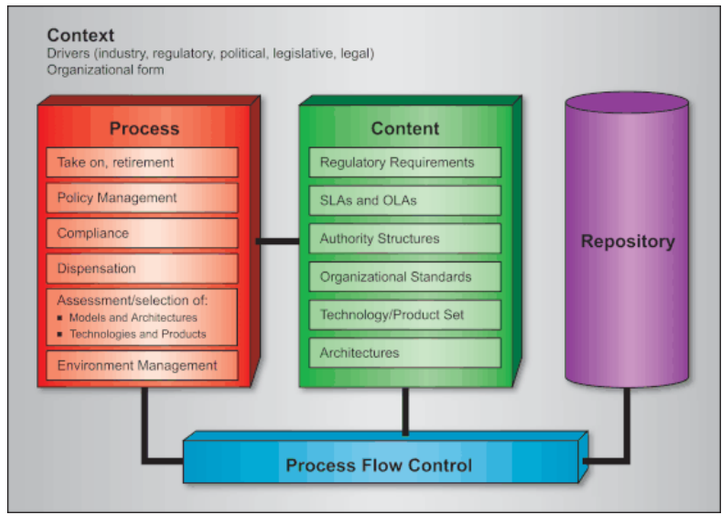


www.isaca.org/Knowledge-Center/COBIT/Pages/Overview.aspx

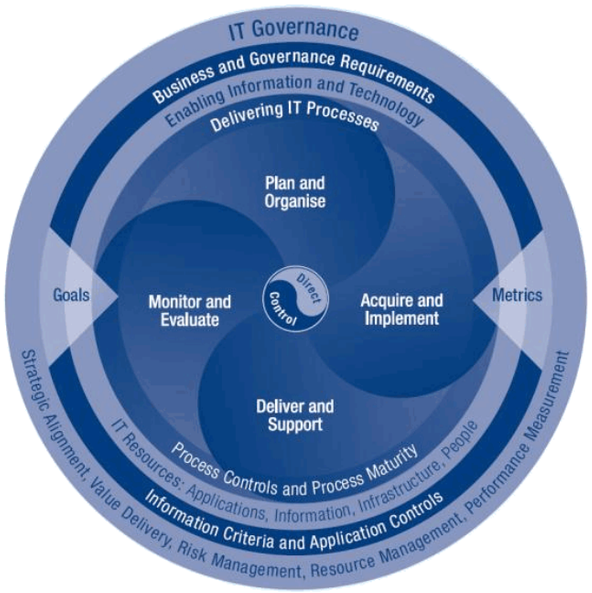
www.researchgate.net/figure/The-COBIT-cube_fig5_224162993

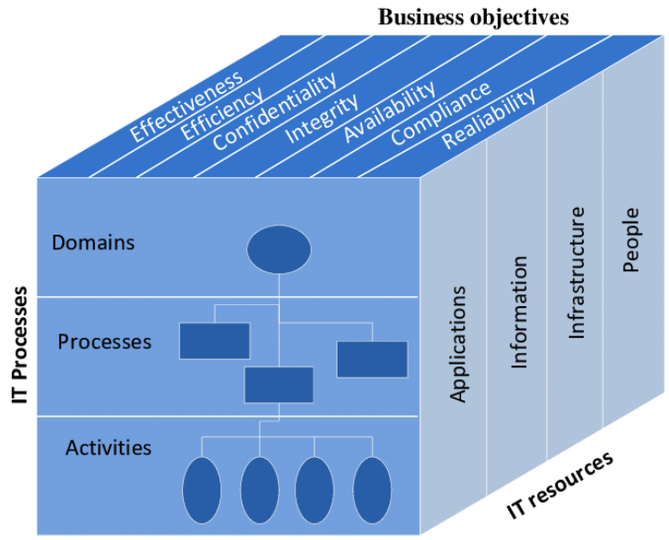
Defining Governance





pubs.opengroup.org/architecture/togaf8-doc/arch/chap26.html

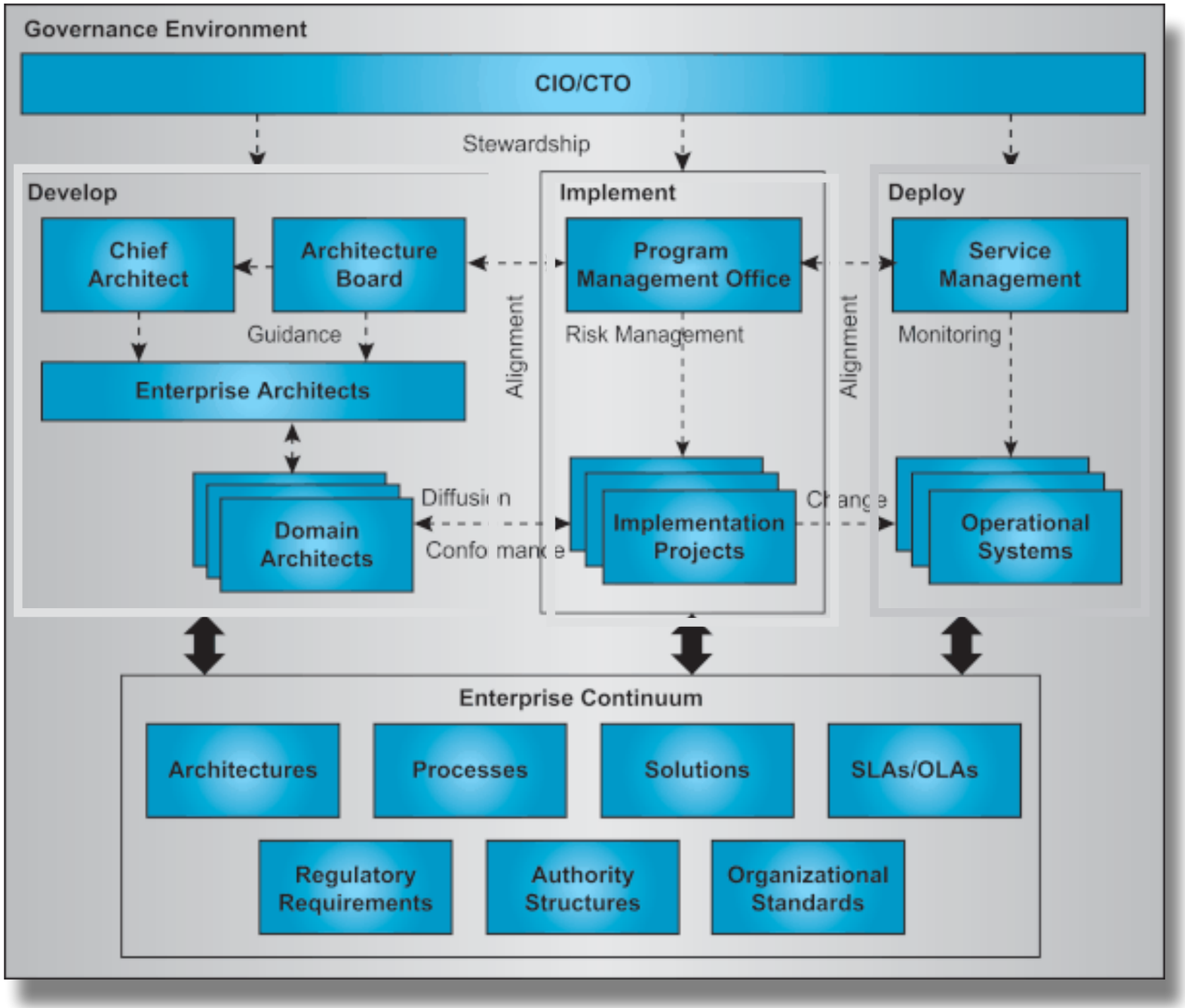




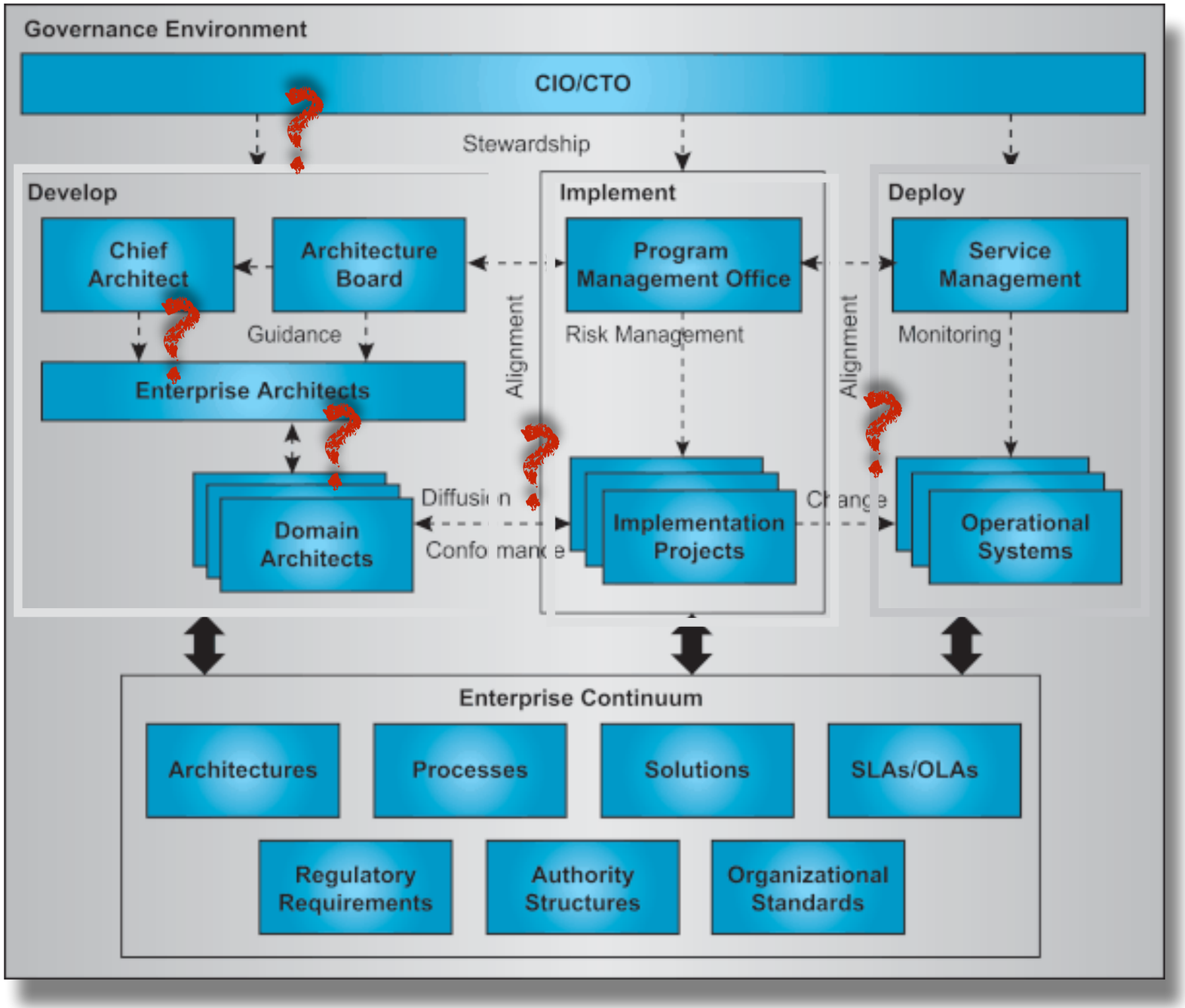
www.isaca.org/Knowledge-Center/COBIT/Pages/Overview.aspx

www.researchgate.net/figure/The-COBIT-cube_fig5_224162993

Defining Governance

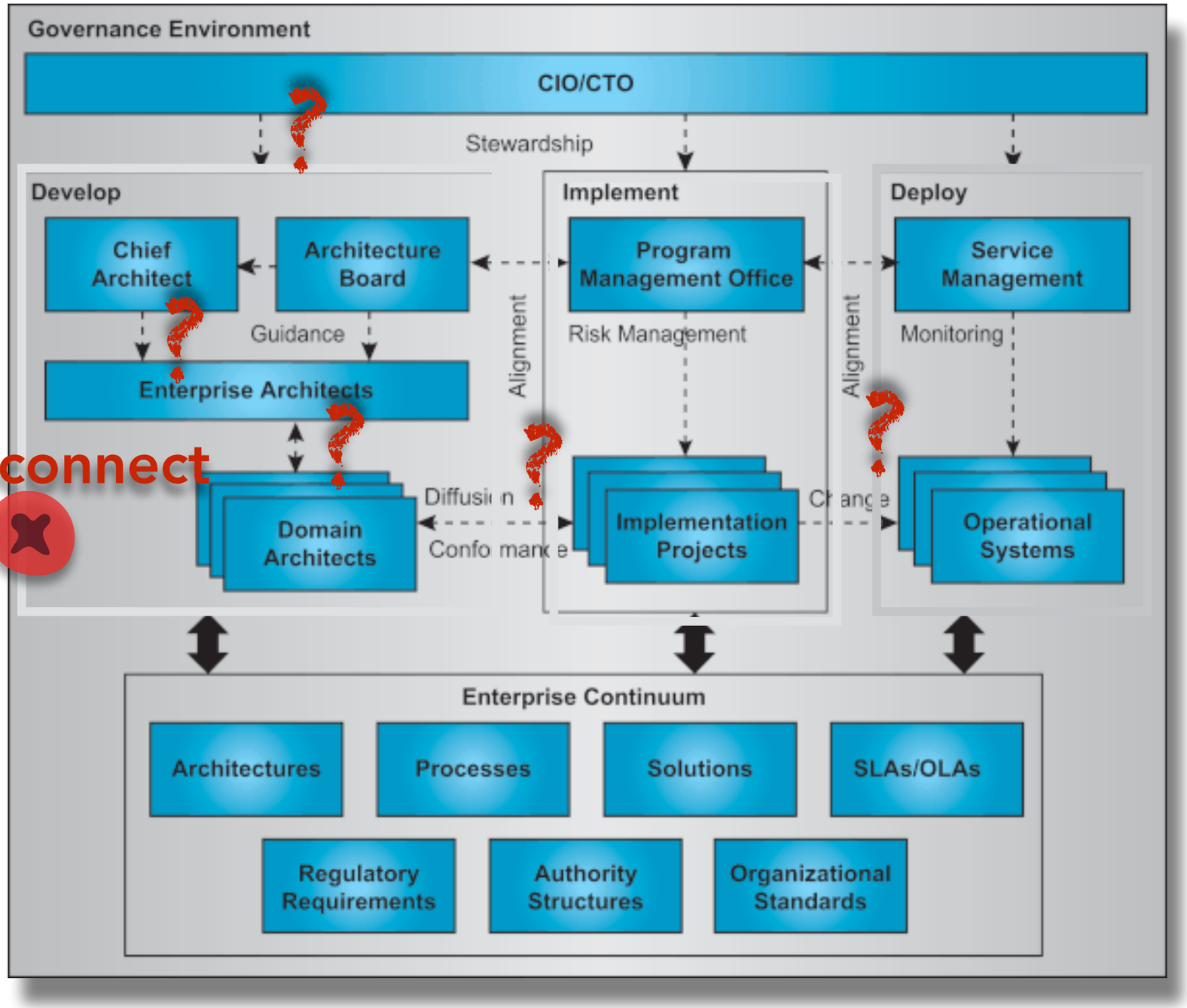


Defining Governance



Defining Governance

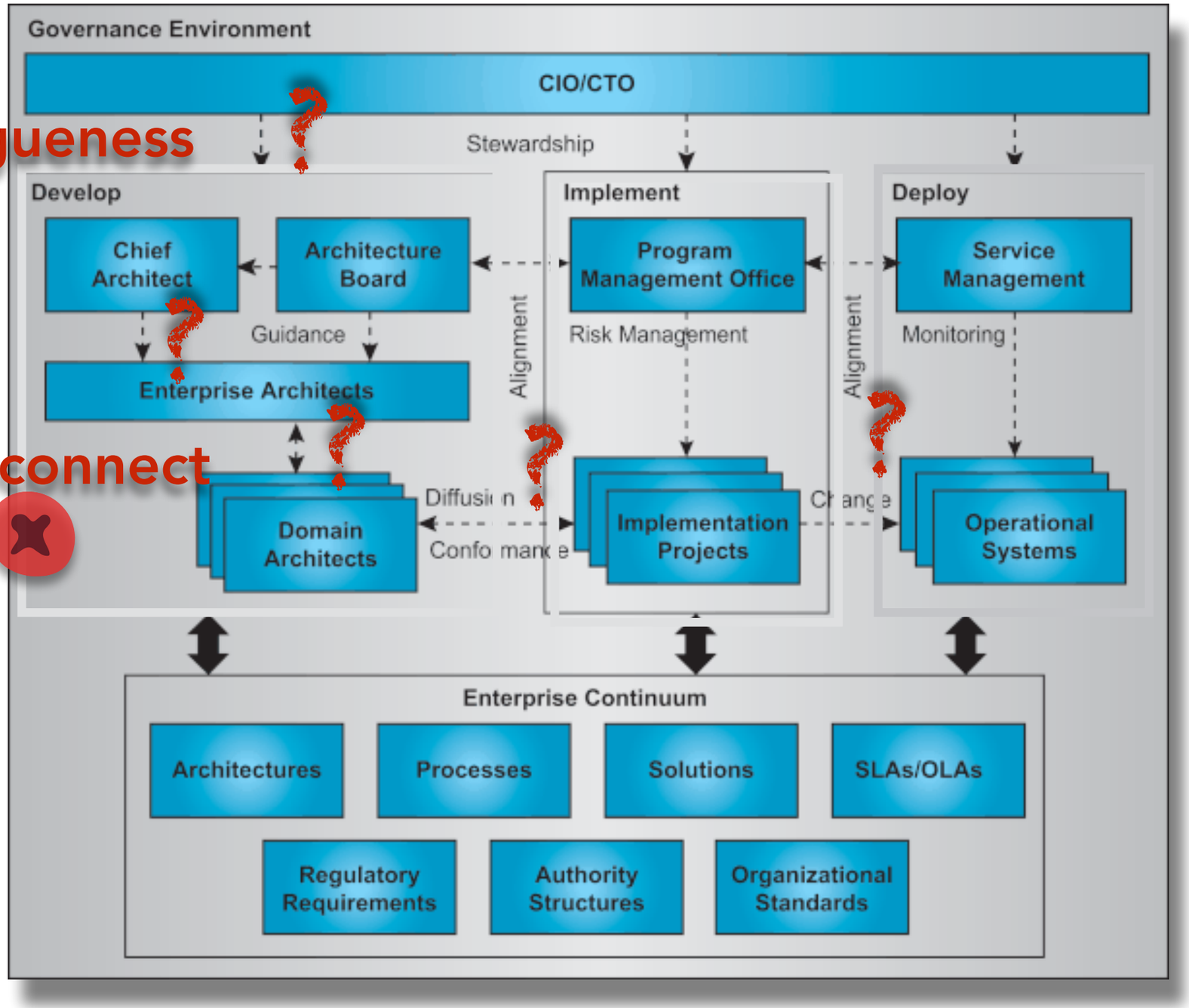
1. disconnect



Defining Governance

2. vagueness

1. disconnect



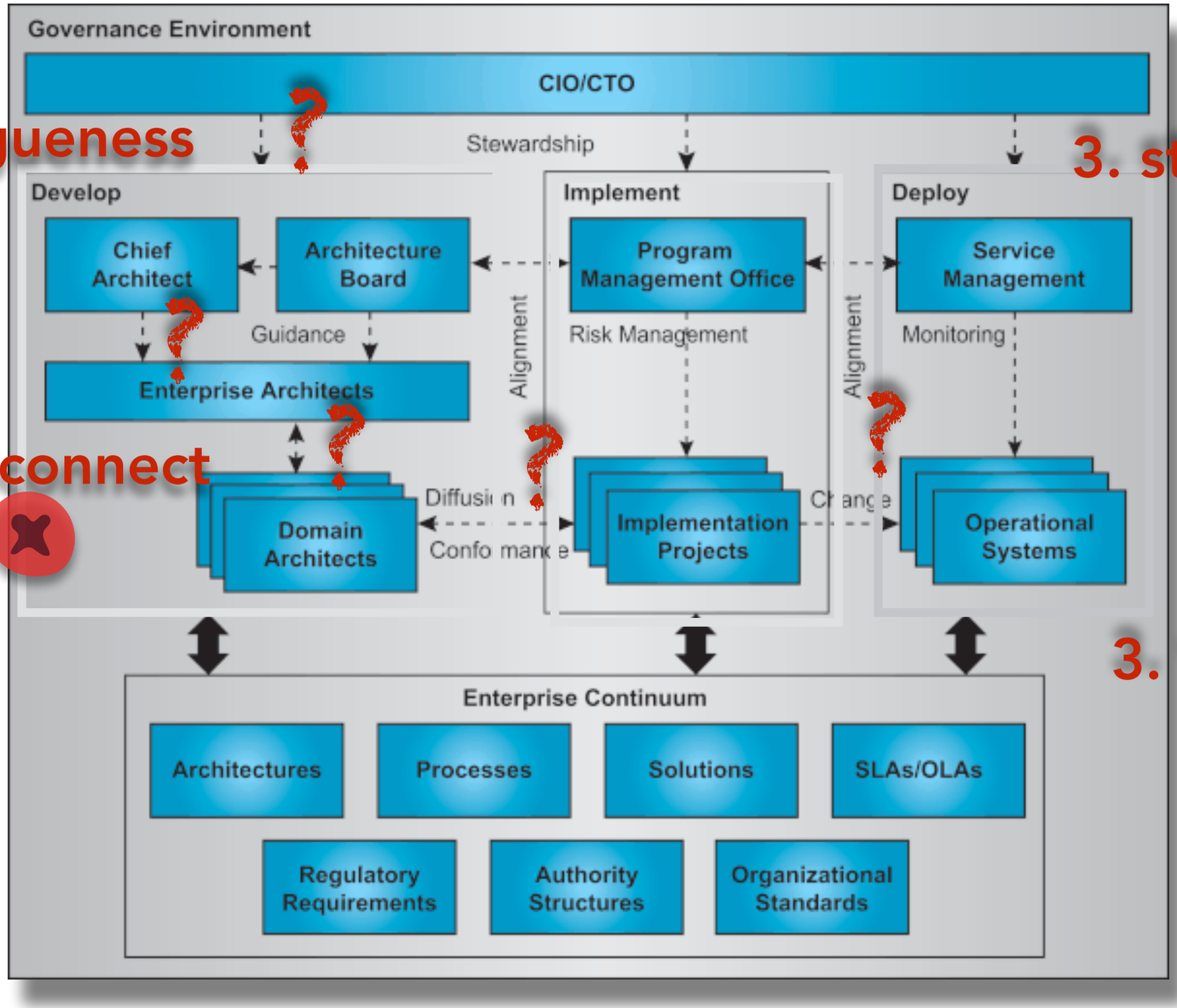
Defining Governance

2. vagueness

3. strategy...

1. disconnect

3. vs...tactics



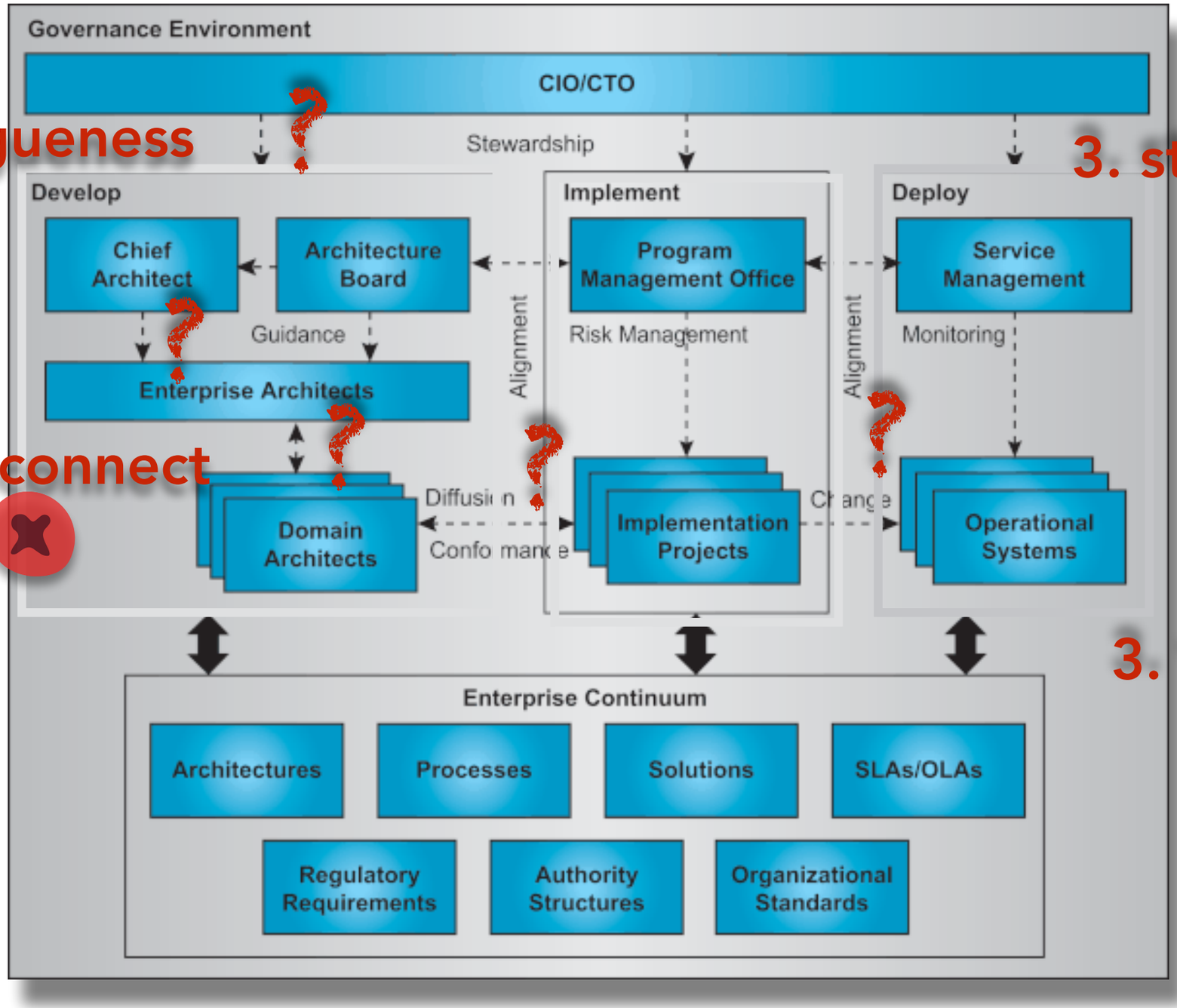
Defining Governance

2. vagueness

3. strategy...

1. disconnect

3. vs...tactics

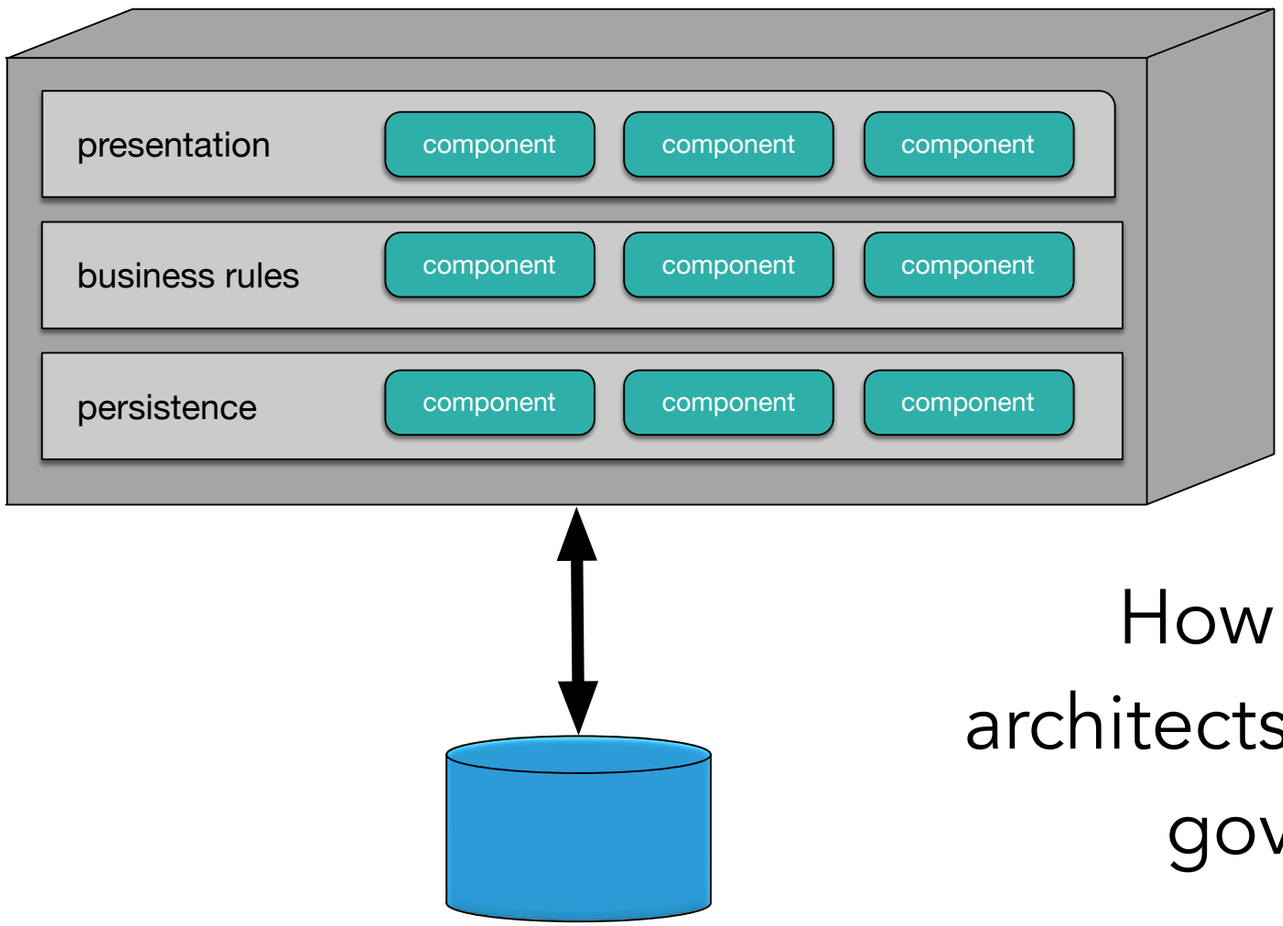


craft versus engineering

question:

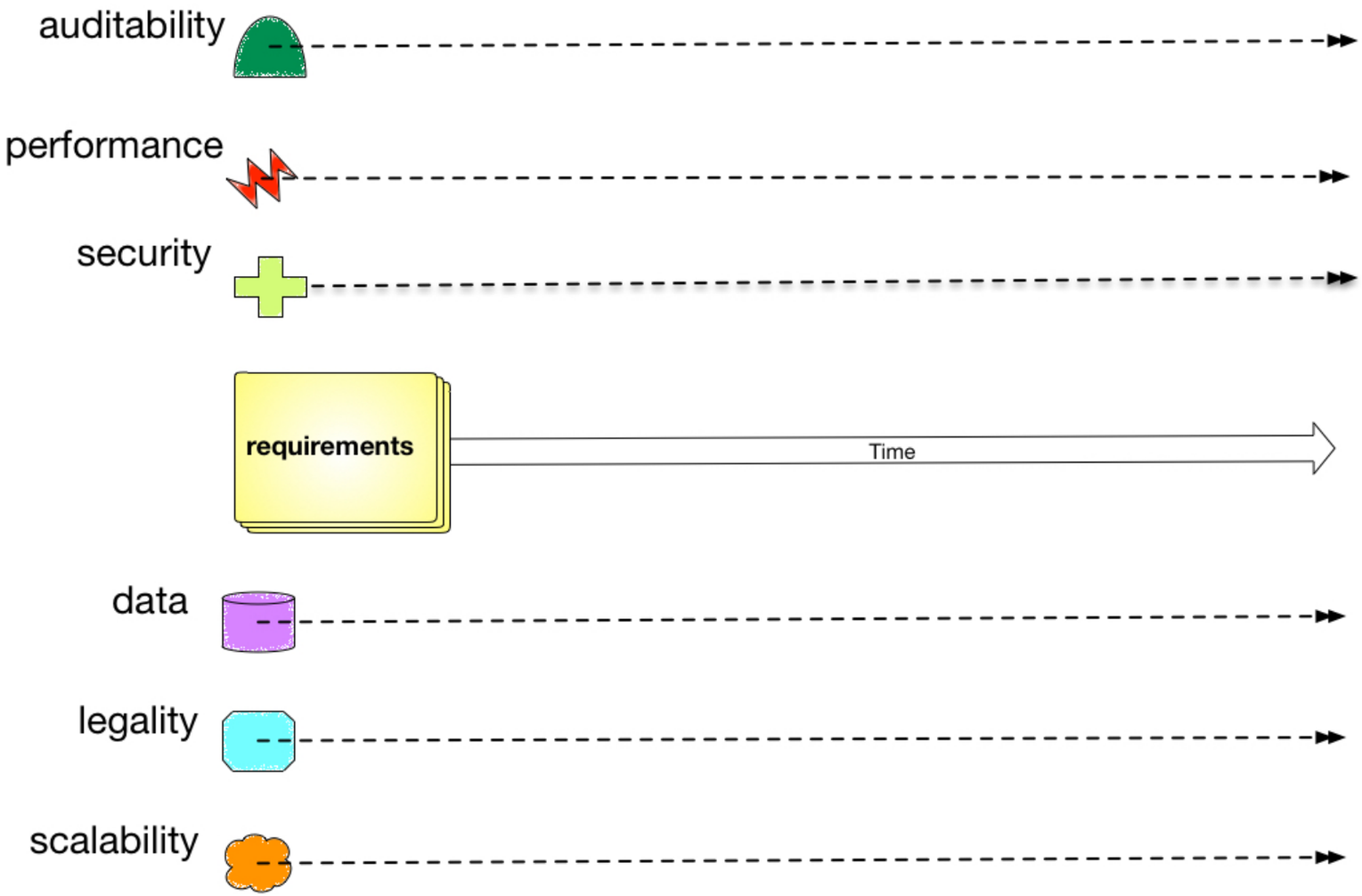
How can enterprise architects/architects/developers concretely define governance for architecture characteristics?

question:



How can enterprise architects/ architects/developers concretely define governance for architecture characteristics?

=> Evolutionary Architecture



Evolutionary Architecture

An evolutionary architecture supports
guided,
incremental change
across multiple dimensions.

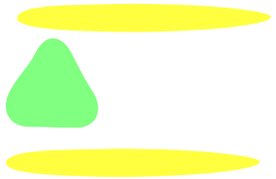
Evolutionary Architecture

An evolutionary architecture supports

guided

incremental change

across multiple dimensions.



guided

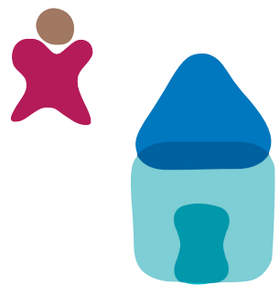
evolutionary computing fitness function:

a particular type of objective function that is used to summarize...how close a given design solution is to achieving the set aims.

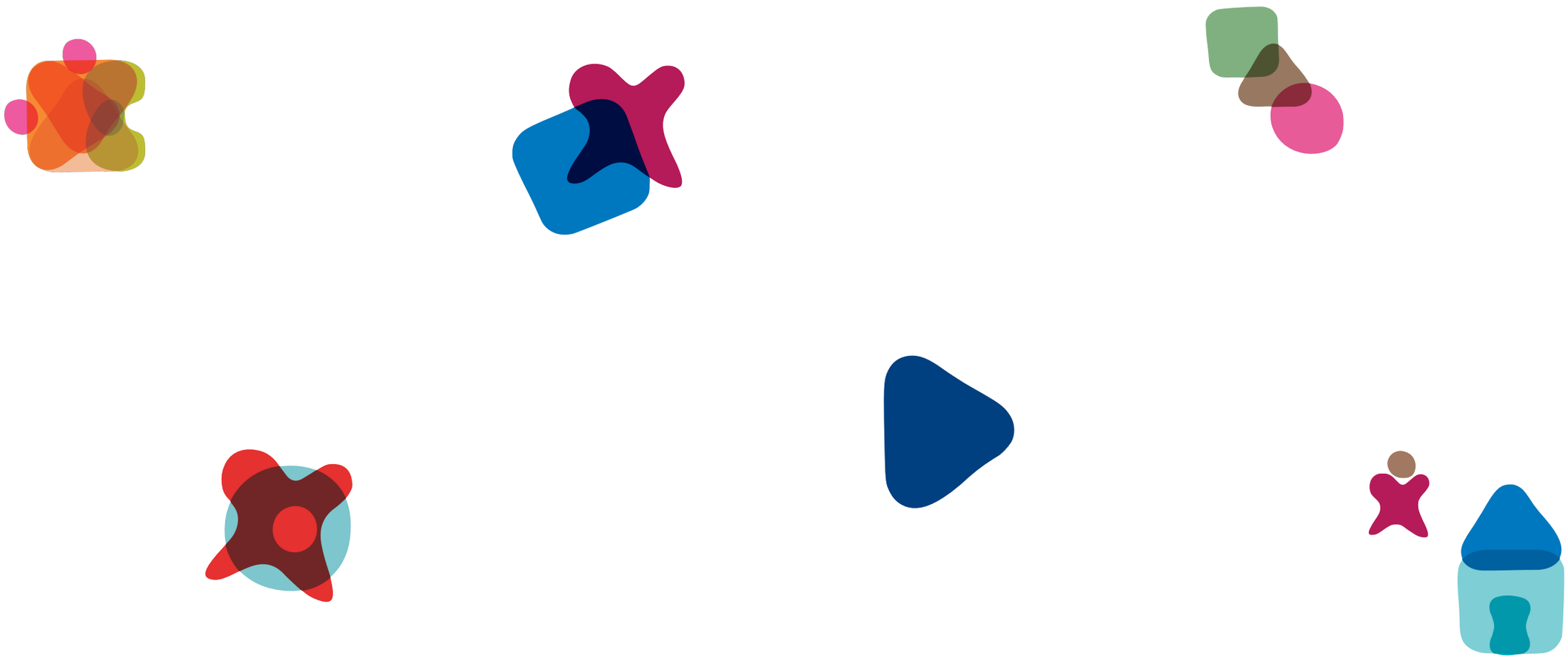
Traveling Salesman Problem





Traveling Salesman Problem



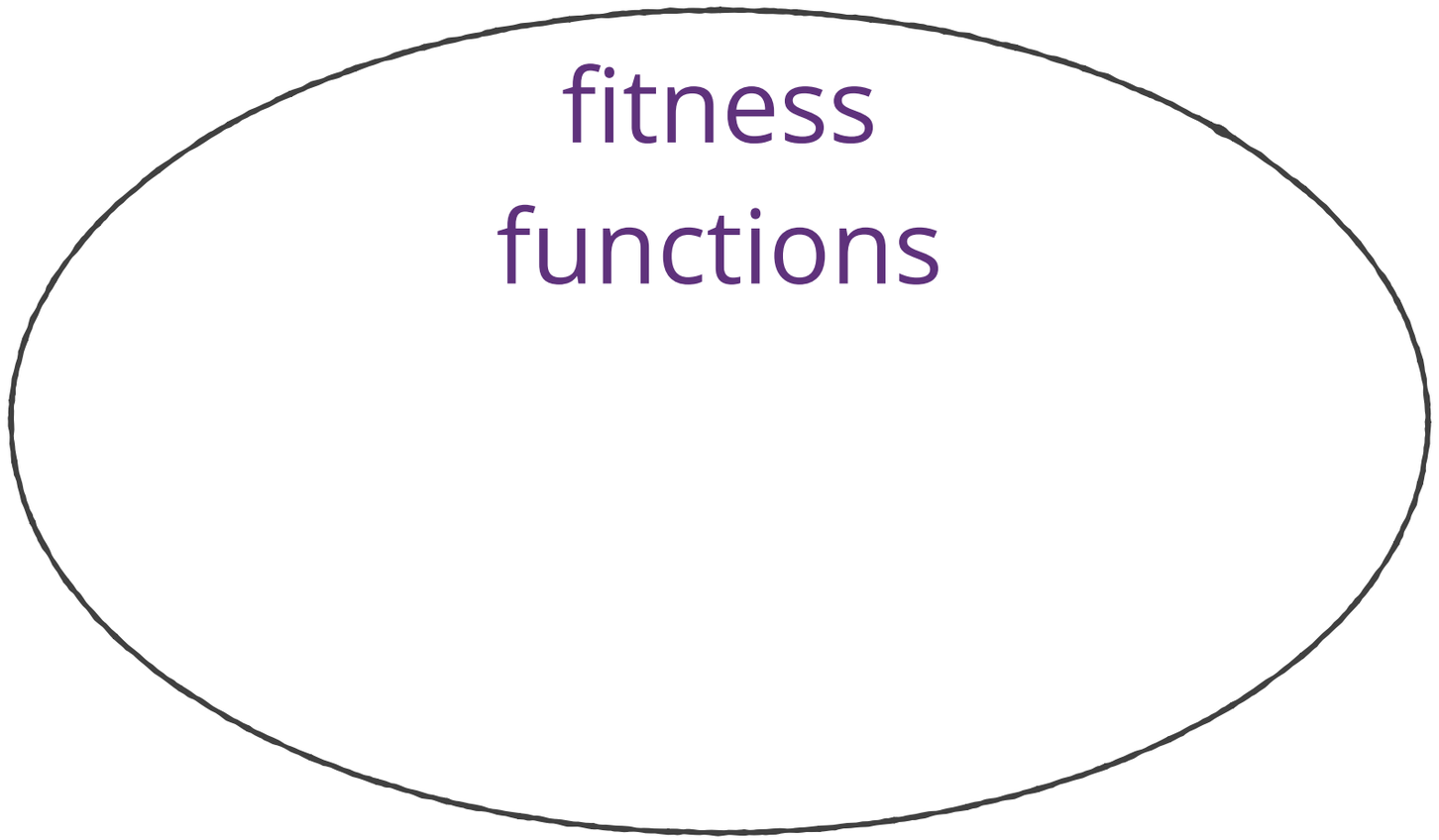
fitness function = length of route

guided

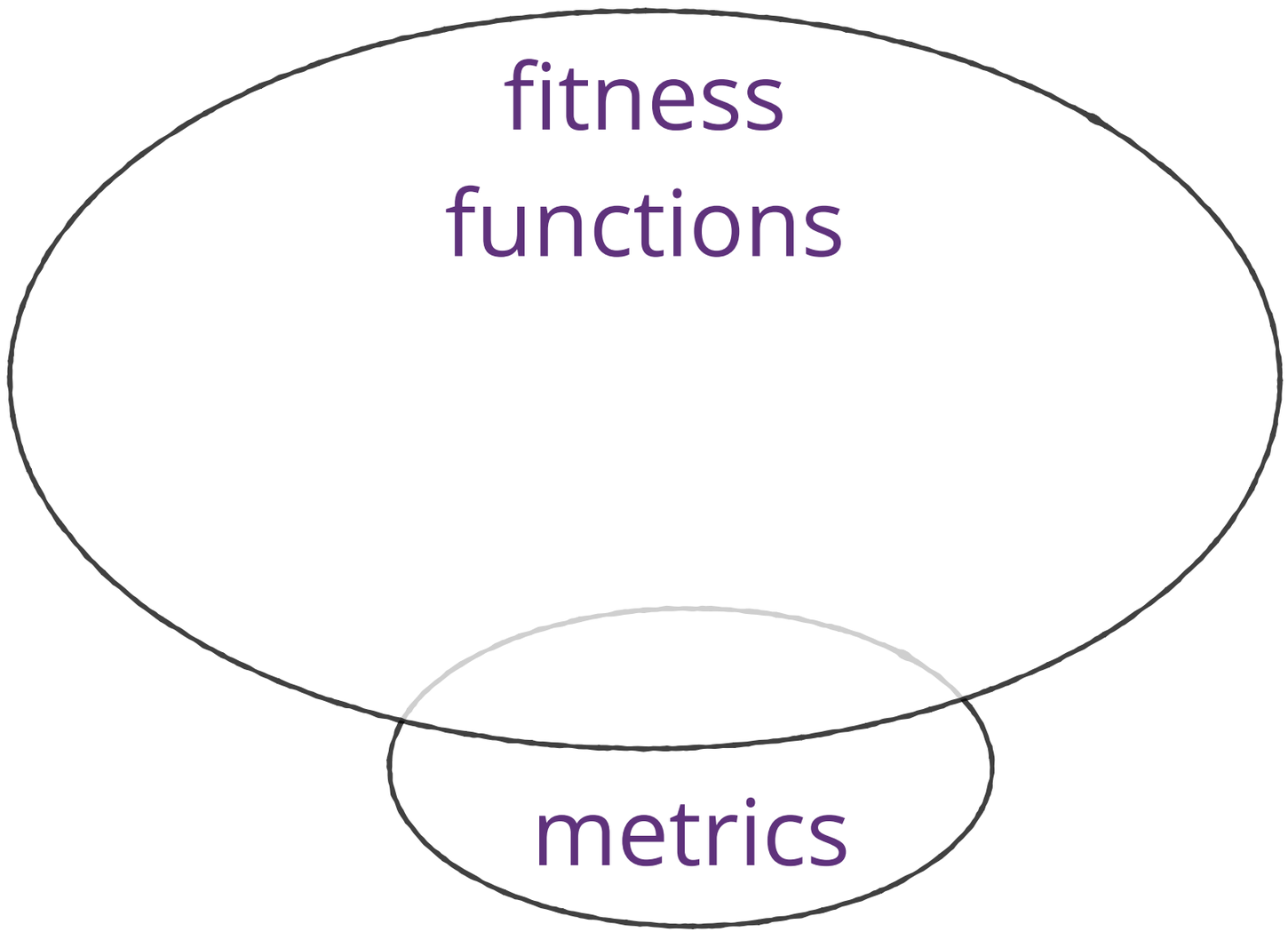
architectural *fitness function*:

any mechanism that provides an objective integrity assessment of some architectural characteristic(s).

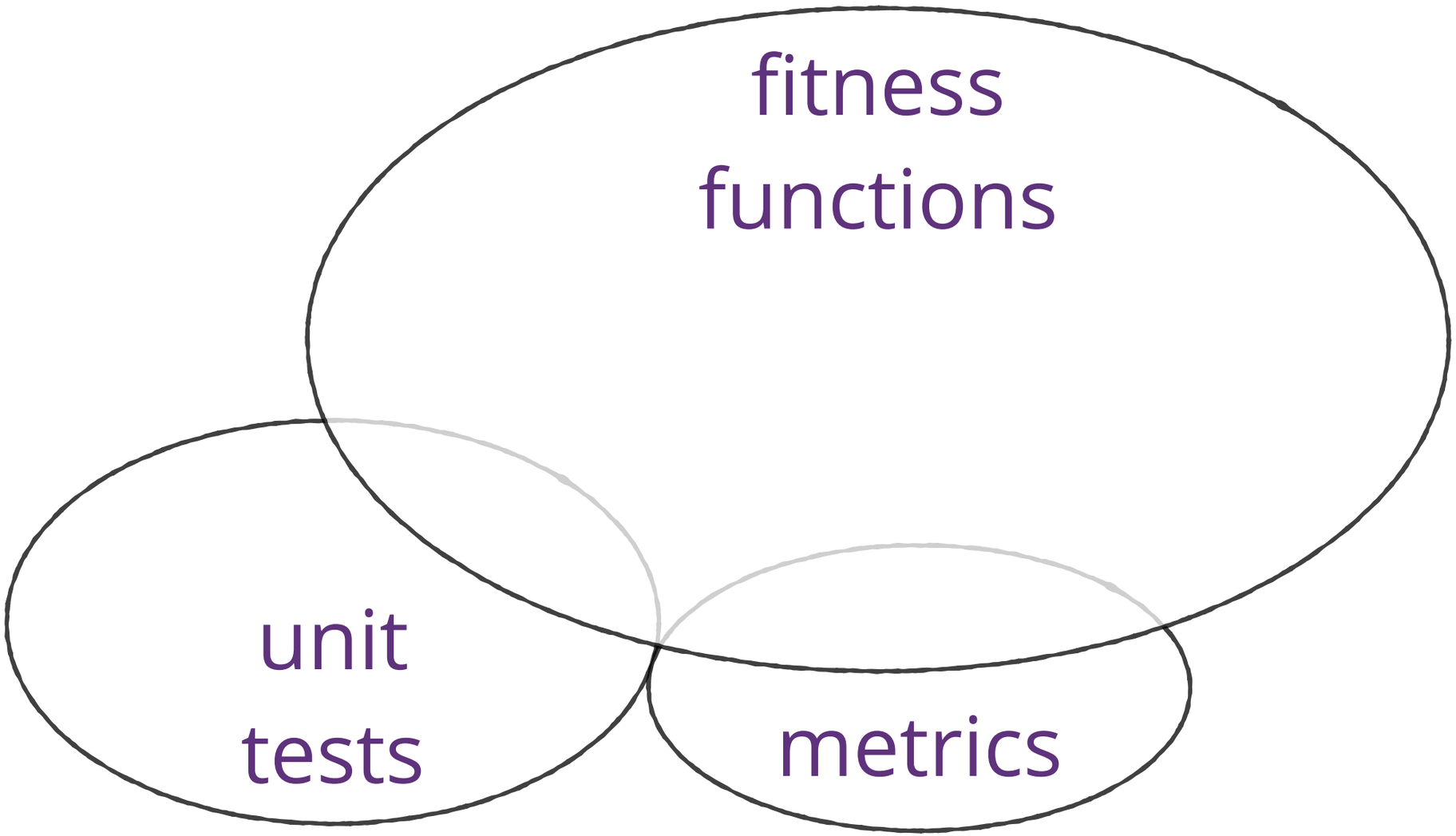
Fitness Functions



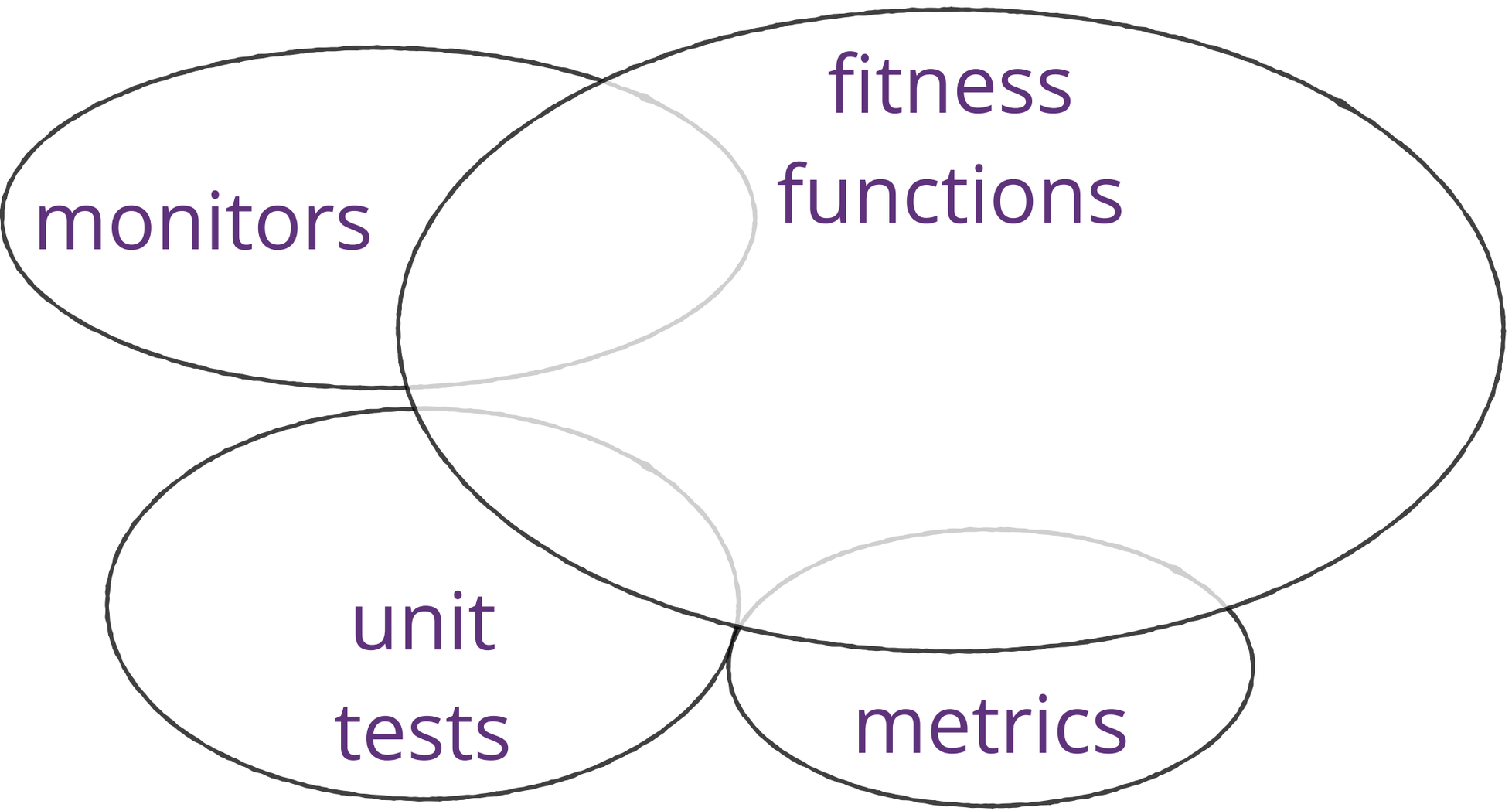
Fitness Functions



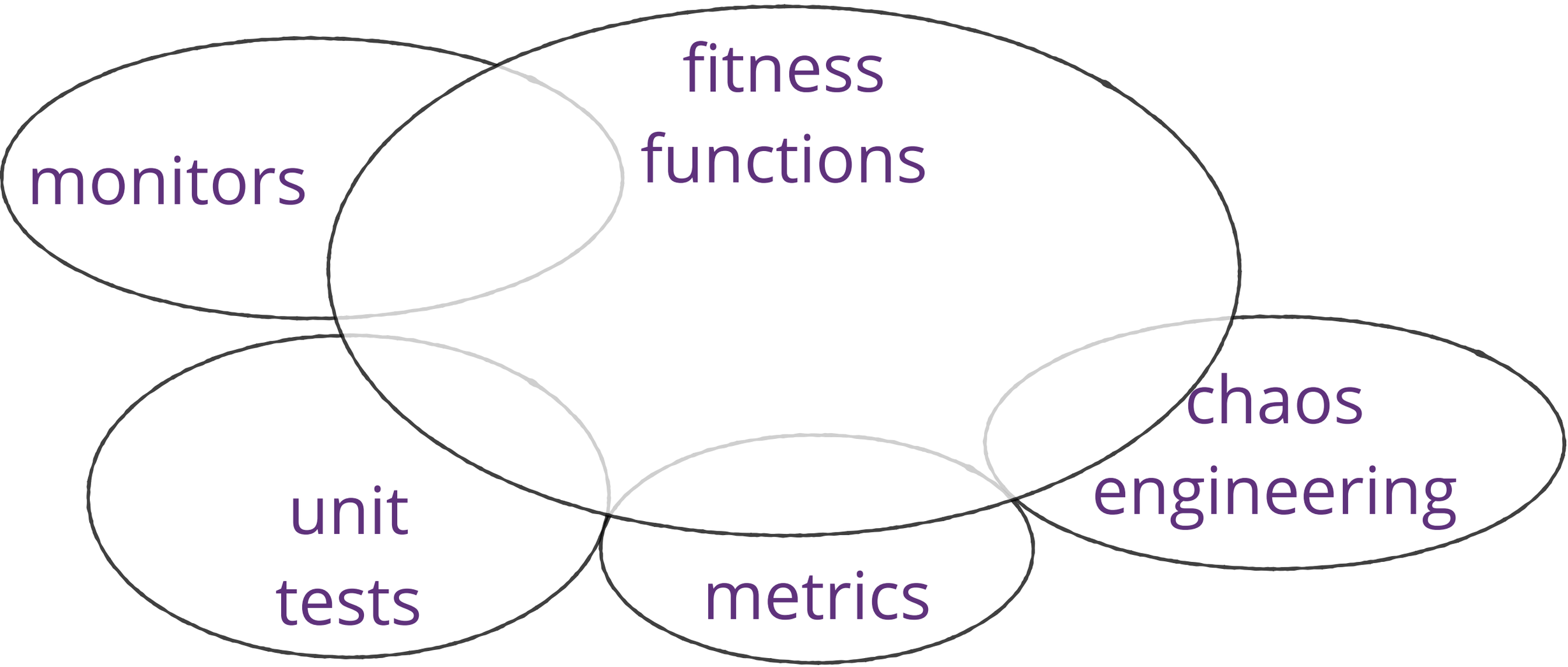
Fitness Functions



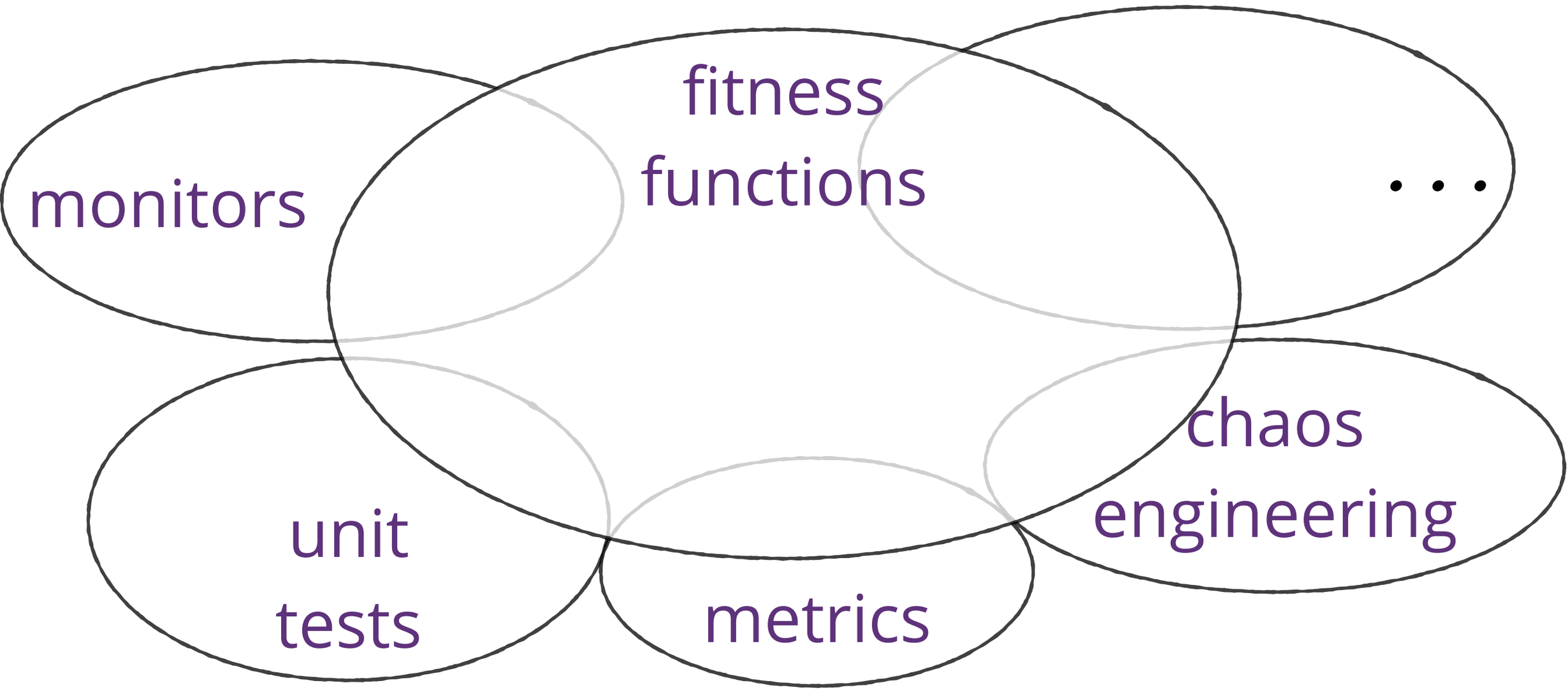
Fitness Functions



Fitness Functions

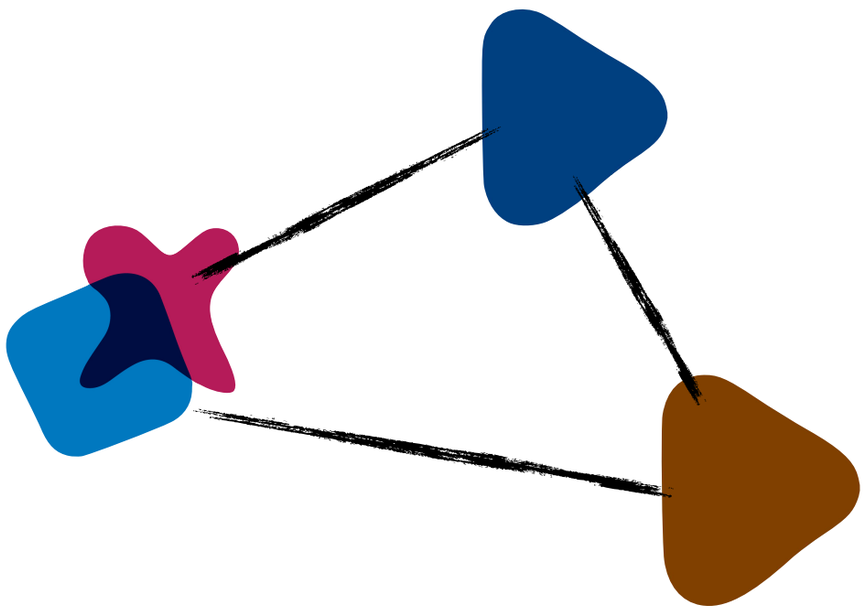


Fitness Functions

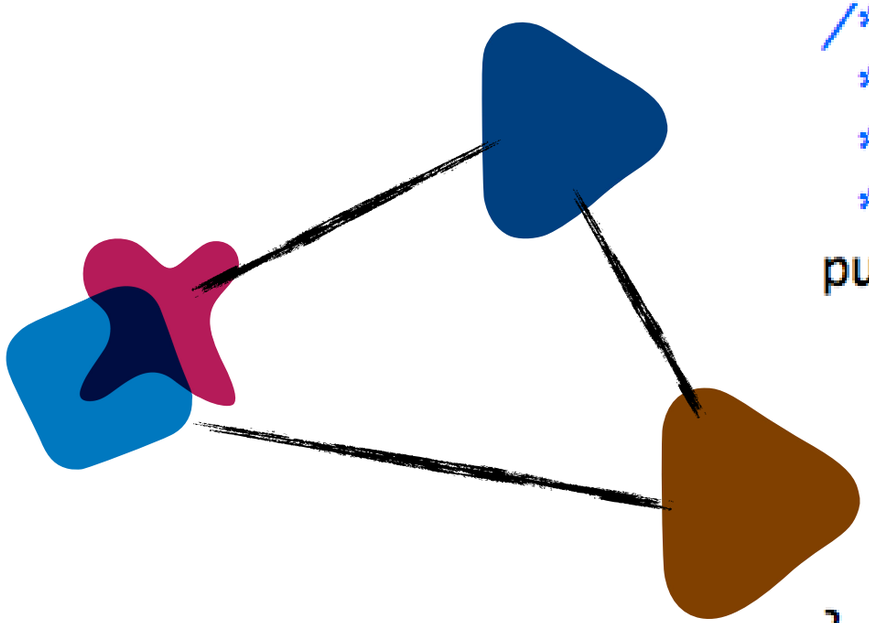


Cyclic Dependency Function

Cyclic Dependency Function



Cyclic Dependency Function



```
/**
 * Tests that a package dependency cycle does not
 * exist for any of the analyzed packages.
 */
public void testAllPackages() {

    Collection packages = jdepend.analyze();

    assertEquals("Cycles exist",
                 false, jdepend.containsCycles());
}
```

clarkware.com/software/JDepend.html

guided

architectural *fitness function*:

any mechanism that provides an objective integrity assessment of some architectural characteristic(s).

Categories of Fitness Functions

run against a singular context and exercise one particular aspect of the architecture.

Categories of Fitness Functions

run against a singular context and exercise one particular aspect of the architecture.

run against a shared context and exercise a combination of architectural aspects such as security and scalability

Categories of Fitness Functions

run based on a particular event:
— developer executing a unit test

Categories of Fitness Functions

run based on a particular event:
— developer executing a unit test
— deployment pipeline running tests

Categories of Fitness Functions

run based on a particular event:

— developer executing a unit test

— deployment pipeline running tests

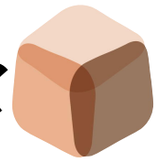
— timed task

Categories of Fitness Functions

run based on a particular event:
— developer executing a unit test
— deployment pipeline running tests
— timed task

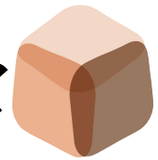
executes constant verification of architectural aspect(s)

Categories of Fitness Functions

static 

have a fixed result, such as the binary pass/ fail of a unit test.

Categories of Fitness Functions

static 

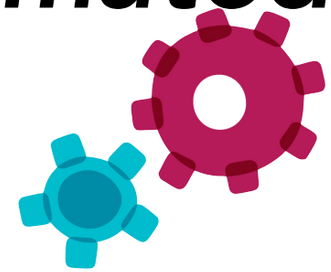
have a fixed result, such as the binary pass/ fail of a unit test.

dynamic 

rely on a shifting definition based on extra context.

Categories of Fitness Functions

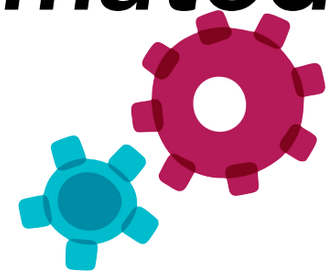
automated



tests and other verification mechanism that run without human interaction.

Categories of Fitness Functions

automated



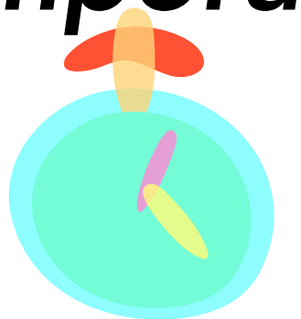
tests and other verification mechanism that run without human interaction.

manual

must involve at least one human.

Categories of Fitness Functions

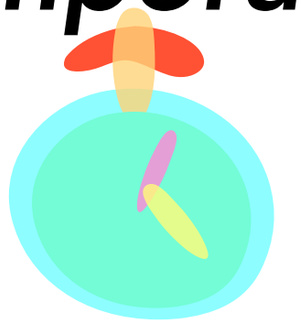
temporal



architects may want to build a time component into assessing fitness

Categories of Fitness Functions

temporal

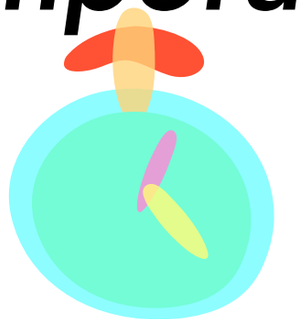


architects may want to build a time component into assessing fitness

break on upgrade

Categories of Fitness Functions

temporal



architects may want to build a time component into assessing fitness

break on upgrade

overdue library update

Categories of Fitness Functions

domain-specific

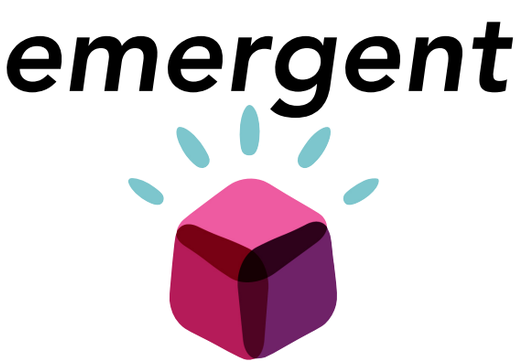
Some architectures have specific concerns, such as special security or regulatory requirements

Categories of Fitness Functions

architects will define most fitness functions at project inception as they elucidate the characteristics of the architecture...

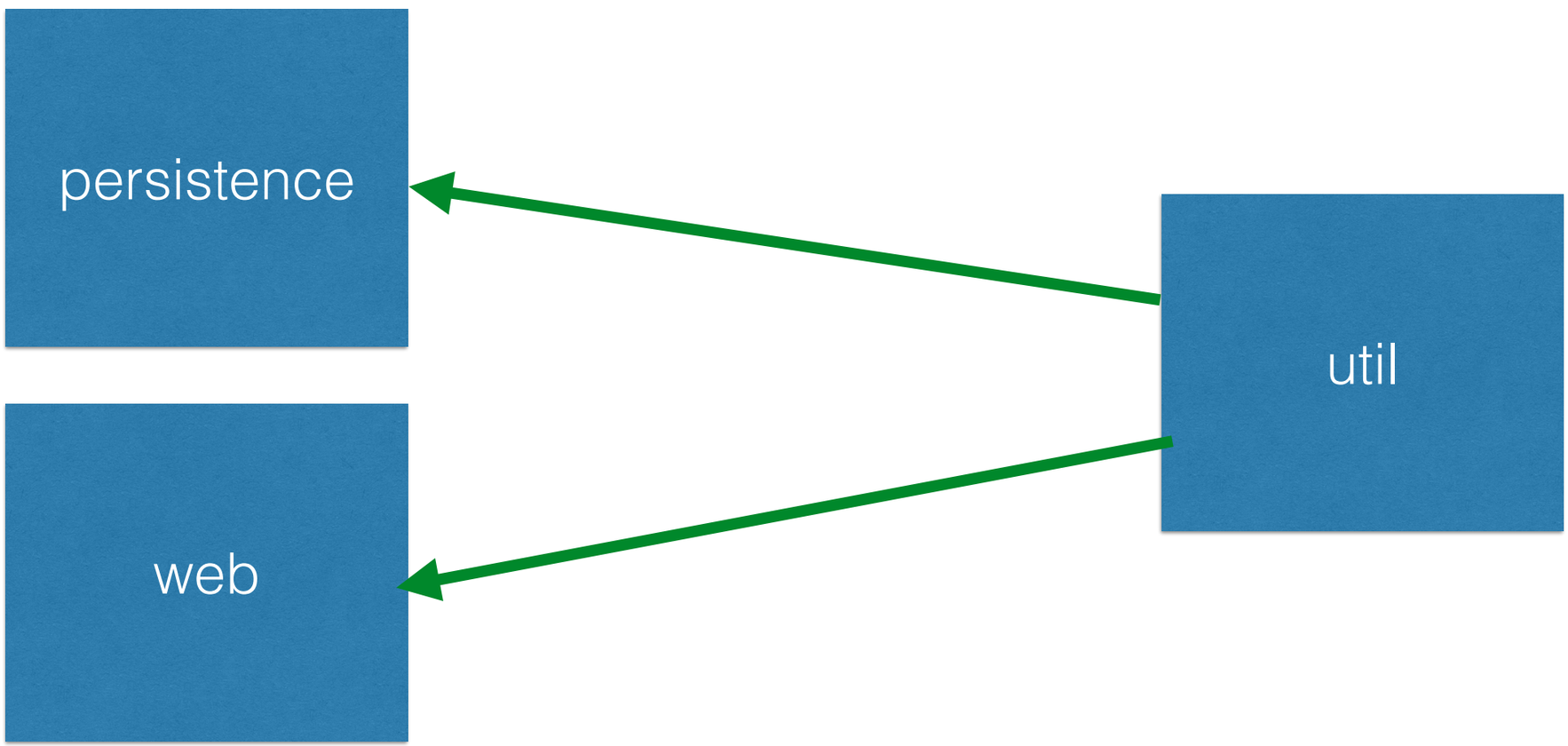
Categories of Fitness Functions

architects will define most fitness functions at project inception as they elucidate the characteristics of the architecture...



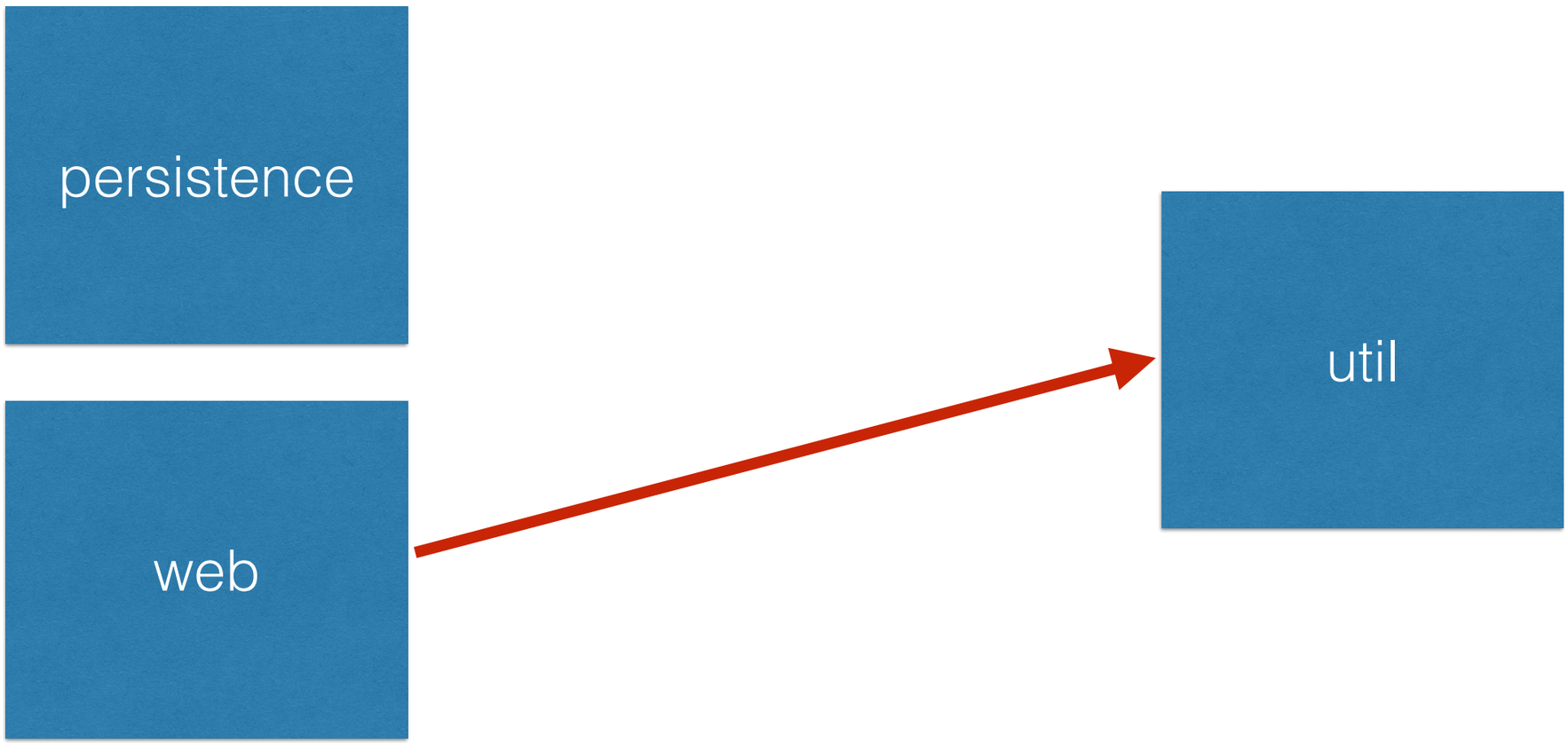
...some fitness functions will emerge during development of the system

Directionality of Imports



packages/namespaces

Directionality of Imports

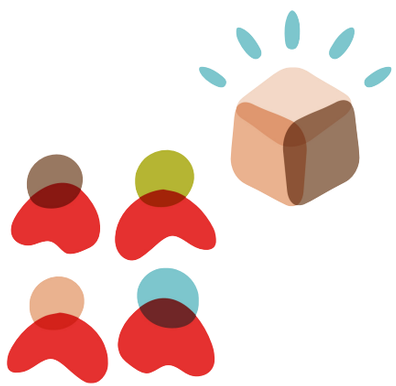


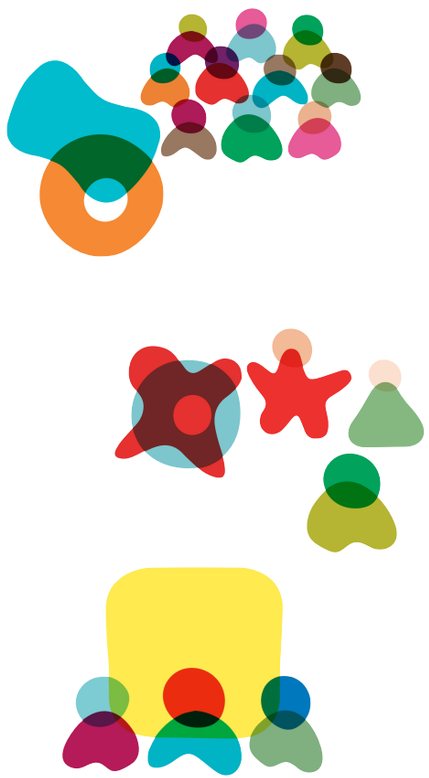
packages/namespaces

Coupling Fitness Function

```
public void testMatch() {
    DependencyConstraint constraint = new DependencyConstraint();

    JavaPackage persistence = constraint.addPackage("com.xyz.persistence");
    JavaPackage web = constraint.addPackage("com.xyz.web");
    JavaPackage util = constraint.addPackage("com.xyz.util");

    persistence.dependsUpon(util);
    web.dependsUpon(util);

    jdepend.analyze();

    assertEquals("Dependency mismatch",
            true, jdepend.dependencyMatch(constraint));
    }
```

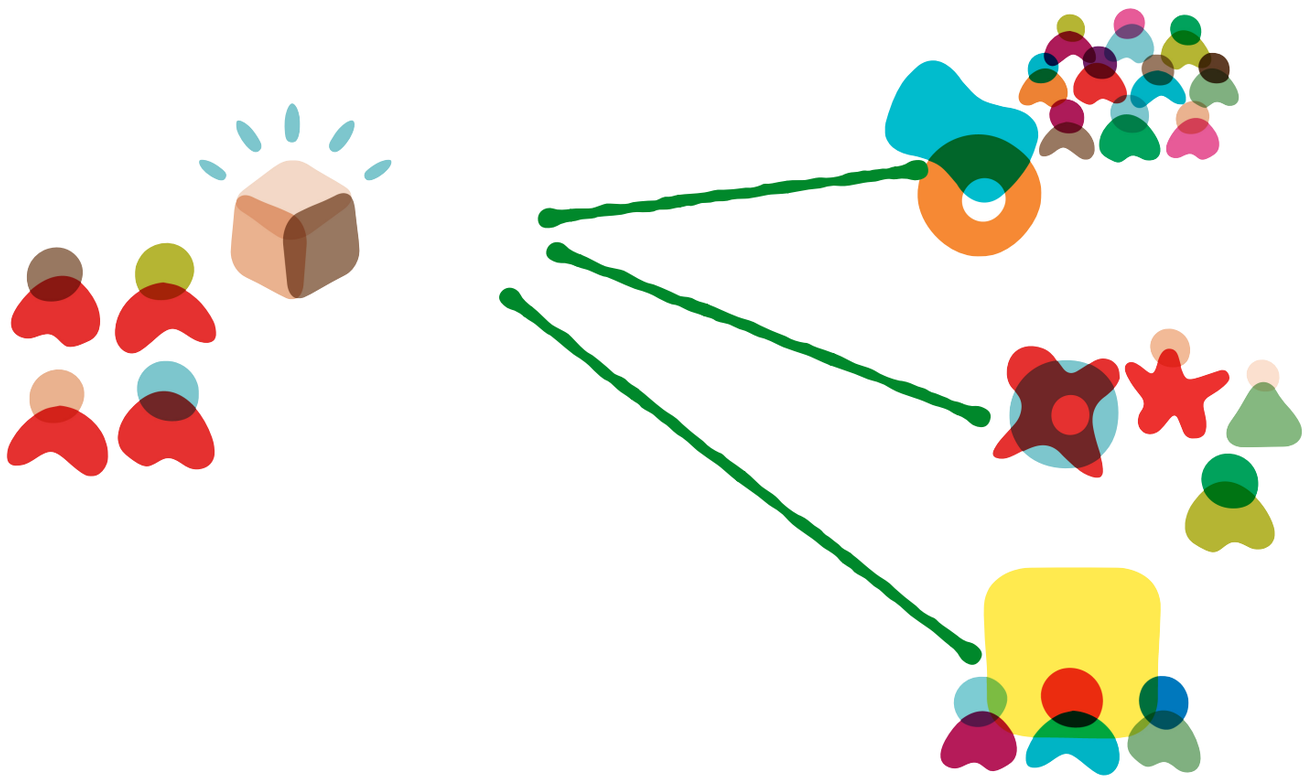
Consumer Driven Contracts





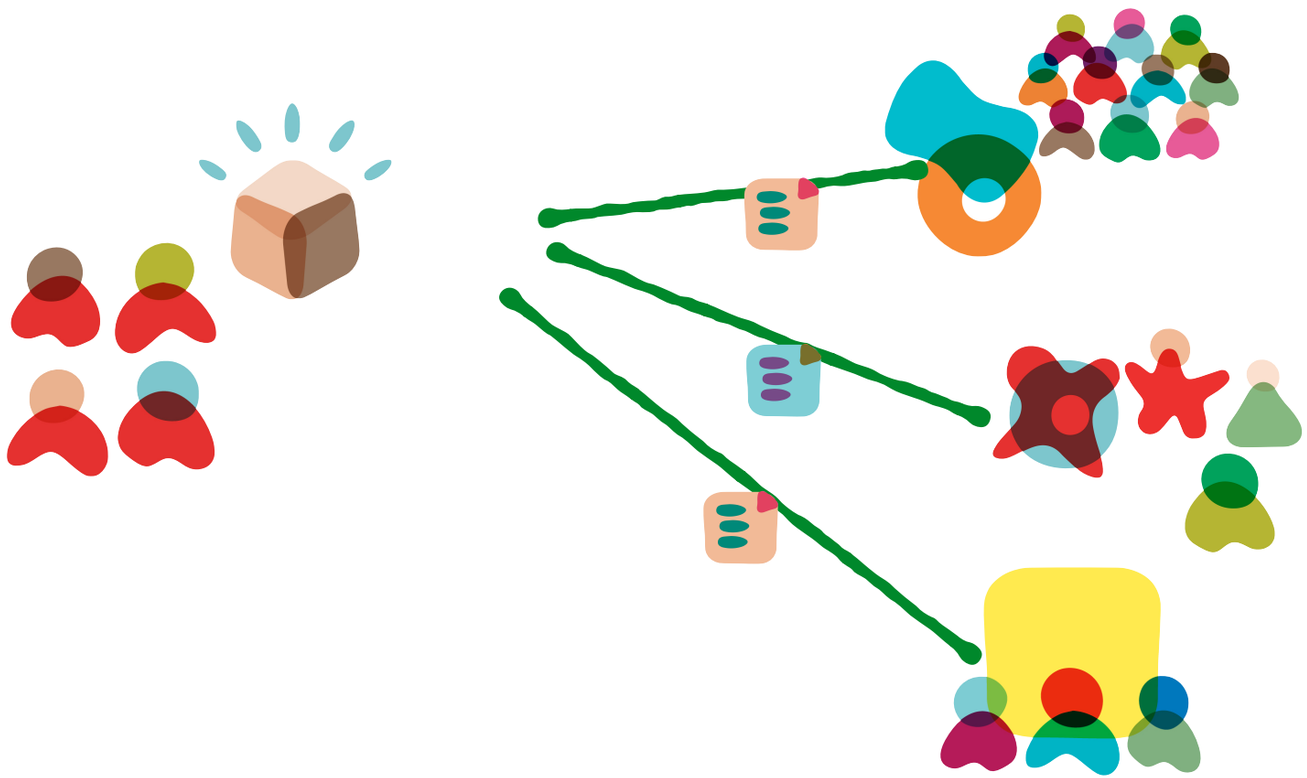
martinfowler.com/articles/consumerDrivenContracts.html

Consumer Driven Contracts



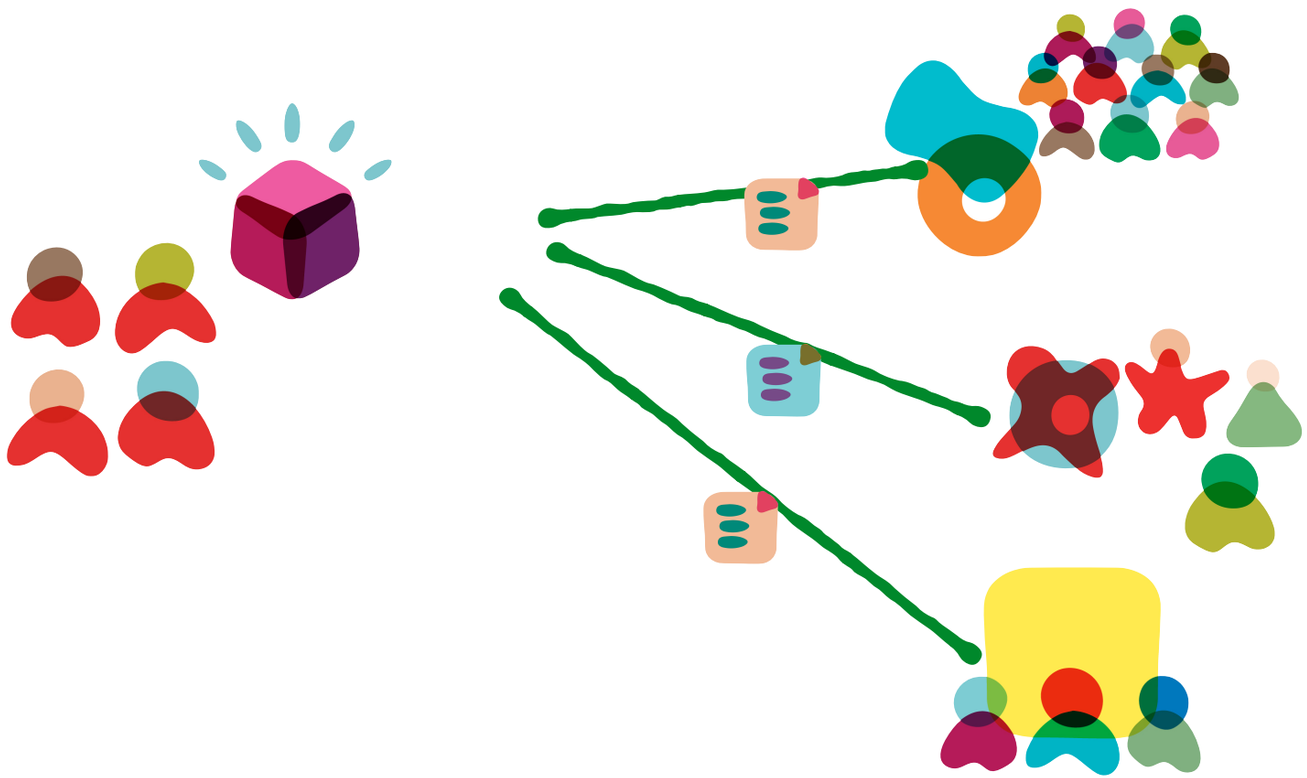
martinfowler.com/articles/consumerDrivenContracts.html

Consumer Driven Contracts



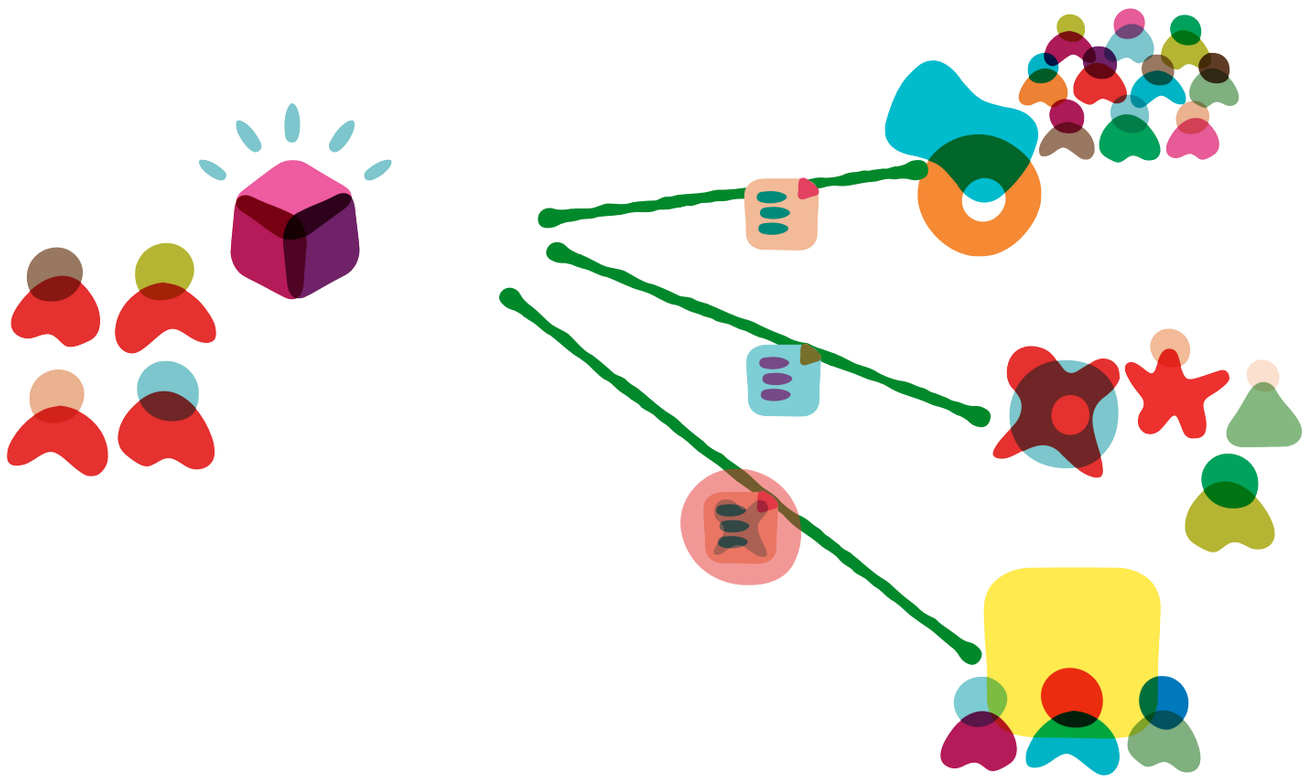
martinfowler.com/articles/consumerDrivenContracts.html

Consumer Driven Contracts



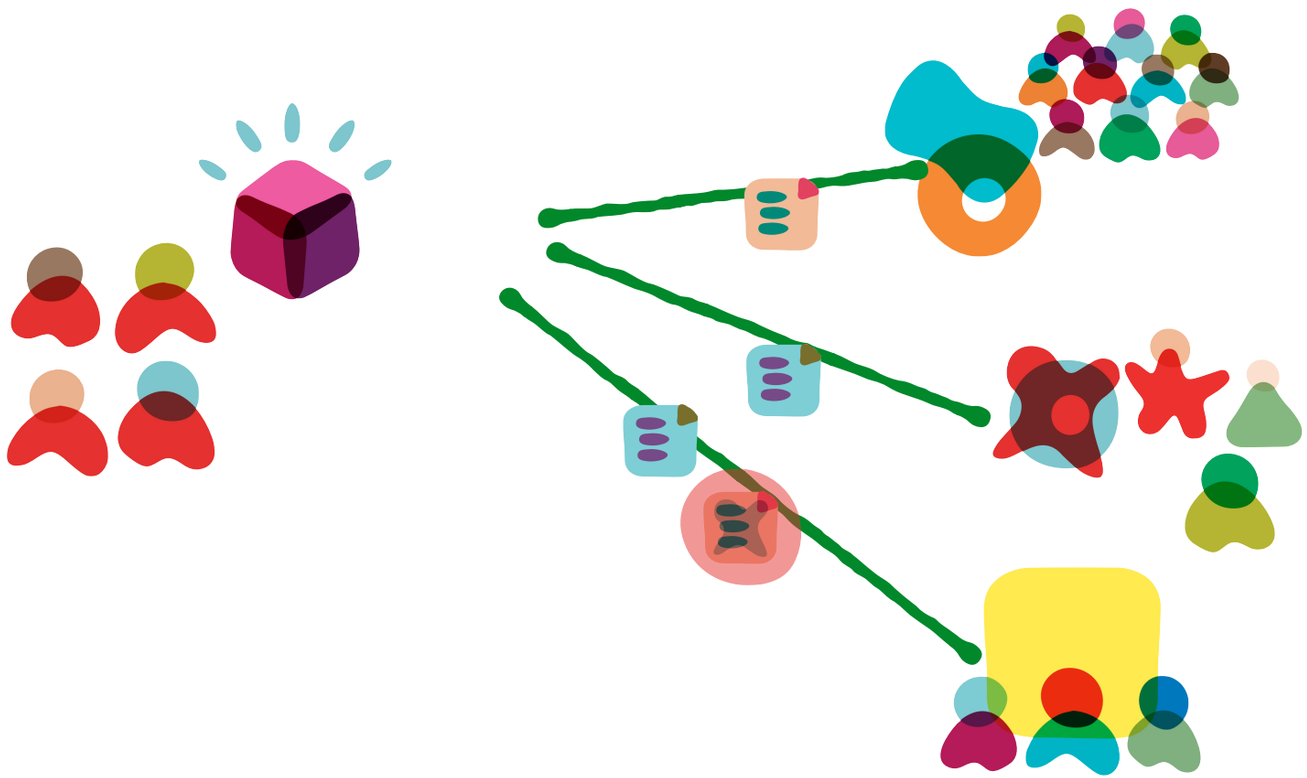
martinfowler.com/articles/consumerDrivenContracts.html

Consumer Driven Contracts



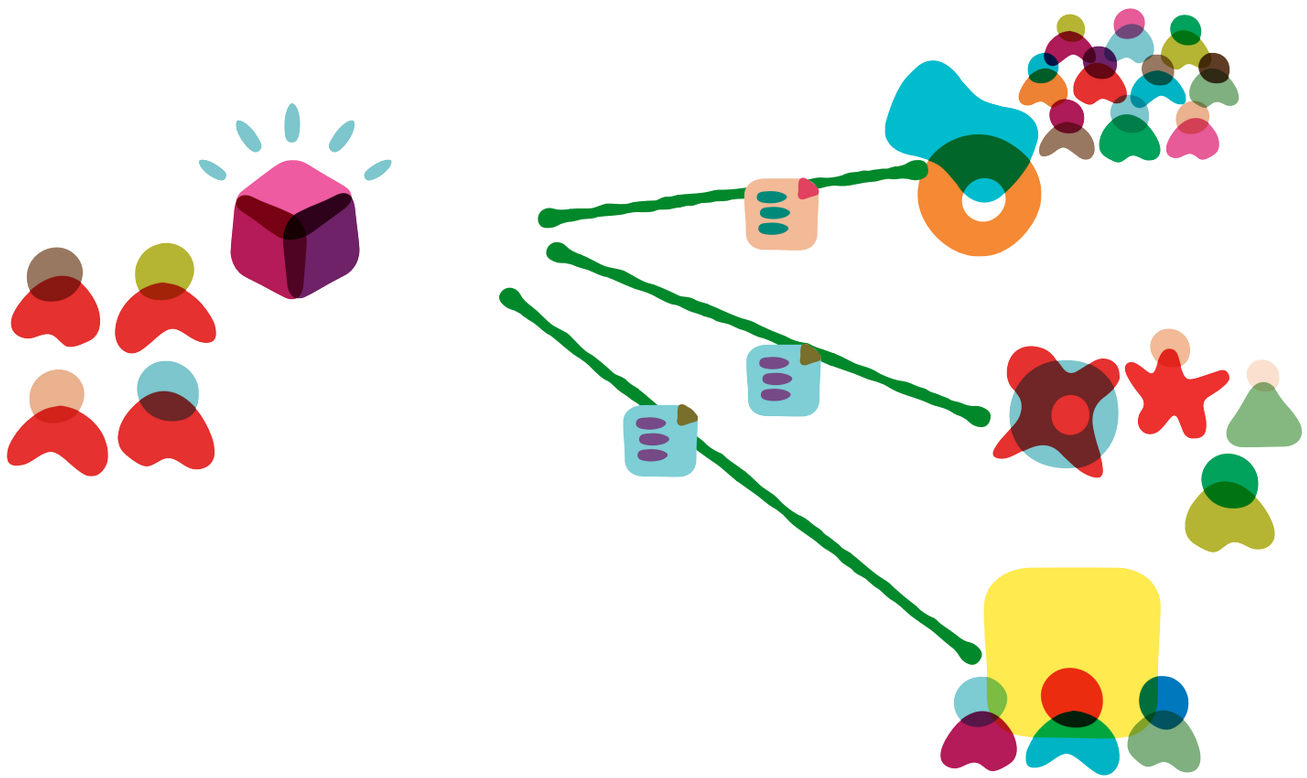
martinfowler.com/articles/consumerDrivenContracts.html

Consumer Driven Contracts



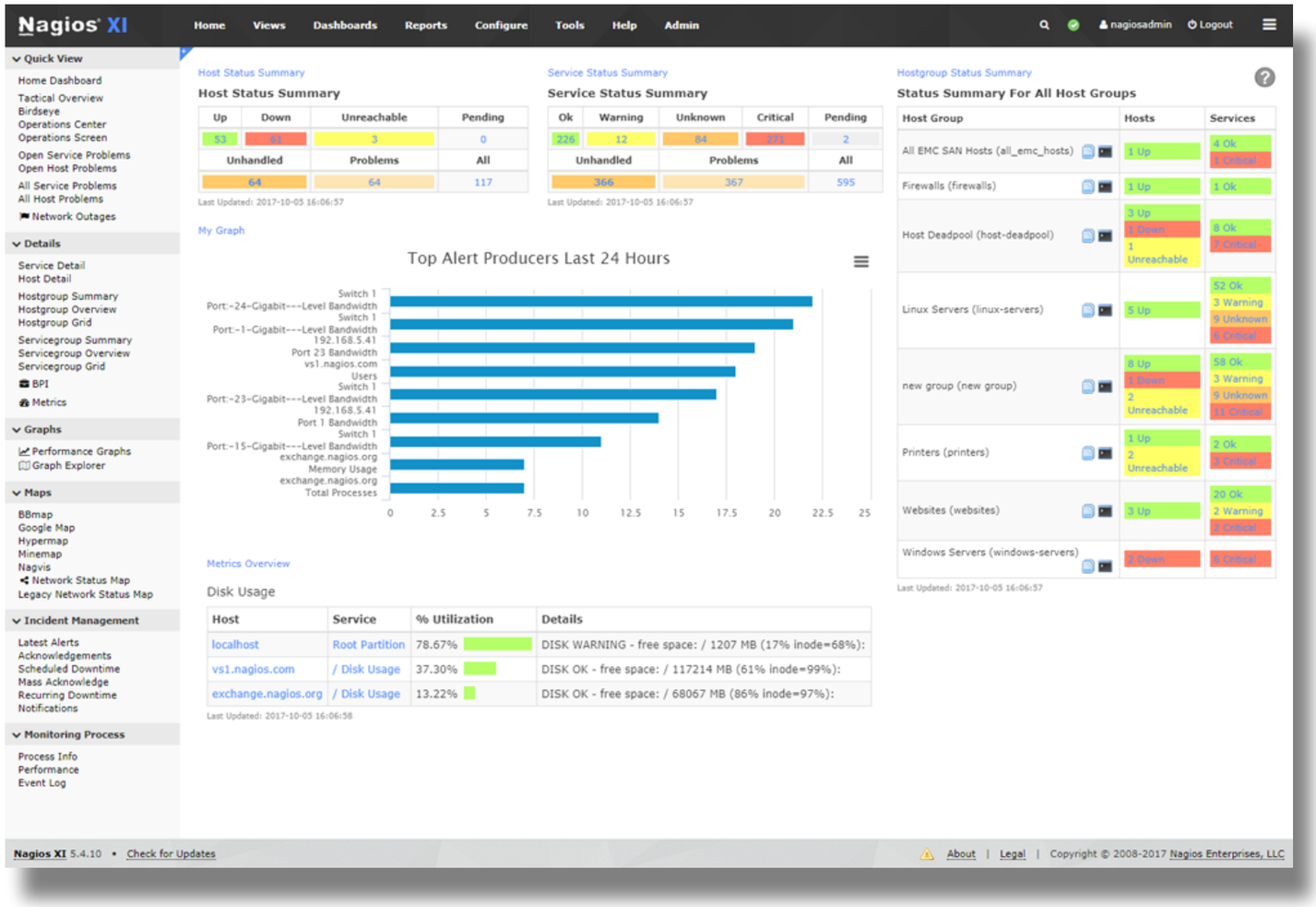
martinfowler.com/articles/consumerDrivenContracts.html

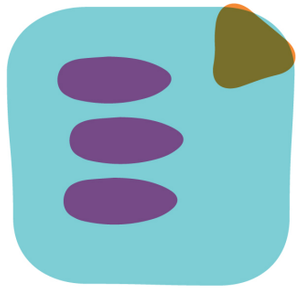
Consumer Driven Contracts



martinfowler.com/articles/consumerDrivenContracts.html

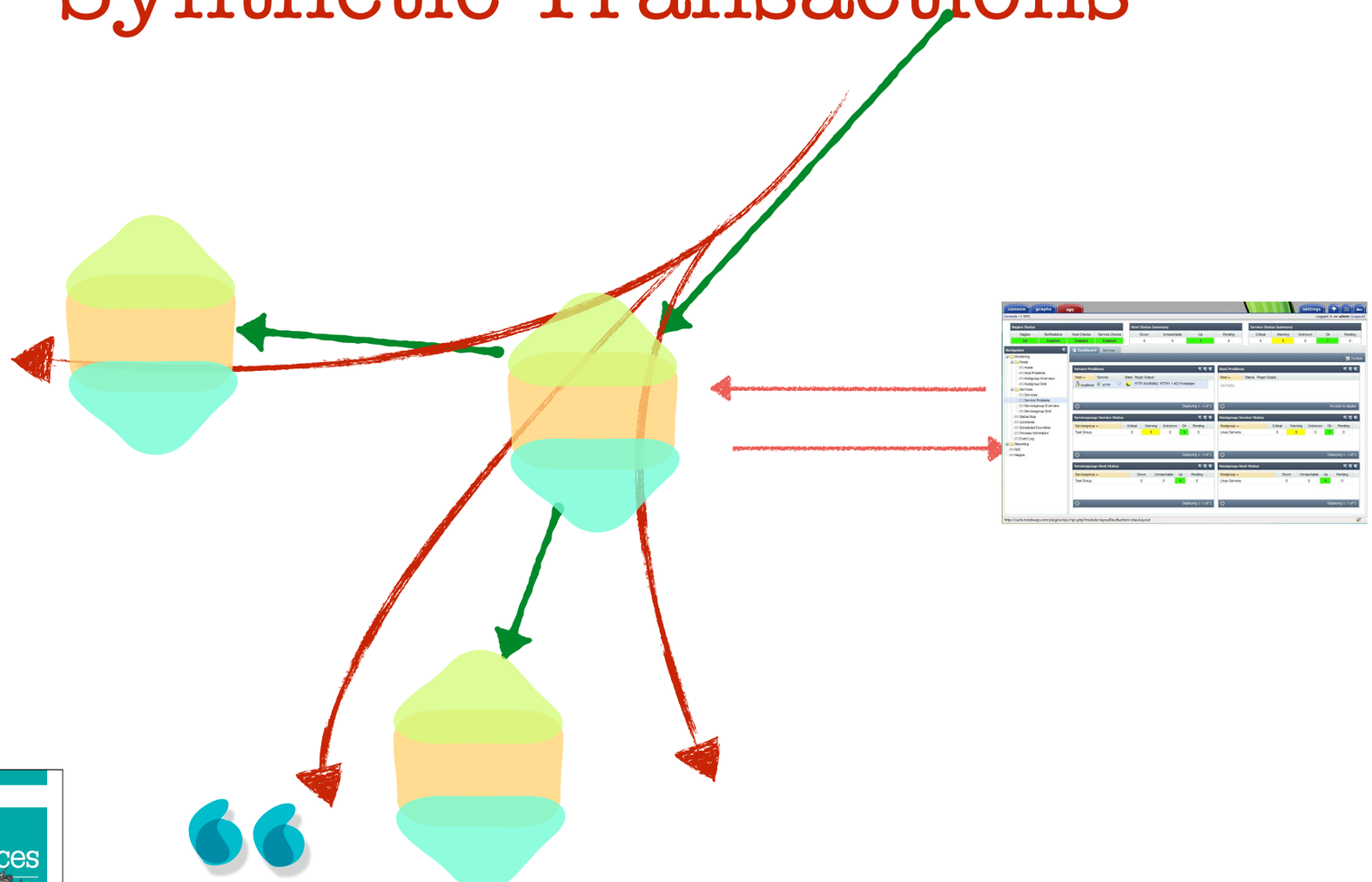
monitoring





logging

Synthetic Transactions



Use synthetic transactions to test production systems.


amazon
web services
CHAOS
MONKEY


amazon
web services
CHAOS
MONKEY


SIMIAN
ARMY

chaos monkey

chaos gorilla

latency monkey


doctor
monkey
SIMIAN
ARMY

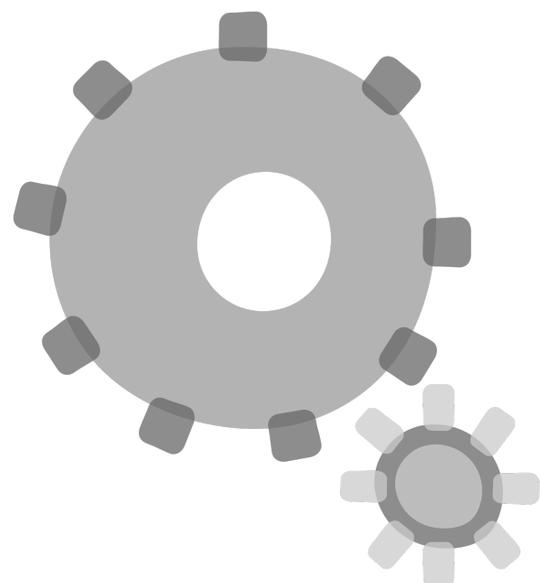
janitor monkey

conformity monkey

security monkey

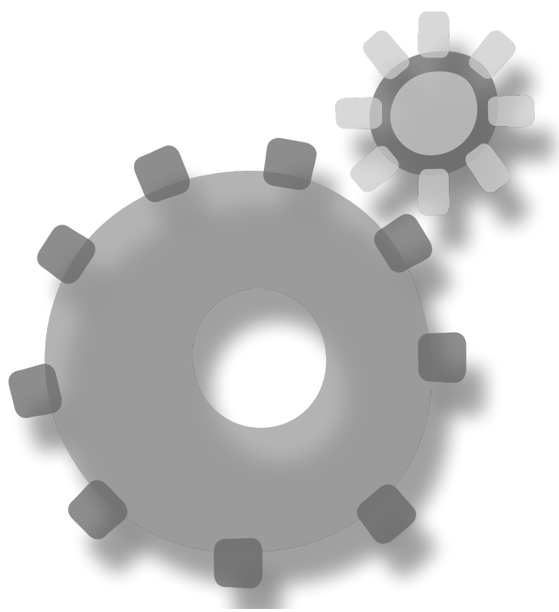
conformity monkey

Building Fitness Functions



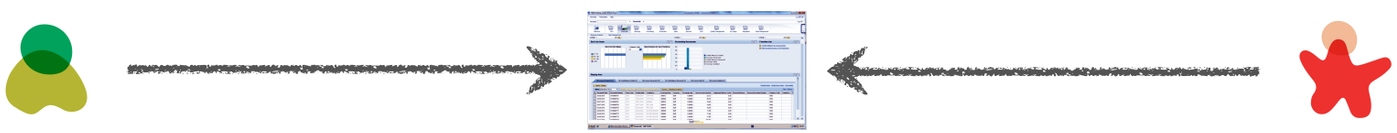
Scope of Fitness Functions

architectural characteristic

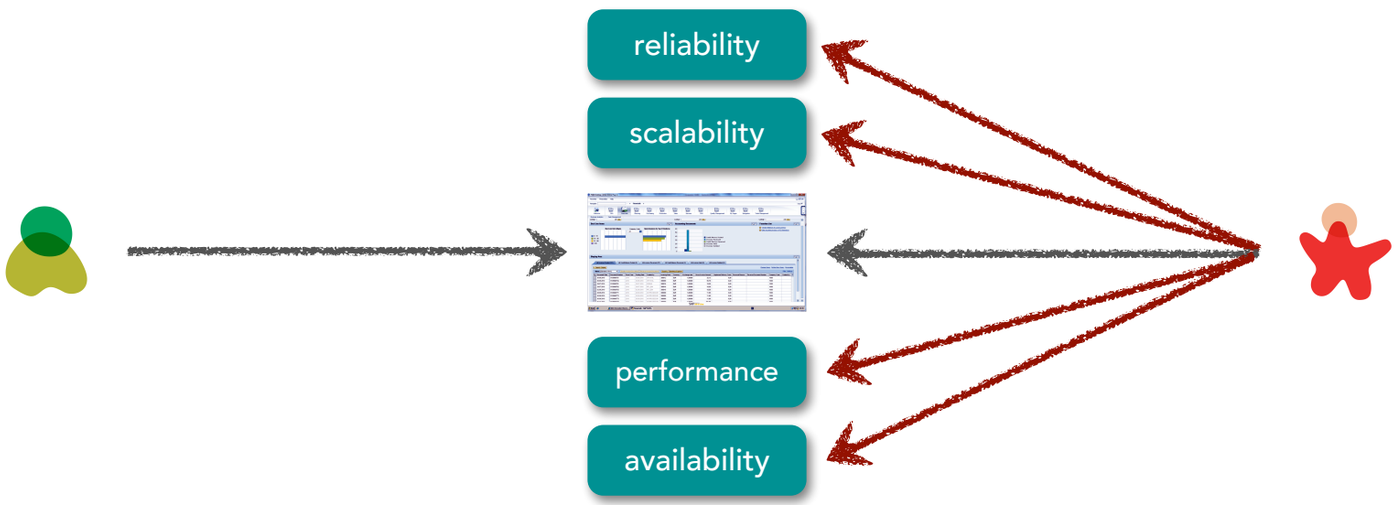


problem domain

Architecture Characteristics



Architecture Characteristics



Architecture Characteristics

1. synergistic

2. ill-defined

3. voluminous

Architecture Characteristics

accessibility
accountability
accuracy
adaptability
administrability
affordability
agility
auditability
autonomy
availability
compatibility
composability
configurability
correctness
credibility
customizability
debugability
degradability
determinability
demonstrability
dependability
deployability
discoverability
distributability
durability
effectiveness
efficiency
evolvability
extensibility
failure transparency
fault-tolerance
fidelity
flexibility
inspectability
installability
integrity
interchangeability
interoperability
learnability
maintainability
manageability
mobility
modifiability
modularity
operability
orthogonality
portability
precision
predictability
process capabilities
producibility
provability
recoverability
relevance
reliability
repeatability
reproducibility
resilience
responsiveness
reusability
robustness
safety
scalability
seamlessness
self-sustainability
serviceability
supportability
securability
simplicity
stability
standards compliance
survivability
sustainability
tailorability
testability
timeliness
traceability
transparency
ubiquity
understandability
upgradability
usability

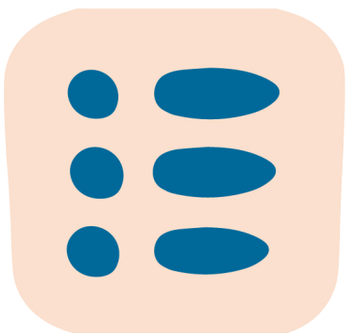
https://en.wikipedia.org/wiki/List_of_system_quality_attributes

Architecture Characteristics

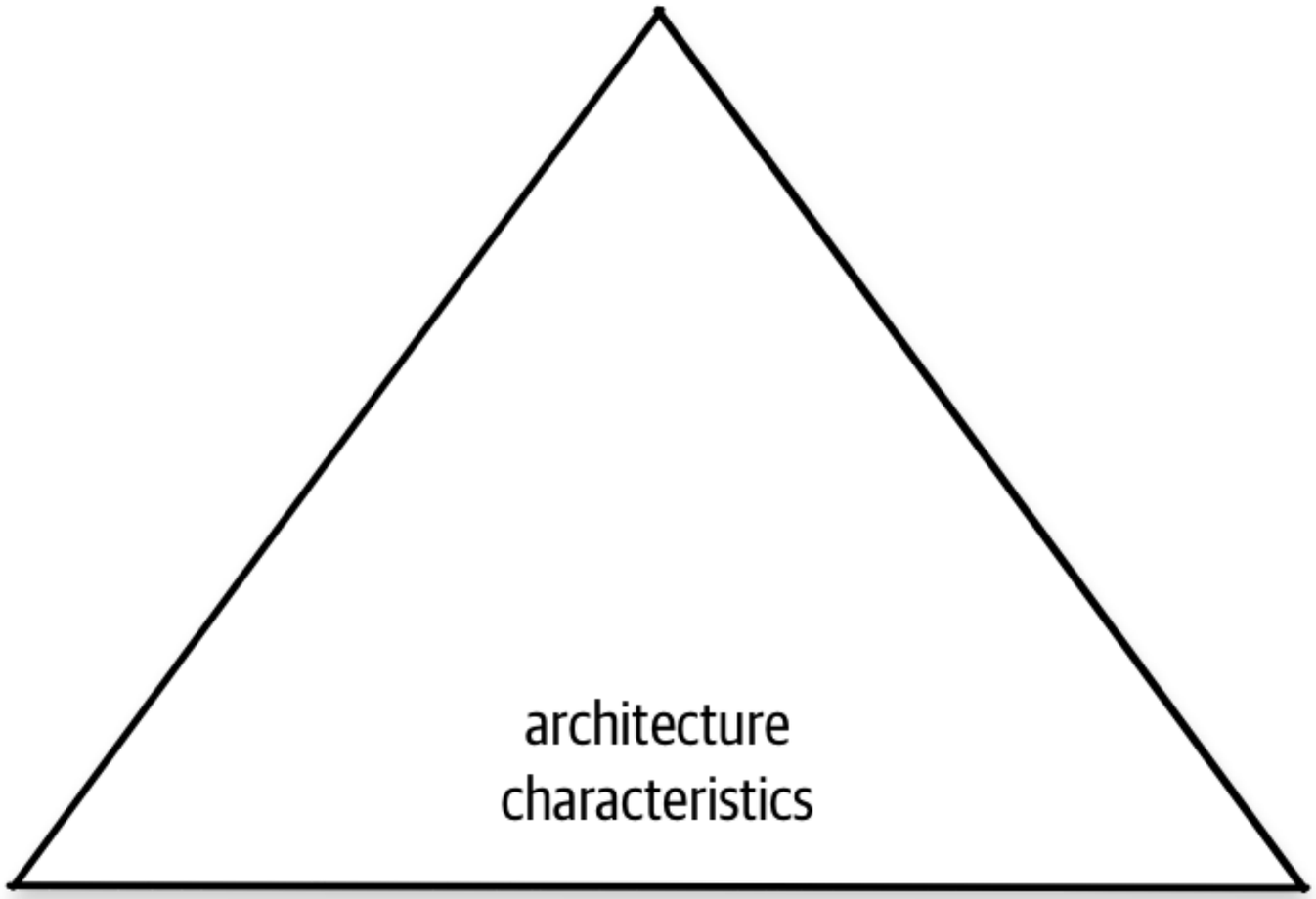
1. synergistic

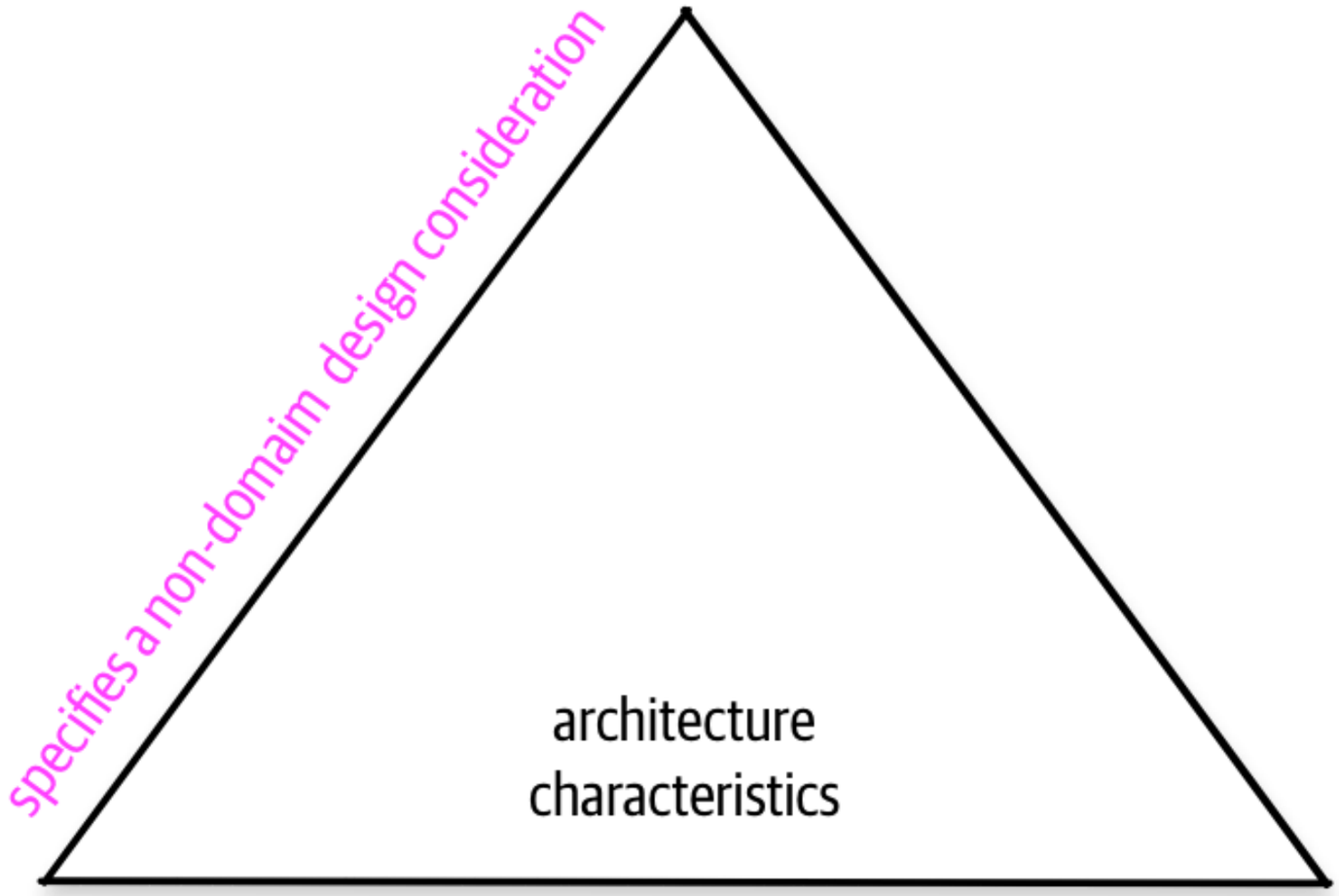
2. ill-defined

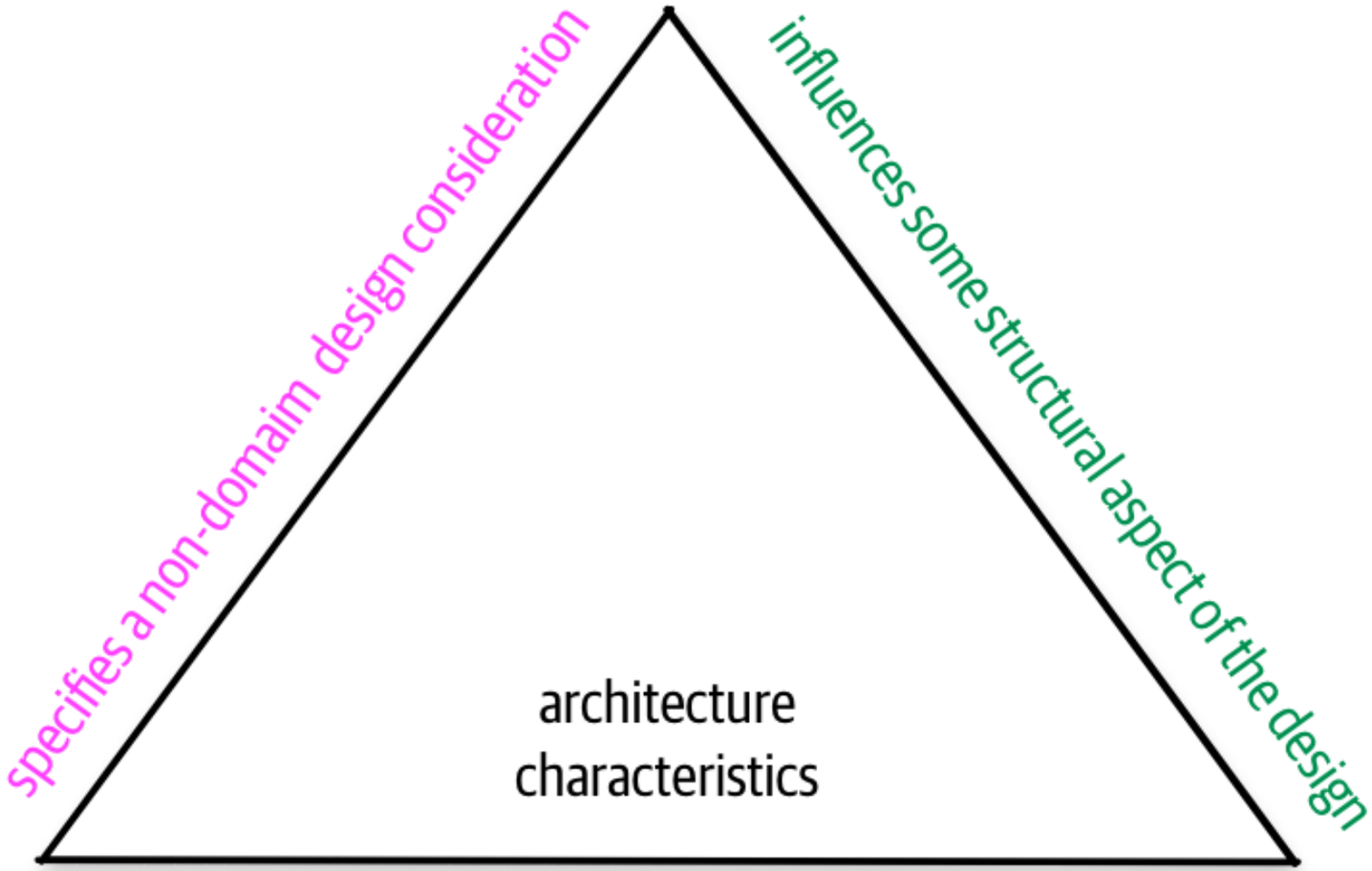
4. numerous categories

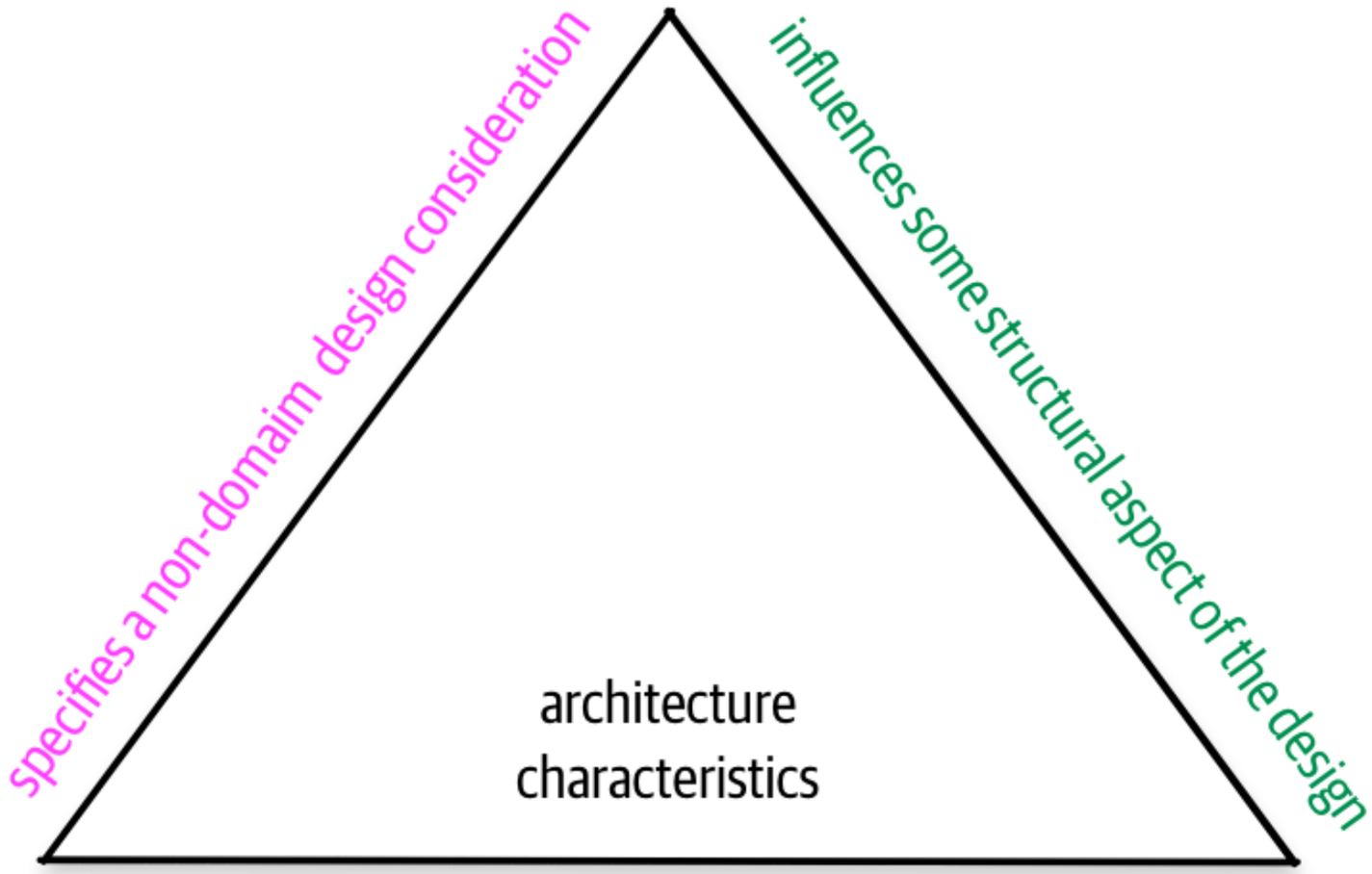


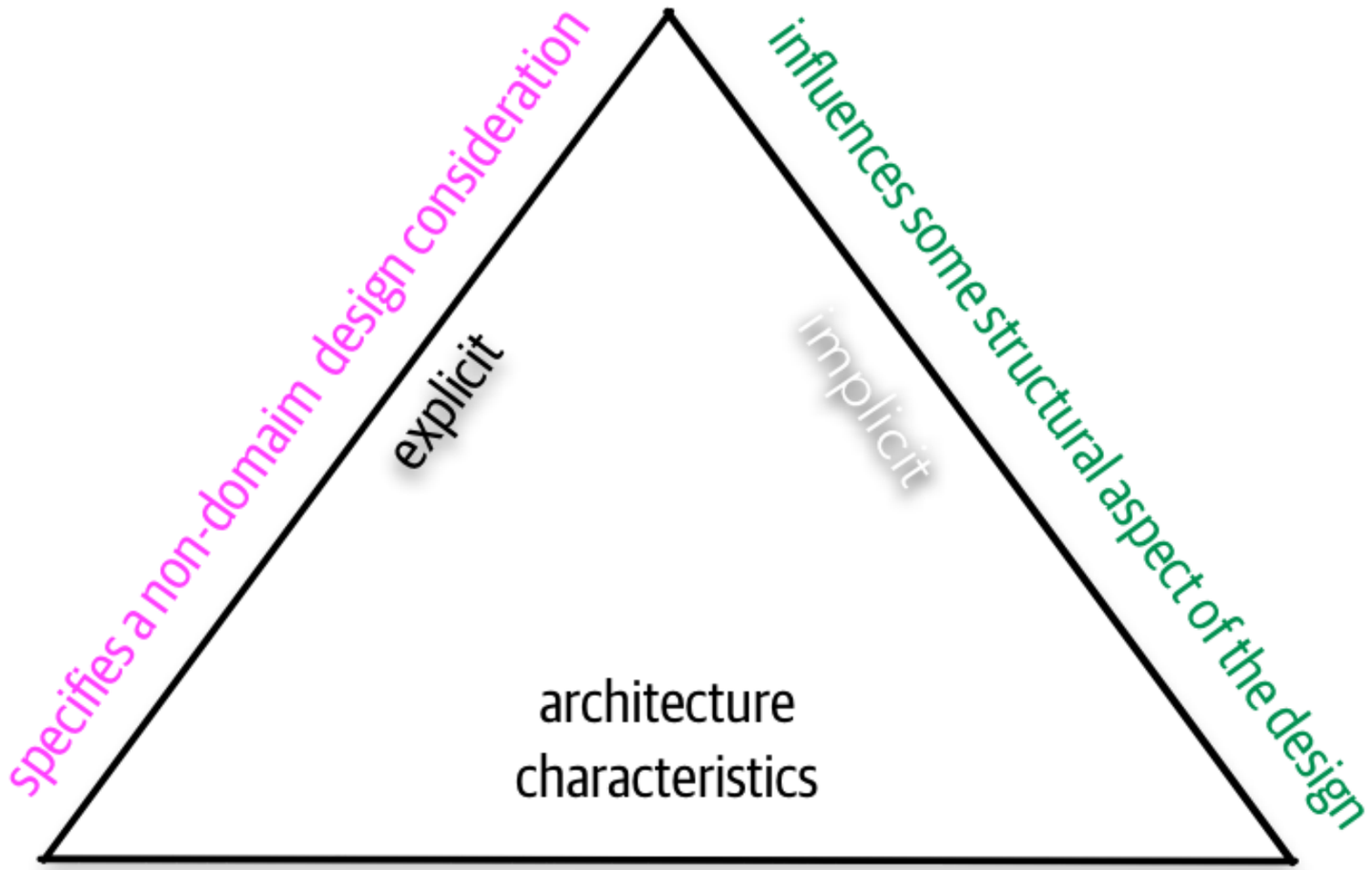
3. voluminous

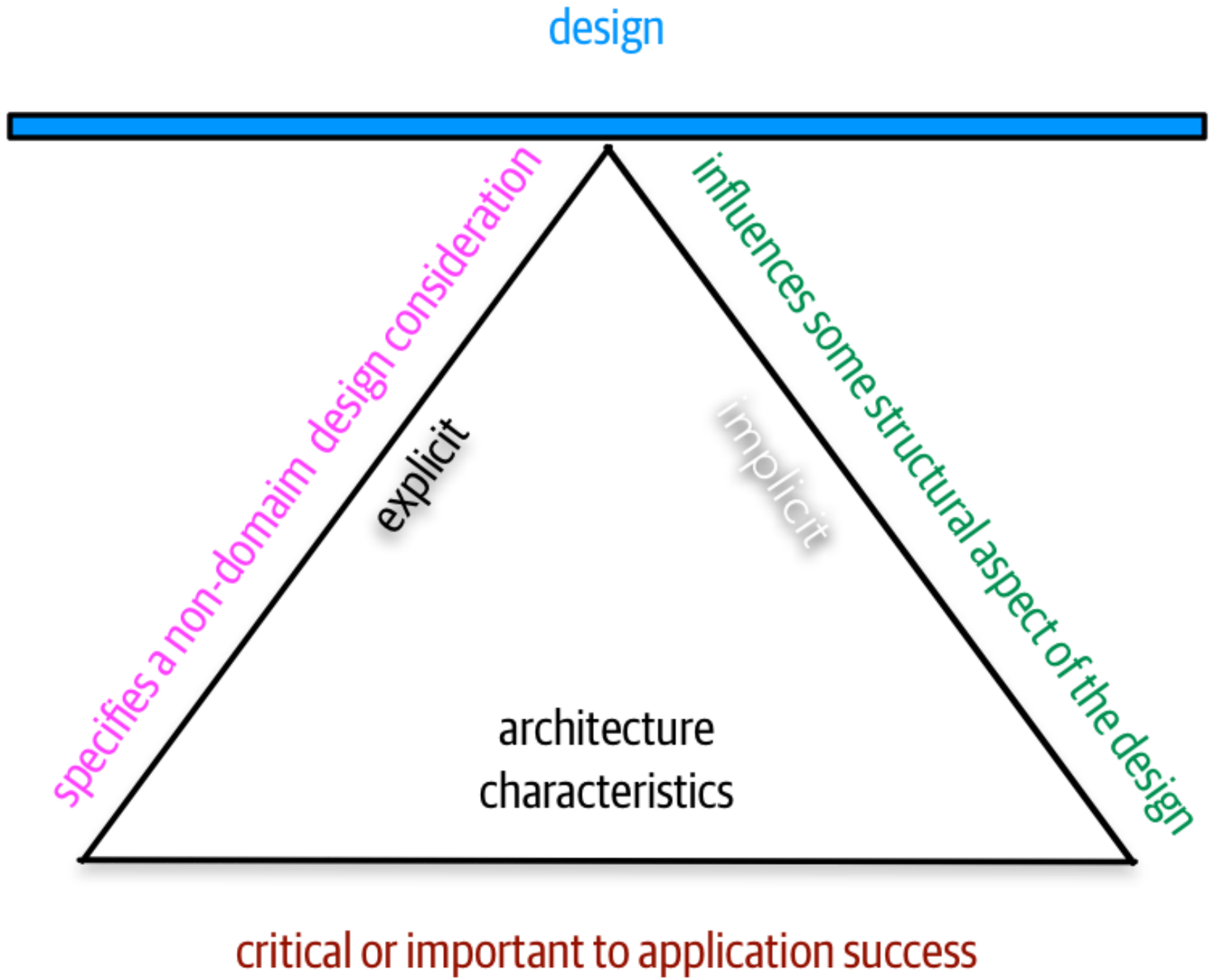

architecture
characteristics


specifies a non-domaim design consideration
architecture
characteristics

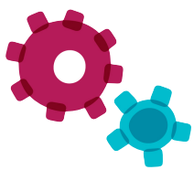

specifies a non-domaim design consideration
influences some structural aspect of the design
architecture
characteristics

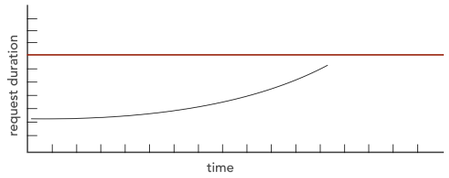

specifies a non-domaim design consideration
influences some structural aspect of the design
architecture
characteristics
critical or important to application success


specifies a non-domaim design consideration
influences some structural aspect of the design
explicit
implicit
architecture
characteristics
critical or important to application success

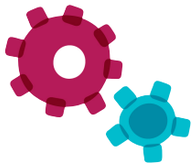

design
specifies a non-domain design consideration
explicit
influences some structural aspect of the design
implicit
architecture
characteristics
critical or important to application success

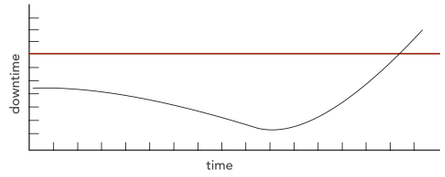
Performance





measure and track average response times (cumulative)





measure and track maximum response times

Performance

https://developers.google.com/web/tools/lighthouse/

https://www.google.com/gmail/about/
Nov 4, 2018, 11:37 AM PST
Emulated Nexus 5X, Simulated Slow 4G network

93	50	43	86	100
Performance	Progressive Web App	Accessibility	Best Practices	SEO

Score scale: 90-100 50-89 0-49

Performance

93

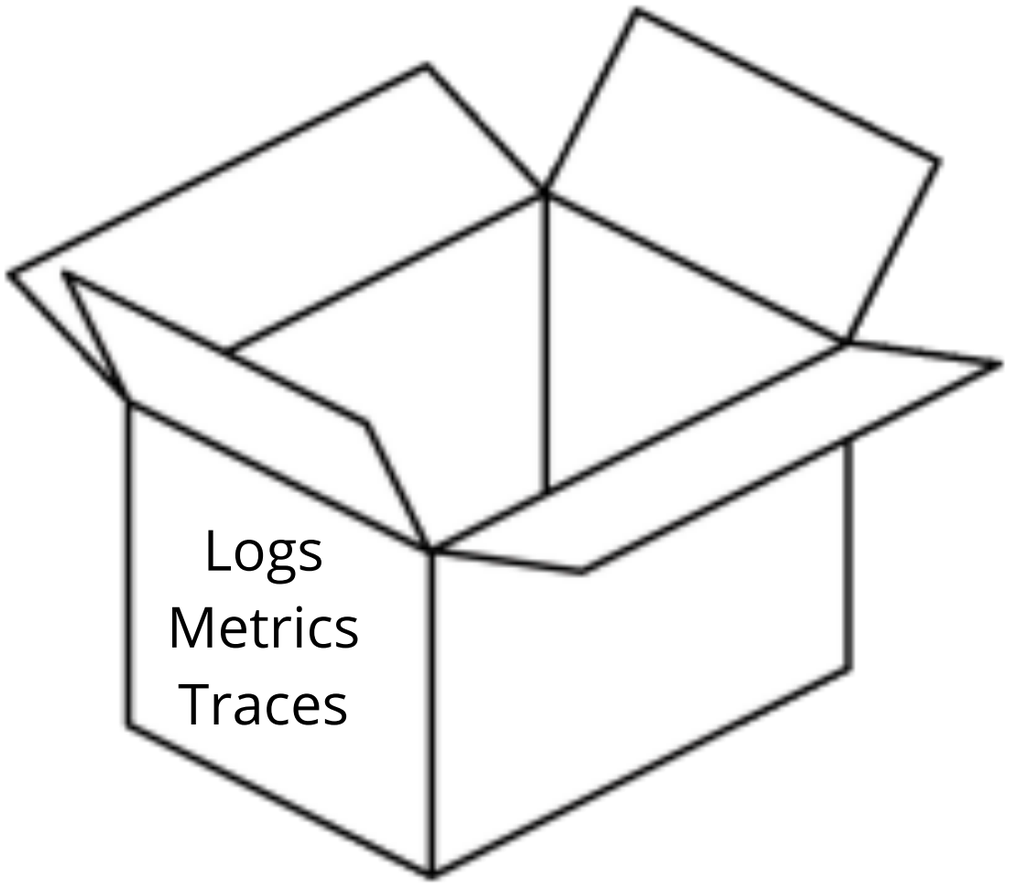
Metrics

First Contentful Paint	1.6 s	First Meaningful Paint	1.6 s
Speed Index	3.3 s	First CPU Idle	3.5 s
Time to Interactive	3.7 s	Estimated Input Latency	40 ms

Values are estimated and may vary.

Observability

Monitoring versus Observability



Logging w/ Decorators

```
@logfunction('charge_account', labels=['accounting','transaction'])
def charge_account(event, context):
    account = event['account']
    account.balance = account.balance - event['charge']
    return account.balance
```

Logging w/ Decorators

```
def logfunction(functionname, labels=[]):
    def logfunction_decorator(func):
        def logfunciton_wrapper(*args, **kwargs):
            raw_data = json.loads(args[0]['body'])
            data = {}
            for l in data:
                data[l] = raw_data.get(l)
                # TODO: Sanatize data here
            logging.info(args[0]['username'], args[0]['request_id'], *lables, data)
            return func(*args, **kwargs)

        return logfunction_wrapper

    return logfunction_decorator
```

question:

How can you measure *agility*?

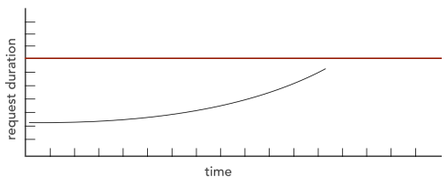
Agility?

deployability

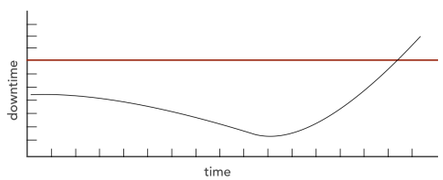
Agility?

Deployability

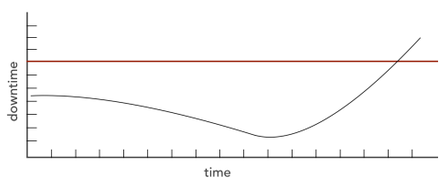
the ease of, risk, and frequency of deployment



measure and track number of actual hours spent to deploy



measure and track failed deployments or errors resulting from deployment

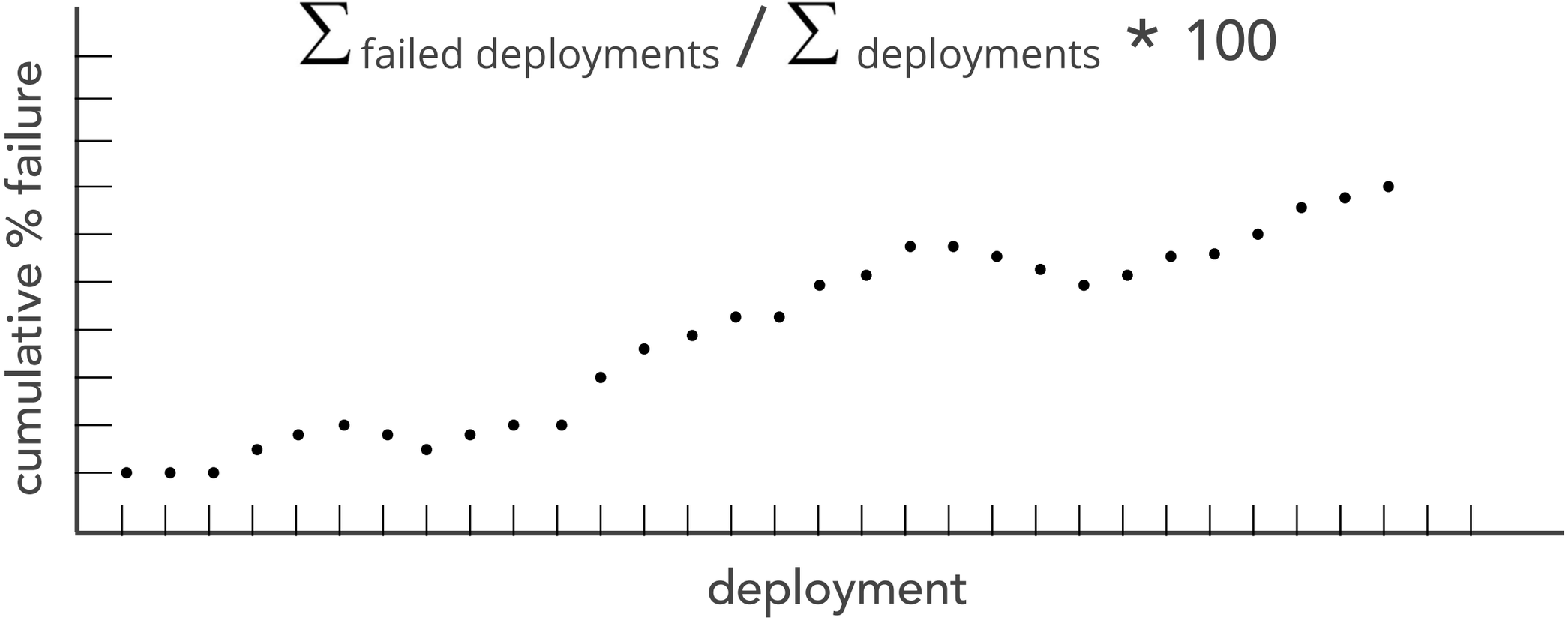


measure and track the frequency of deployment

Deployment Errors/Failures

(per service, domain, or application context)

$$\sum \text{failed deployments} \;/\; \sum \text{deployments} \;*\; 100$$



testability

deployability

Agility?

testability

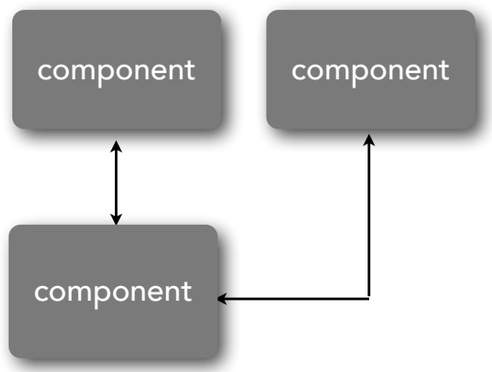
deployability

Agility?

modularity

Component Coupling

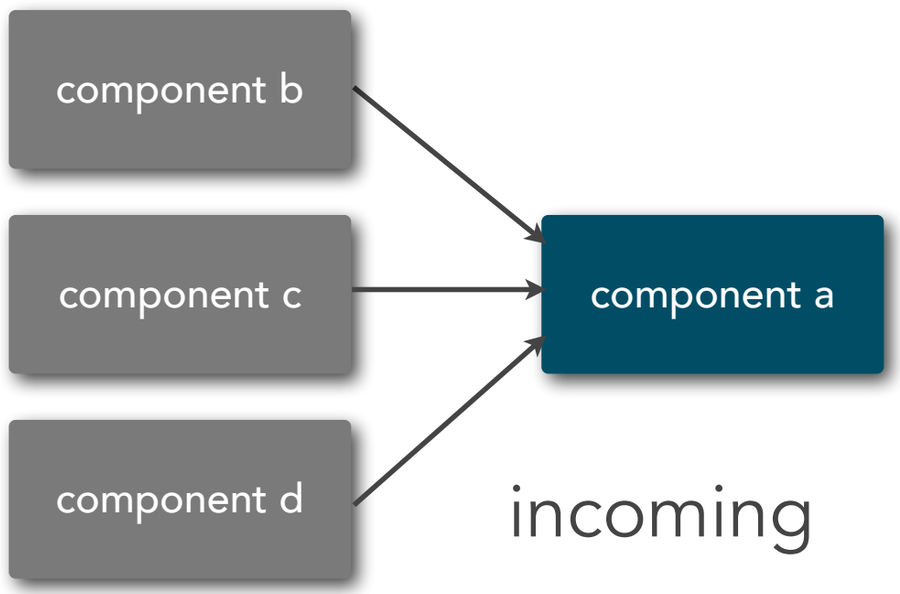
the extent to which components know about each other



Component Coupling

afferent coupling

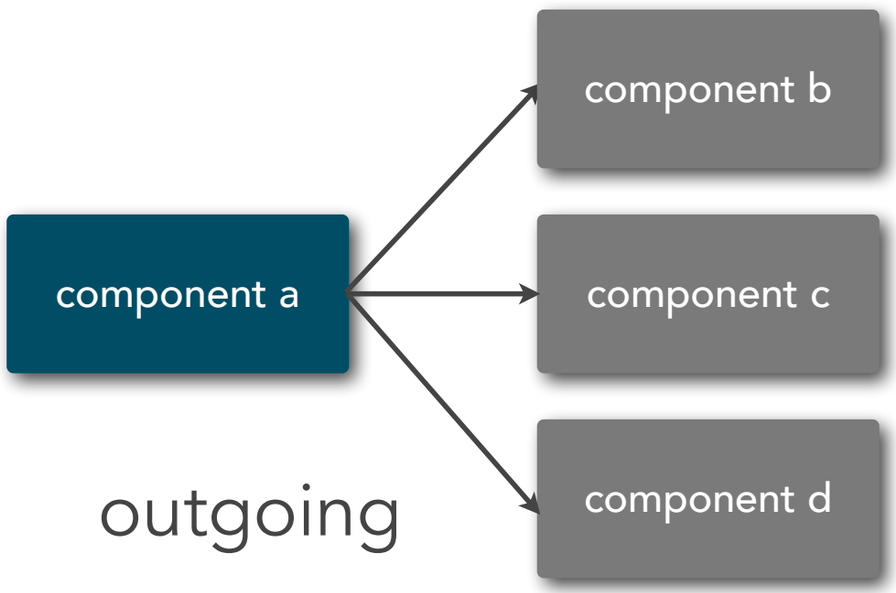
the degree to which other components are dependent on the target component



Component Coupling

efferent coupling

the degree to which the target component
is dependent on other components



dependency metrics

abstractness

$$A = \frac{\sum m^a}{\sum m^c}$$

where:
m^a => abstract elements
m^c => concrete elements

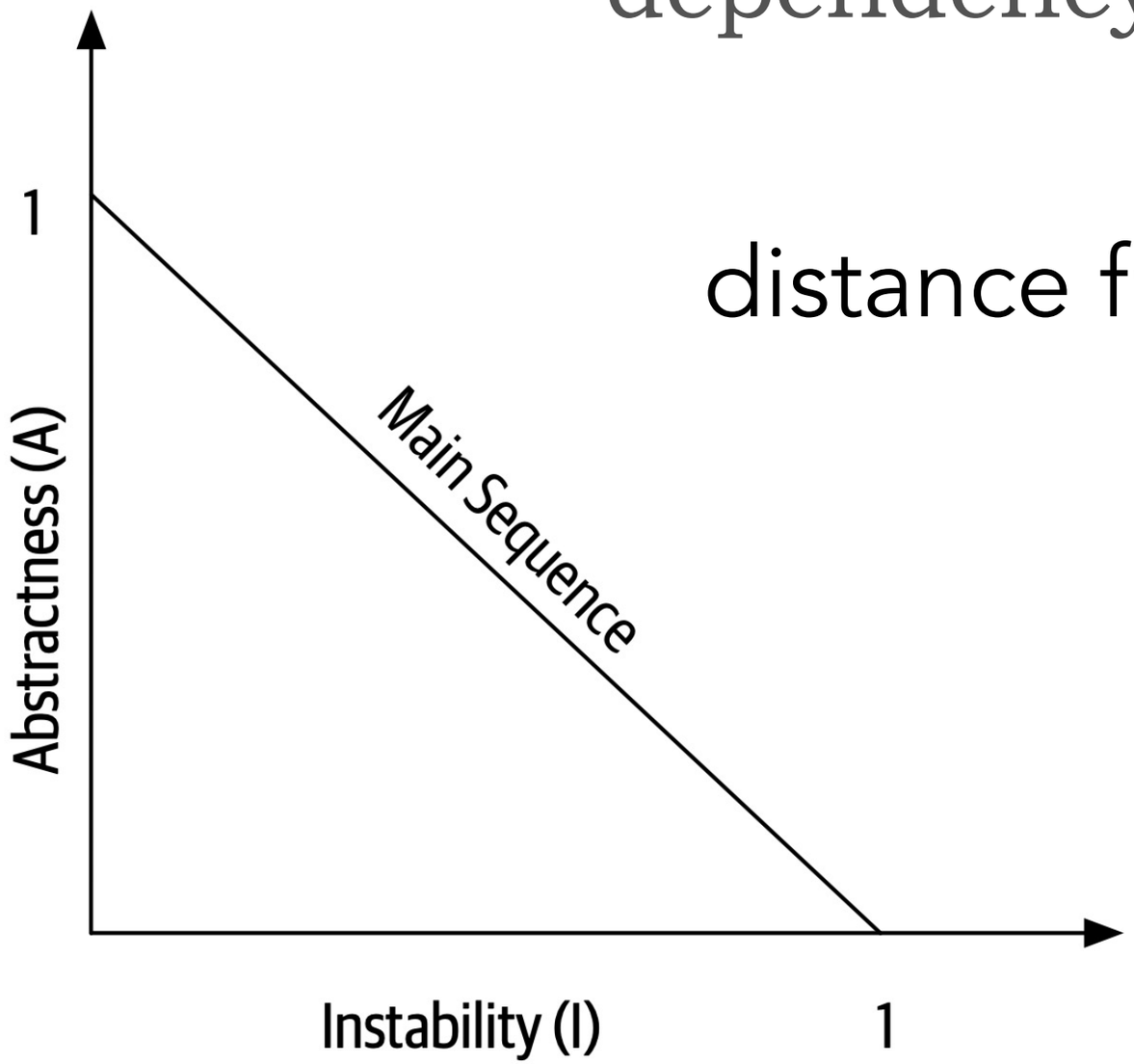
dependency metrics

instability

$$I = \frac{C^e}{C^e + C^a}$$

where:
c^e => efferent coupling
c^a => afferent coupling

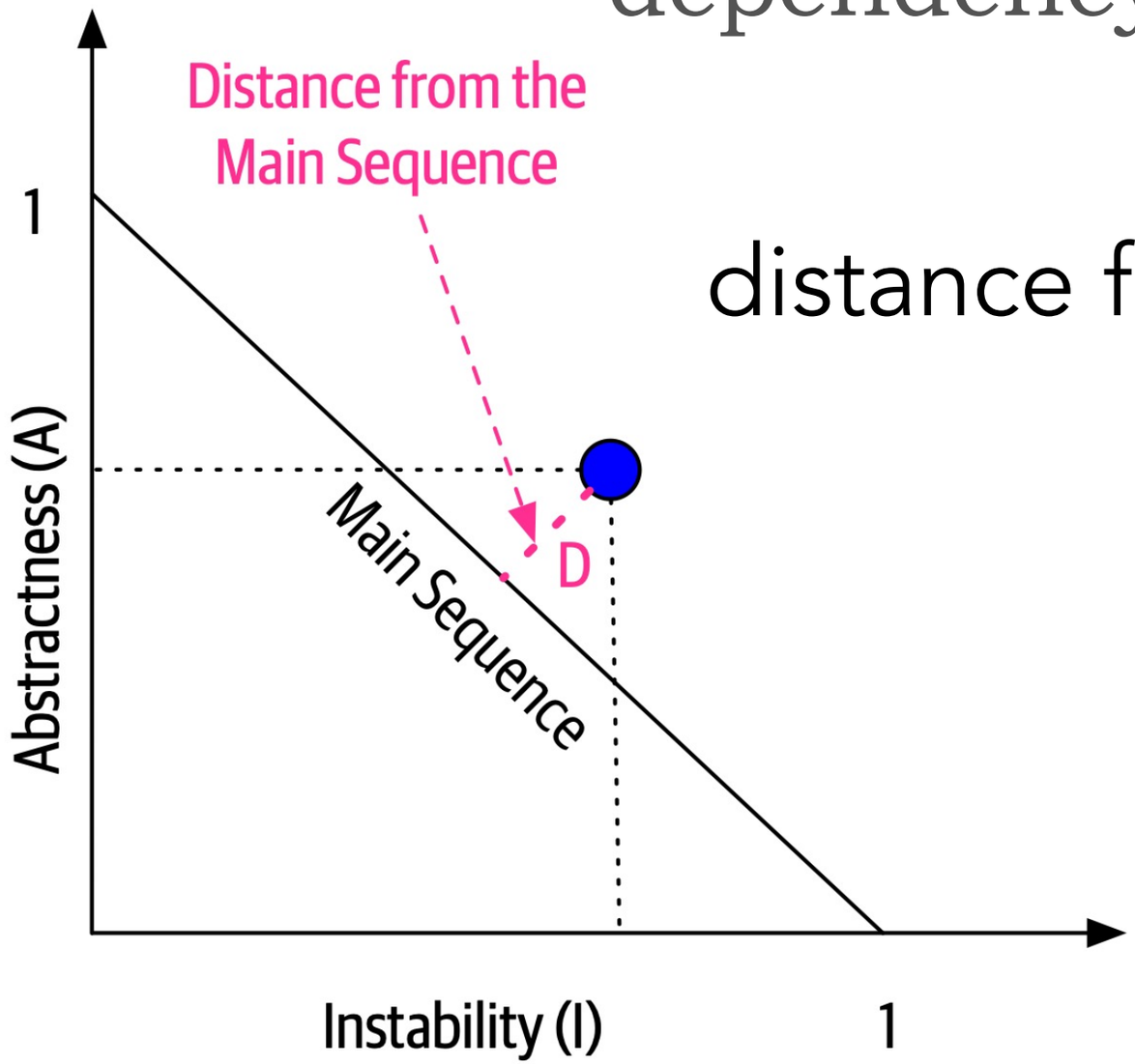
dependency metrics



distance from the main sequence

$$D = |A + I - 1|$$

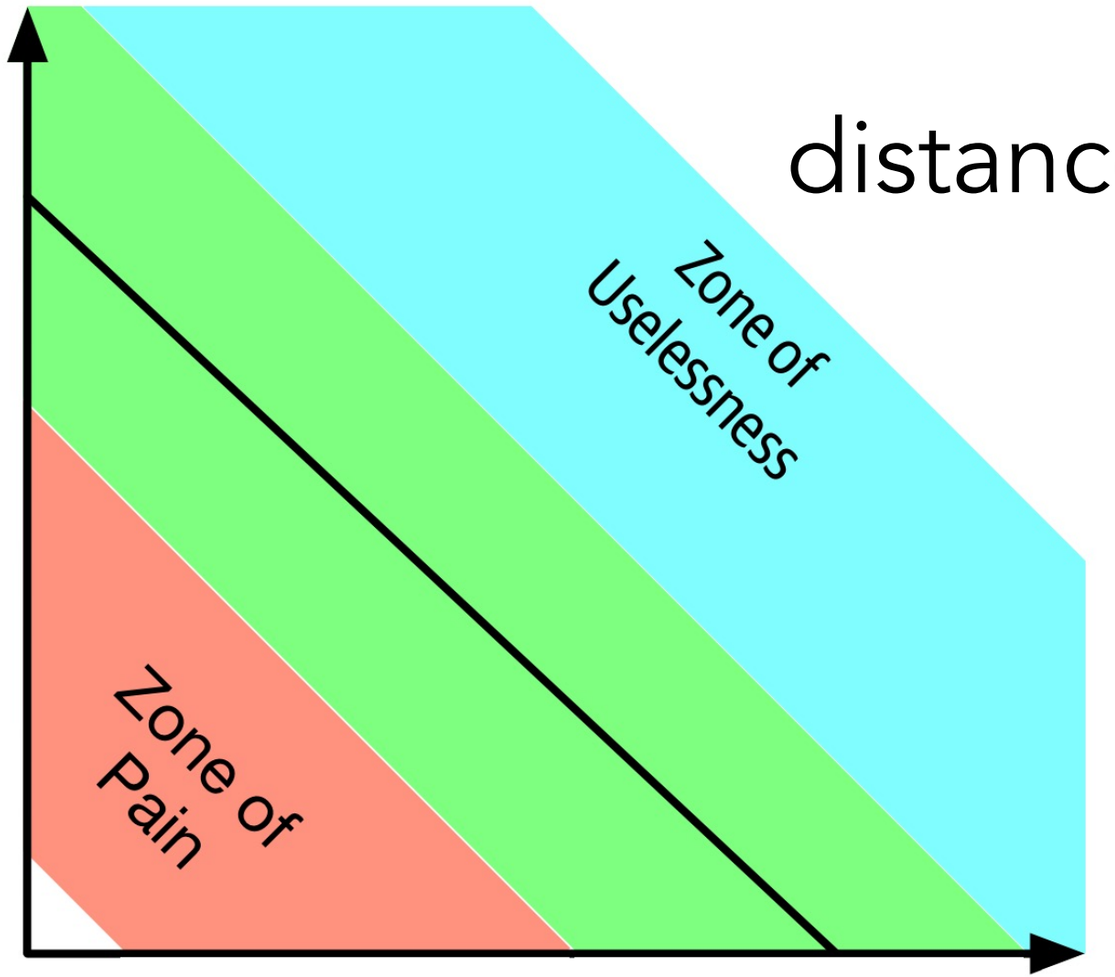
dependency metrics



distance from the main sequence

$$D = |A + I - 1|$$

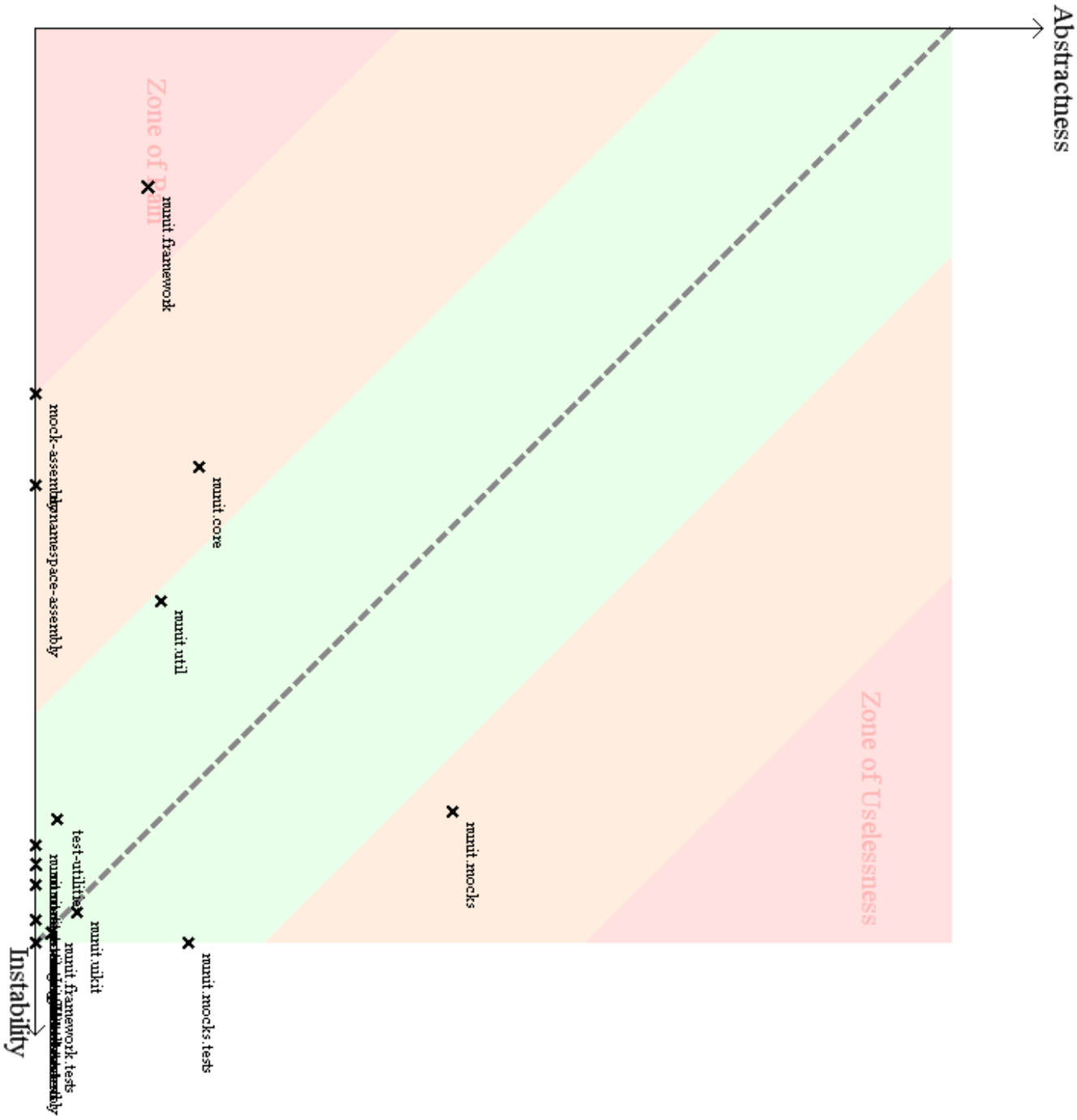
dependency metrics



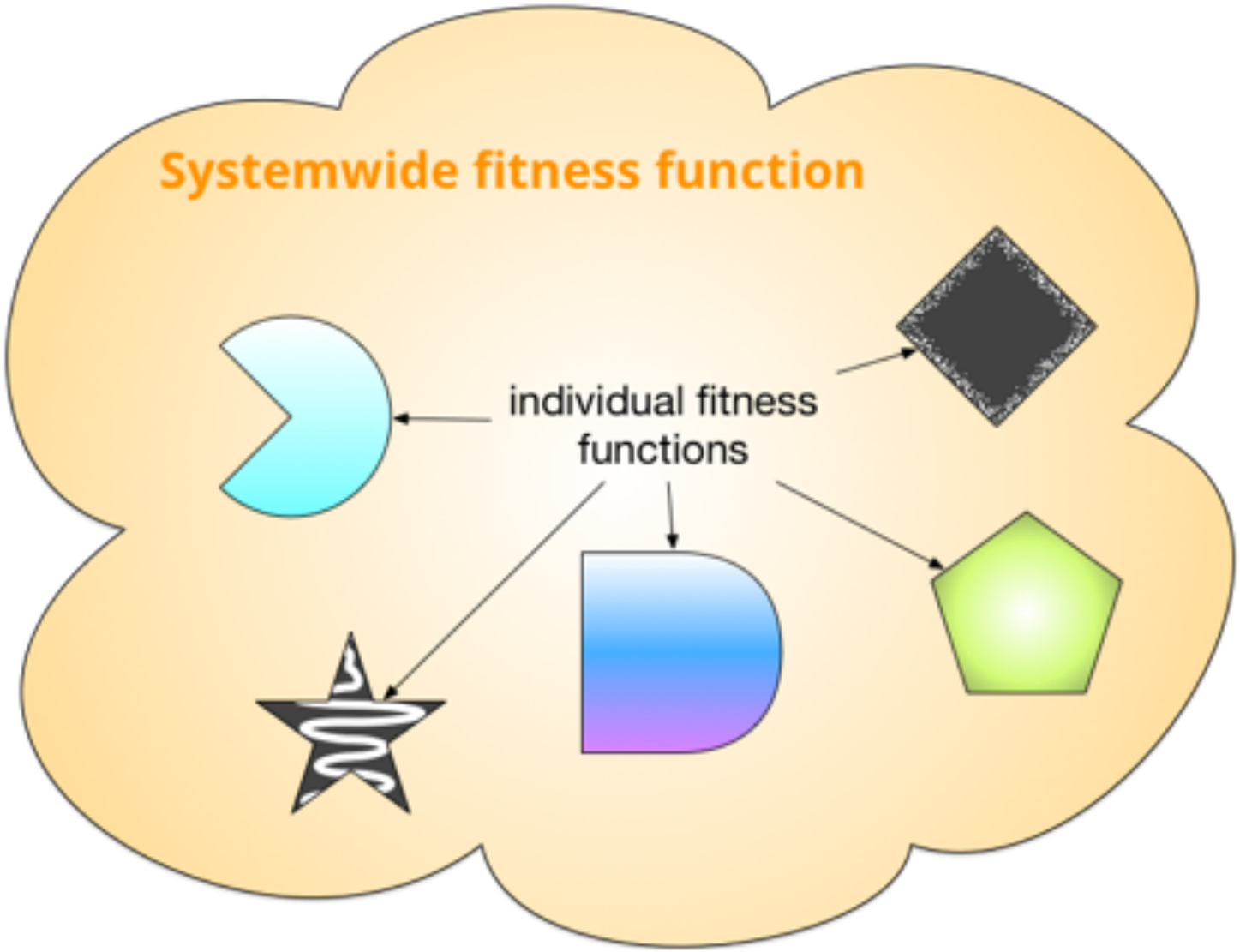
distance from the main sequence

$$D = |A + I - 1|$$

distance from the main sequence

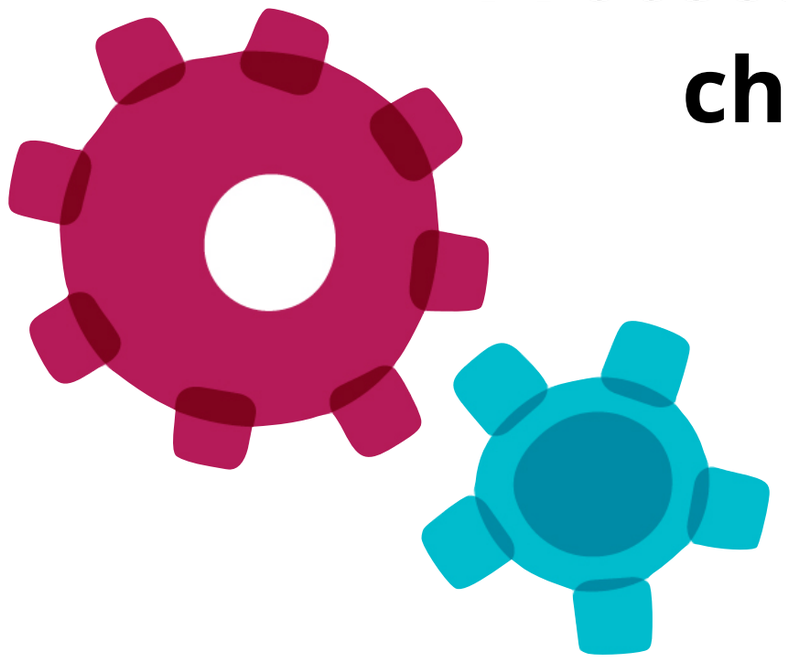


System-wide Fitness Function



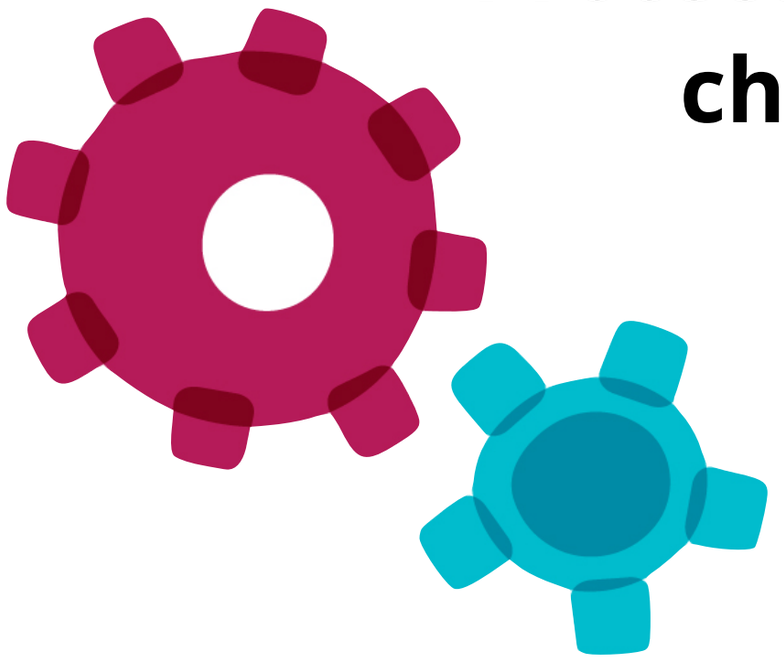
Implementing Fitness Functions

Protecting architectural characteristics

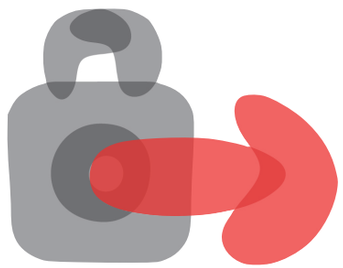


Implementing Fitness Functions

Protecting architectural characteristics



Automating governance



maintainable?

maintainable?

Cyclomatic complexity < 50 for all projects

maintainable?

Cyclomatic complexity < 50 for all projects

Naming conventions

maintainable?

Cyclomatic complexity < 50 for all projects

Naming conventions

maintainable?

(incoming/outgoing)

Controlled afferent/efferent coupling

Cyclomatic complexity < 50 for all projects

Naming conventions

immutability

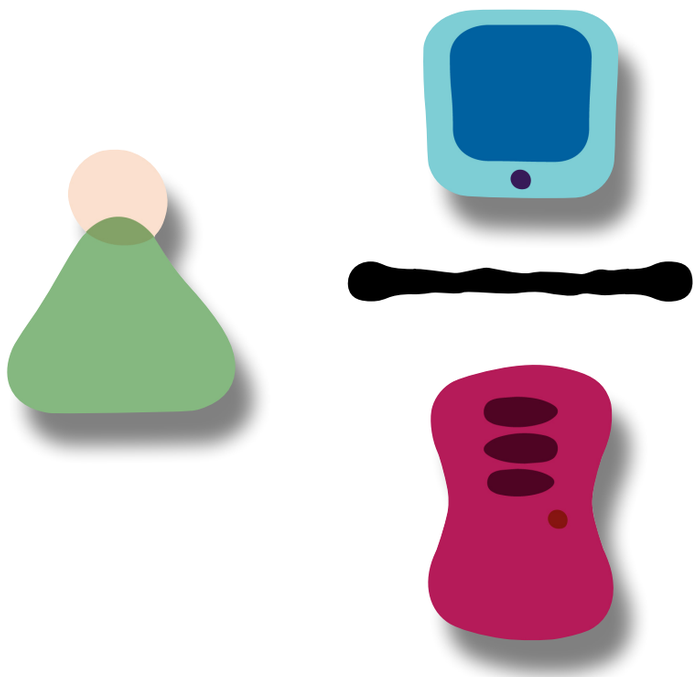
maintainable?

(incoming/outgoing)

Controlled afferent/efferent coupling

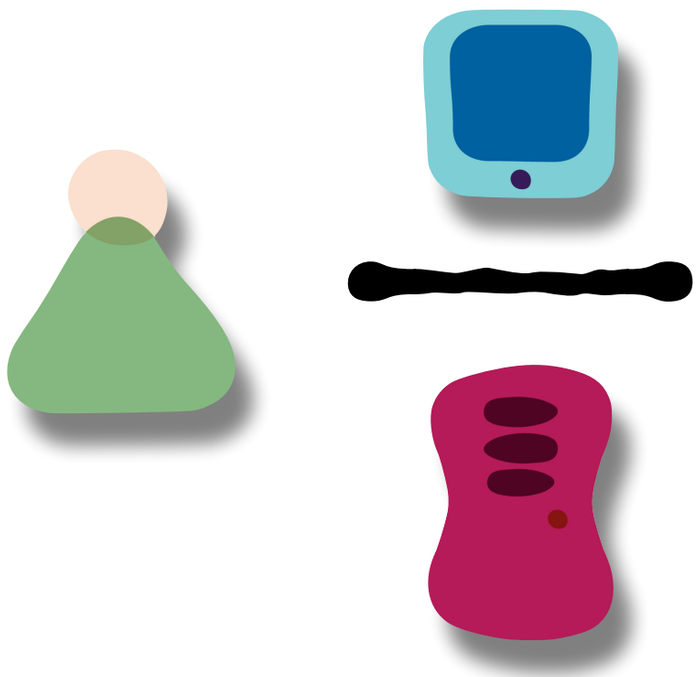
Penultimate

Governing Code Quality



Penultimate

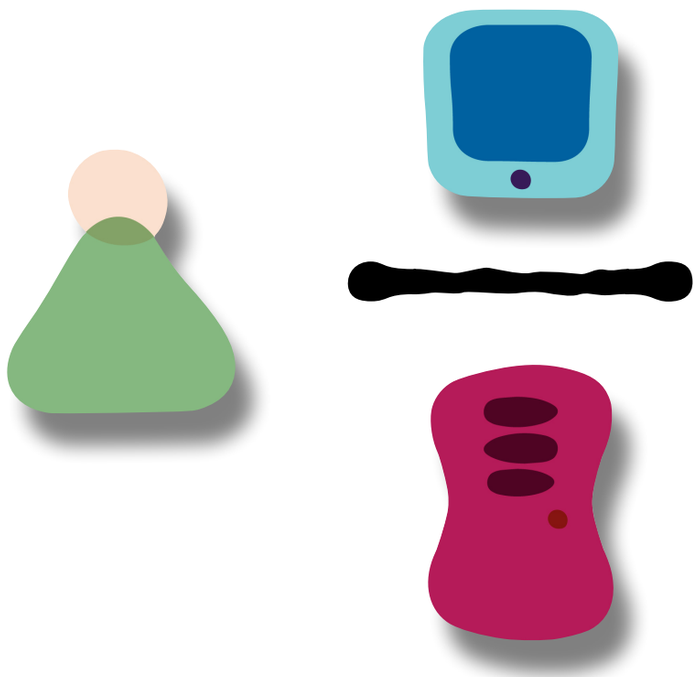
Governing Code Quality

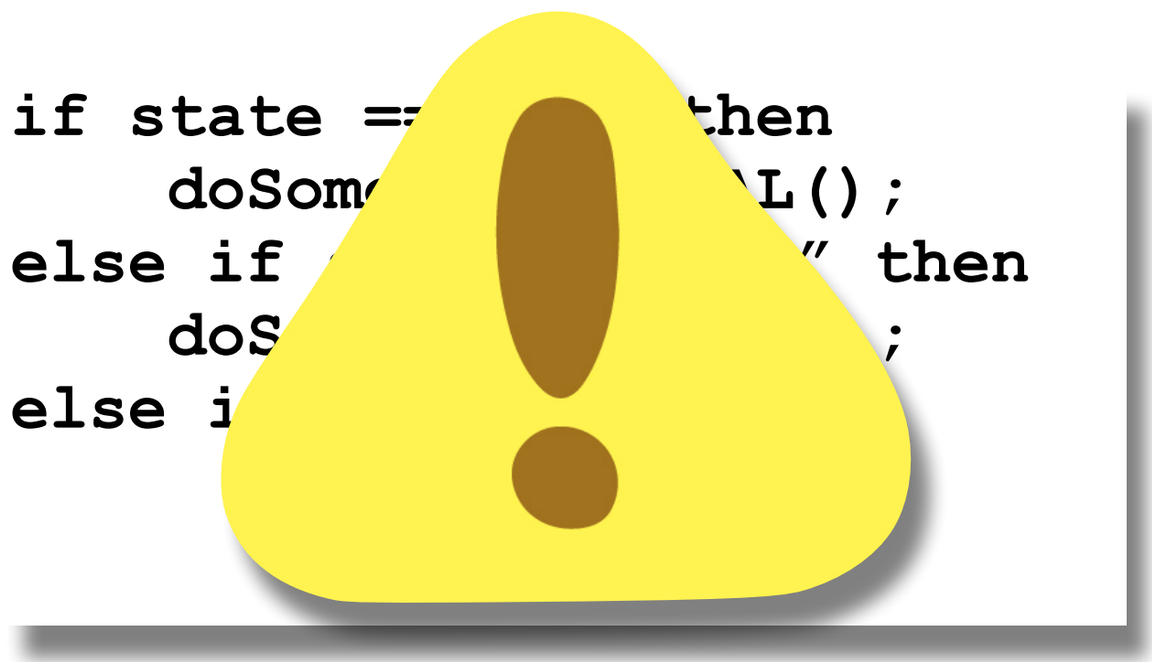


```
if state == “AL” then
    doSomethingForAL();
else if state == “GA” then
    doSomethingForGA();
else if …
```

Penultimate

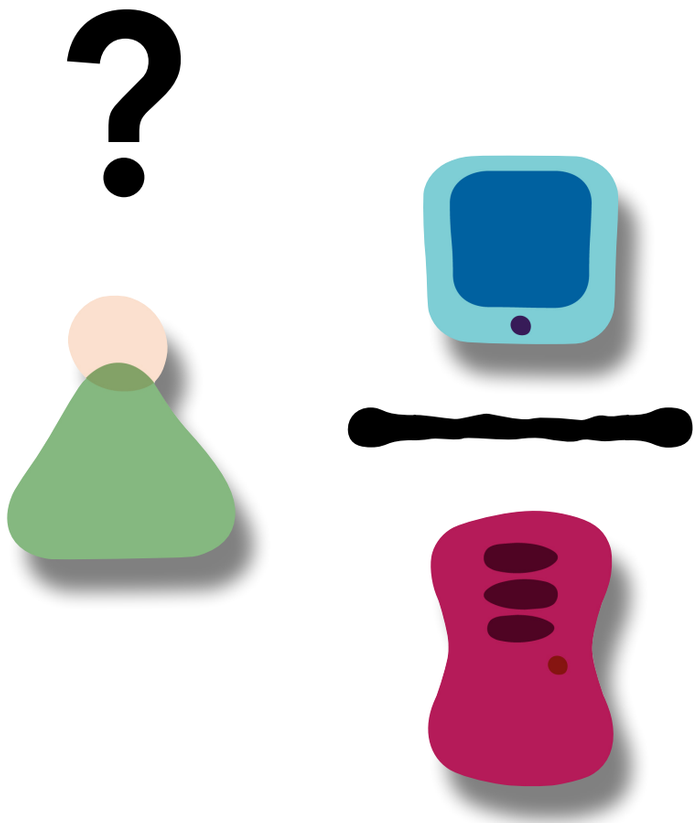
Governing Code Quality





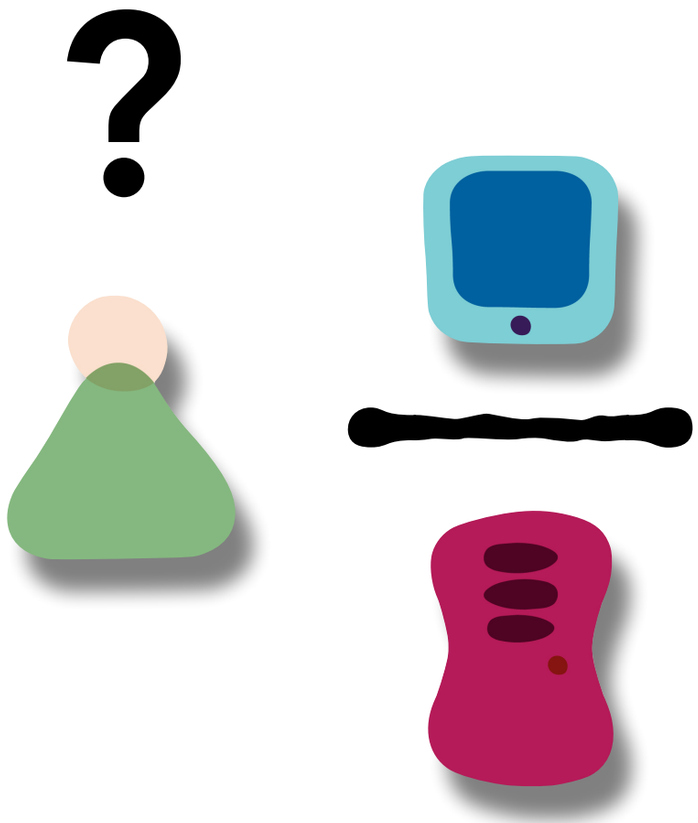
Penultimate

Governing Code Quality



Penultimate

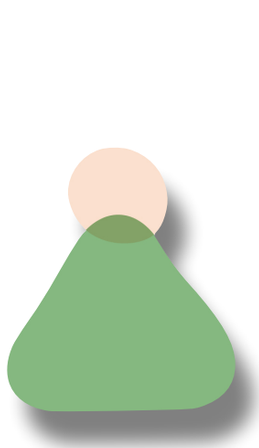
Governing Code Quality

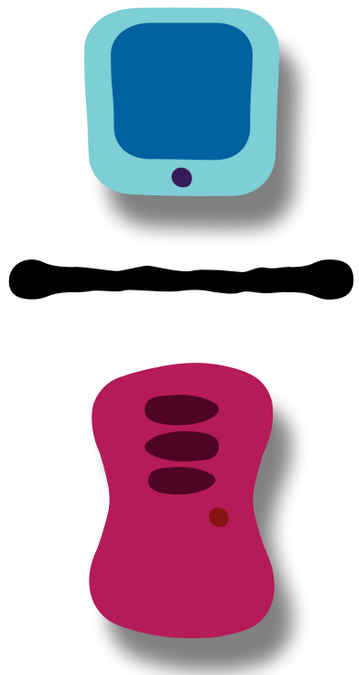


```
if state == "AL" then
    doSomethingForAL();
else if state == "GA" then
    doSomethingForGA();
else if …
```

Penultimate

Governing Code Quality



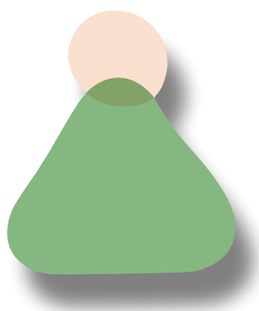


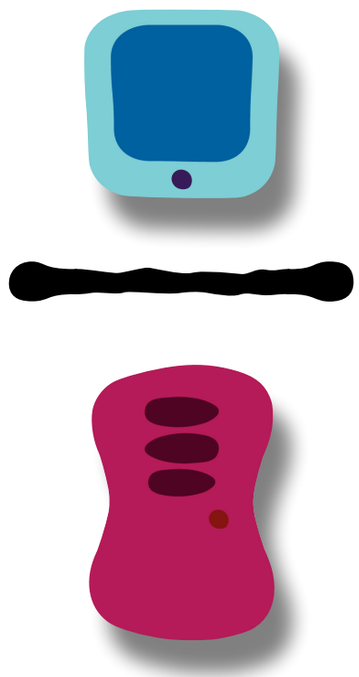
```
if state ==                    then
     doSome                    AL();
else if s                      A" then
     doSo                      );
else if
```

Penultimate

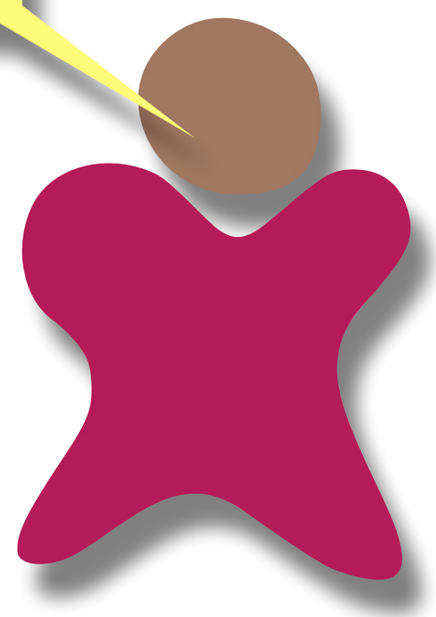
Governing Code Quality

?!



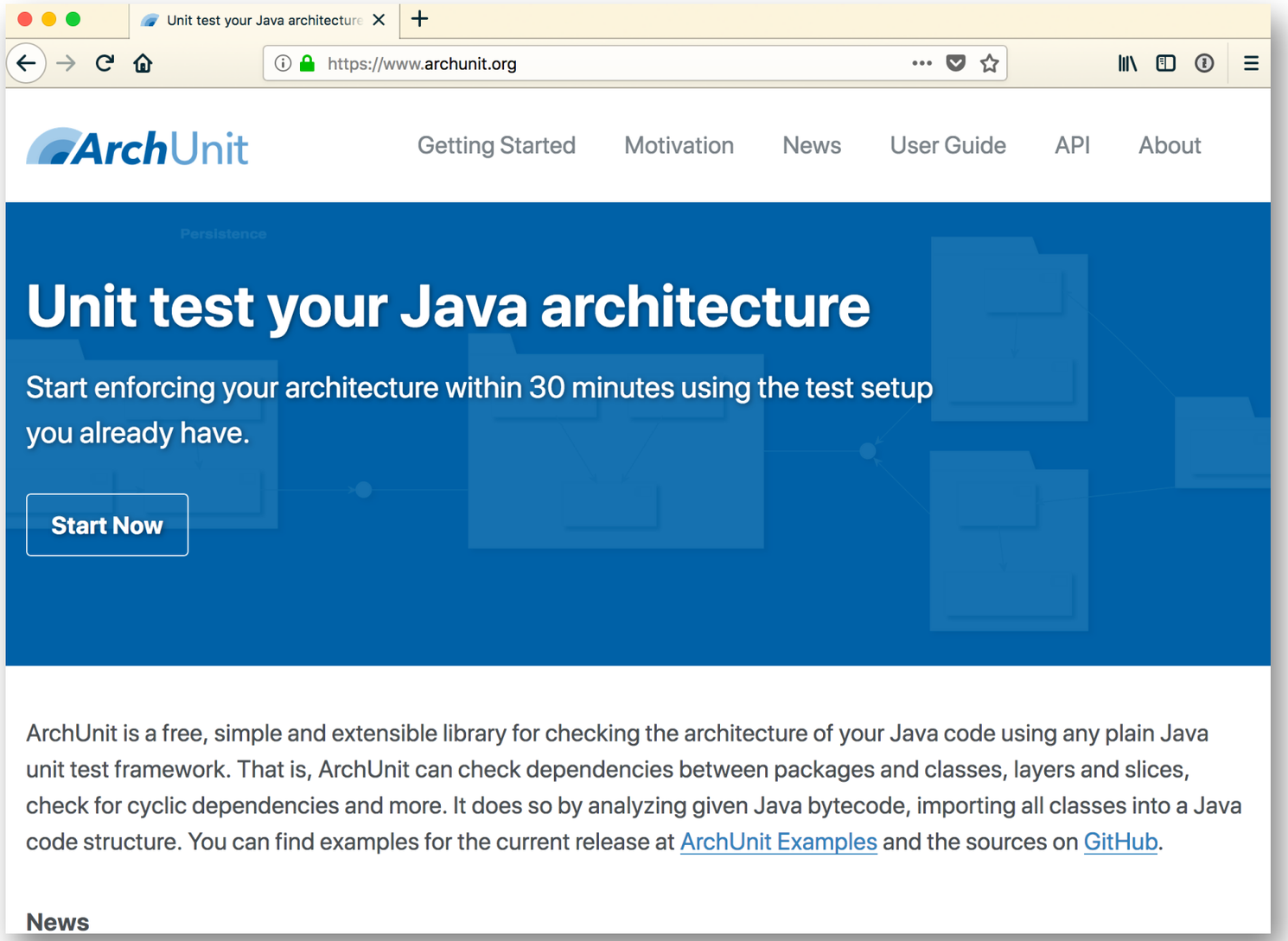


Strategy Design Pattern



ArchUnit

https://www.archunit.org/



ArchUnit

https://www.archunit.org/

```java
import static com.tngtech.archunit.lang.syntax.ArchRuleDefinition.noClasses;
import static com.tngtech.archunit.library.GeneralCodingRules.ACCESS_STANDARD_STREAMS;
import static com.tngtech.archunit.library.GeneralCodingRules.NO_CLASSES_SHOULD_ACCESS_STANDARD_STREAMS;
import static com.tngtech.archunit.library.GeneralCodingRules.NO_CLASSES_SHOULD_THROW_GENERIC_EXCEPTIONS;
import static com.tngtech.archunit.library.GeneralCodingRules.NO_CLASSES_SHOULD_USE_JAVA_UTIL_LOGGING;

public class CodingRulesTest {
    private JavaClasses classes;

    @Before
    public void setUp() throws Exception {
        classes = new ClassFileImporter().importPackagesOf(ClassViolatingCodingRules.class);
    }

    @Test
    public void classes_should_not_access_standard_streams_defined_by_hand() {
        noClasses().should(ACCESS_STANDARD_STREAMS).check(classes);
    }

    @Test
    public void classes_should_not_access_standard_streams_from_library() {
        NO_CLASSES_SHOULD_ACCESS_STANDARD_STREAMS.check(classes);
    }

    @Test
    public void classes_should_not_throw_generic_exceptions() {
        NO_CLASSES_SHOULD_THROW_GENERIC_EXCEPTIONS.check(classes);
    }

    @Test
    public void classes_should_not_use_java_util_logging() {
        NO_CLASSES_SHOULD_USE_JAVA_UTIL_LOGGING.check(classes);
    }
}
```

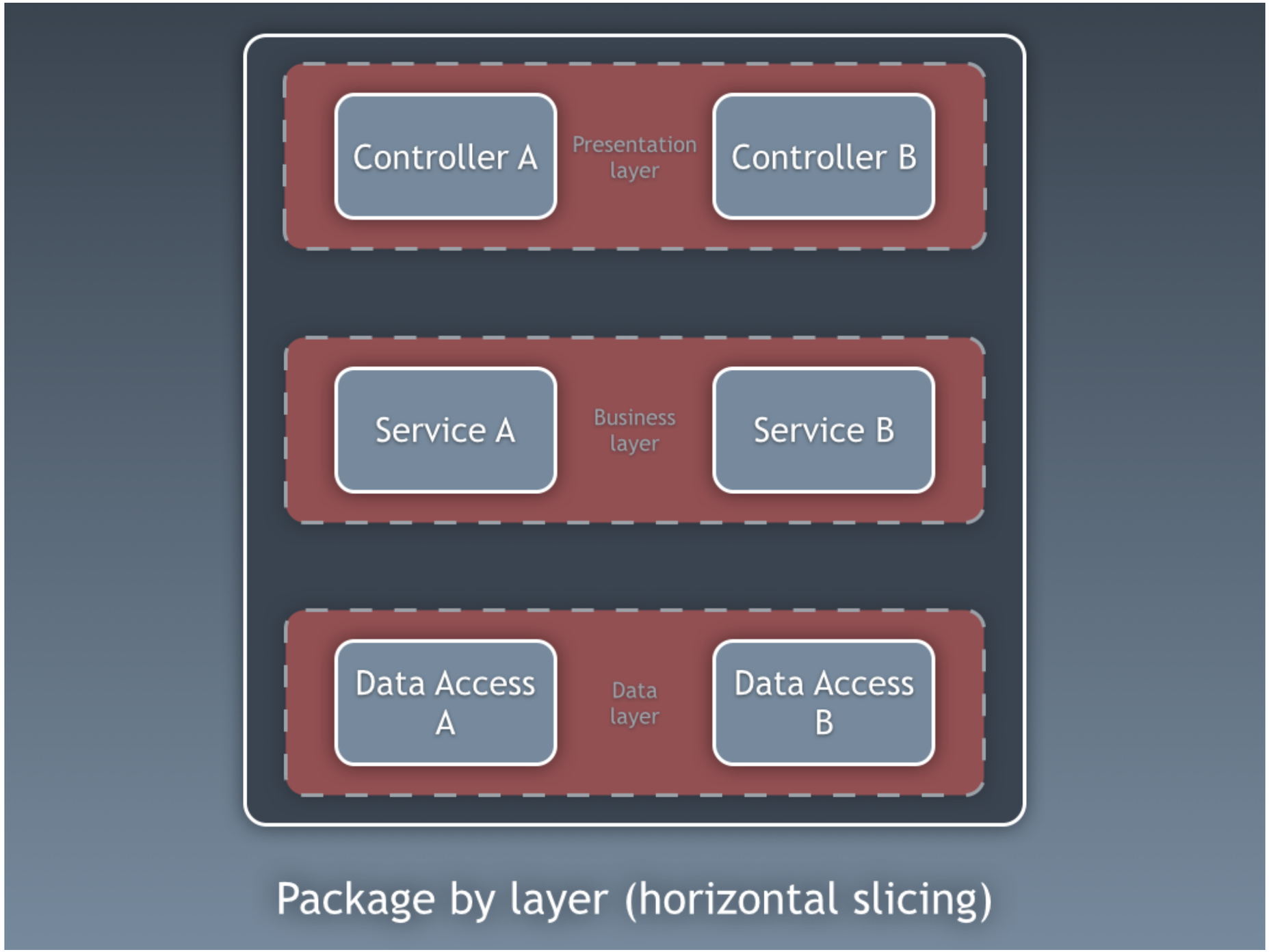
coding rules

ArchUnit

https://www.archunit.org/

```java
public class InterfaceRules {

    @Test
    public void interfaces_should_not_have_names_ending_with_the_word_interface() {
        JavaClasses classes = new ClassFileImporter().importClasses(
                SomeBusinessInterface.class,
                SomeDao.class
        );

        noClasses().that().areInterfaces().should().haveNameMatching(".*Interface").check(classes);
    }

    @Test
    public void interfaces_should_not_have_simple_class_names_ending_with_the_word_interface() {
        JavaClasses classes = new ClassFileImporter().importClasses(
                SomeBusinessInterface.class,
                SomeDao.class
        );

        noClasses().that().areInterfaces().should().haveSimpleNameContaining("Interface").check(classes);
    }

    @Test
    public void interfaces_must_not_be_placed_in_implementation_packages() {
        JavaClasses classes = new ClassFileImporter().importPackagesOf(SomeInterfacePlacedInTheWrongPackage.class);

        noClasses().that().resideInAPackage("..impl..").should().beInterfaces().check(classes);
    }
}
```

interface rules


Controller A
Presentation layer
Controller B
Service A
Business layer
Service B
Data Access A
Data layer
Data Access B
Package by layer (horizontal slicing)

ArchUnit

https://www.archunit.org/

```java
public class LayerDependencyRulesTest {
    private JavaClasses classes;

    @Before
    public void setUp() throws Exception {
        classes = new ClassFileImporter().importPackagesOf(ClassViolatingCodingRules.class);
    }

    @Test
    public void services_should_not_access_controllers() {
        noClasses().that().resideInAPackage("..service..")
                .should().accessClassesThat().resideInAPackage("..controller..").check(classes);
    }

    @Test
    public void persistence_should_not_access_services() {
        noClasses().that().resideInAPackage("..persistence..")
                .should().accessClassesThat().resideInAPackage("..service..").check(classes);
    }

    @Test
    public void services_should_only_be_accessed_by_controllers_or_other_services() {
        classes().that().resideInAPackage("..service..")
                .should().onlyBeAccessed().byAnyPackage("..controller..", "..service..").check(classes);
    }
}
```

layer dependency

ArchUnit

https://www.archunit.org/

```java
@Test
public void third_party_class_should_only_be_instantiated_via_workaround() {
    classes().should(notCreateProblematicClassesOutsideOfWorkaroundFactory()
            .as(THIRD_PARTY_CLASS_RULE_TEXT))
            .check(classes);
}

private ArchCondition<JavaClass> notCreateProblematicClassesOutsideOfWorkaroundFactory() {
    DescribedPredicate<JavaCall<?>> constructorCallOfThirdPartyClass =
            target(is(constructor())).and(targetOwner(is(assignableTo(ThirdPartyClassWithProblem.class))));

    DescribedPredicate<JavaCall<?>> notFromWithinThirdPartyClass =
            originOwner(is(not(assignableTo(ThirdPartyClassWithProblem.class)))).forSubType();

    DescribedPredicate<JavaCall<?>> notFromWorkaroundFactory =
            originOwner(is(not(equivalentTo(ThirdPartyClassWorkaroundFactory.class)))).forSubType();

    DescribedPredicate<JavaCall<?>> targetIsIllegalConstructorOfThirdPartyClass =
            constructorCallOfThirdPartyClass.
                    and(notFromWithinThirdPartyClass).
                    and(notFromWorkaroundFactory);

    return never(callCodeUnitWhere(targetIsIllegalConstructorOfThirdPartyClass));
}
```

governance

NetArchTest

https://github.com/BenMorris/NetArchTest/

```
// Controllers should not directly reference repositories
var result = Types.InCurrentDomain()
    .That()
    .ResideInNamespace("NetArchTest.SampleLibrary.Presentation")
    .ShouldNot()
    .HaveDependencyOn("NetArchTest.SampleLibrary.Data")
    .GetResult().IsSuccessful;
```

NetArchTest

https://github.com/BenMorris/NetArchTest/

```
// Only classes in the data namespace can have a dependency on System.Data
result = Types.InCurrentDomain()
    .That().HaveDependencyOn("System.Data")
    .And().ResideInNamespace(("ArchTest"))
    .Should().ResideInNamespace(("NetArchTest.SampleLibrary.Data"))
    .GetResult().IsSuccessful;
```

NetArchTest

https://github.com/BenMorris/NetArchTest/

```
// All the classes in the data namespace should implement IRepository
result = Types.InCurrentDomain()
    .That().ResideInNamespace(("NetArchTest.SampleLibrary.Data"))
    .And().AreClasses()
    .Should().ImplementInterface(typeof(IRepository<>))
    .GetResult().IsSuccessful;

// Classes that implement IRepository should have the suffix "Repository"
result = Types.InCurrentDomain()
    .That().ResideInNamespace(("NetArchTest.SampleLibrary.Data"))
    .And().AreClasses()
    .Should().HaveNameEndingWith("Repository")
    .GetResult().IsSuccessful;

// Classes that implement IRepository must reside in the Data namespace
result = Types.InCurrentDomain()
    .That().ImplementInterface(typeof(IRepository<>))
    .Should().ResideInNamespace(("NetArchTest.SampleLibrary.Data"))
    .GetResult().IsSuccessful;

// All the service classes should be sealed
result = Types.InCurrentDomain()
    .That().ImplementInterface(typeof(IWidgetService))
    .Should().BeSealed()
    .GetResult().IsSuccessful;
```

Legality of Open Source Libraries

Penultimate

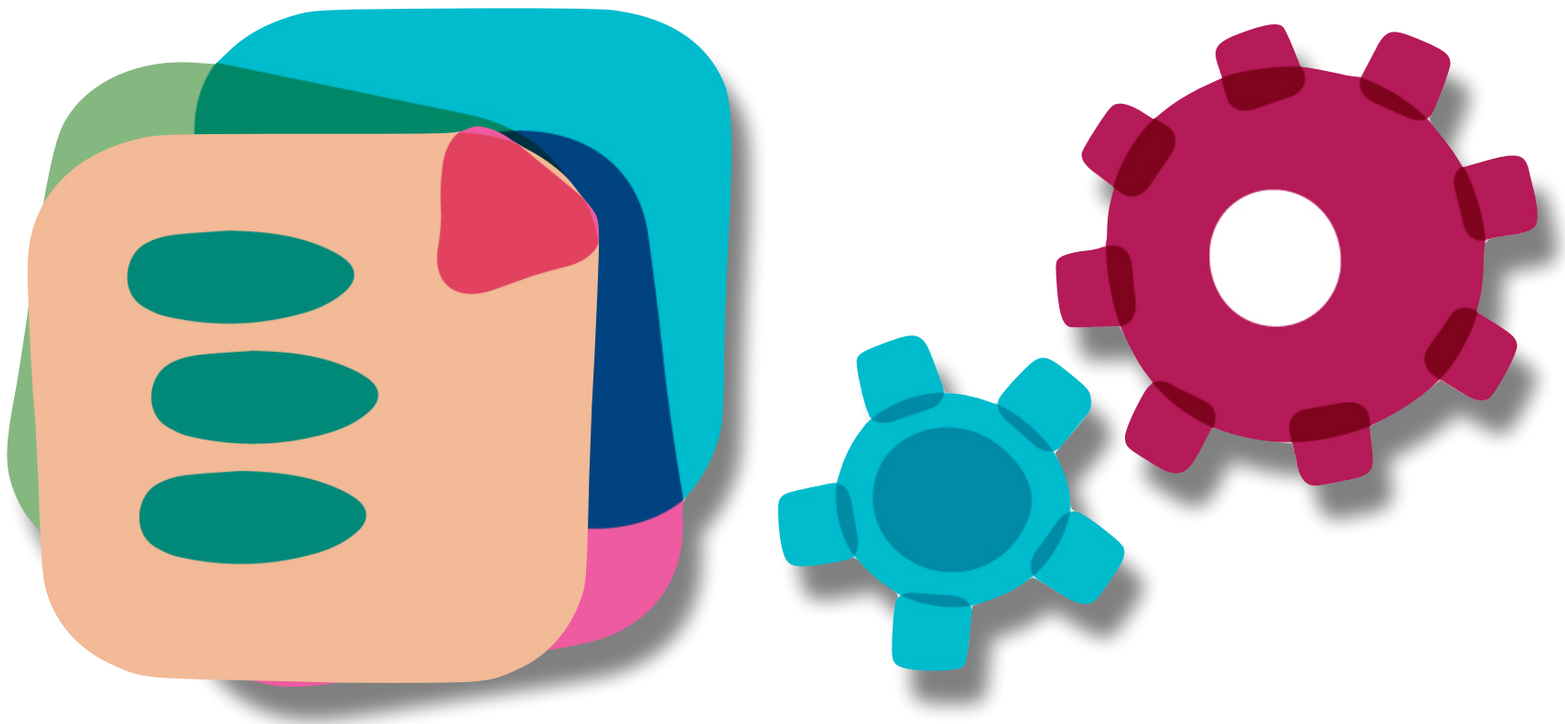
Legality of Open Source Libraries

Penultimate

Legality of Open Source Libraries

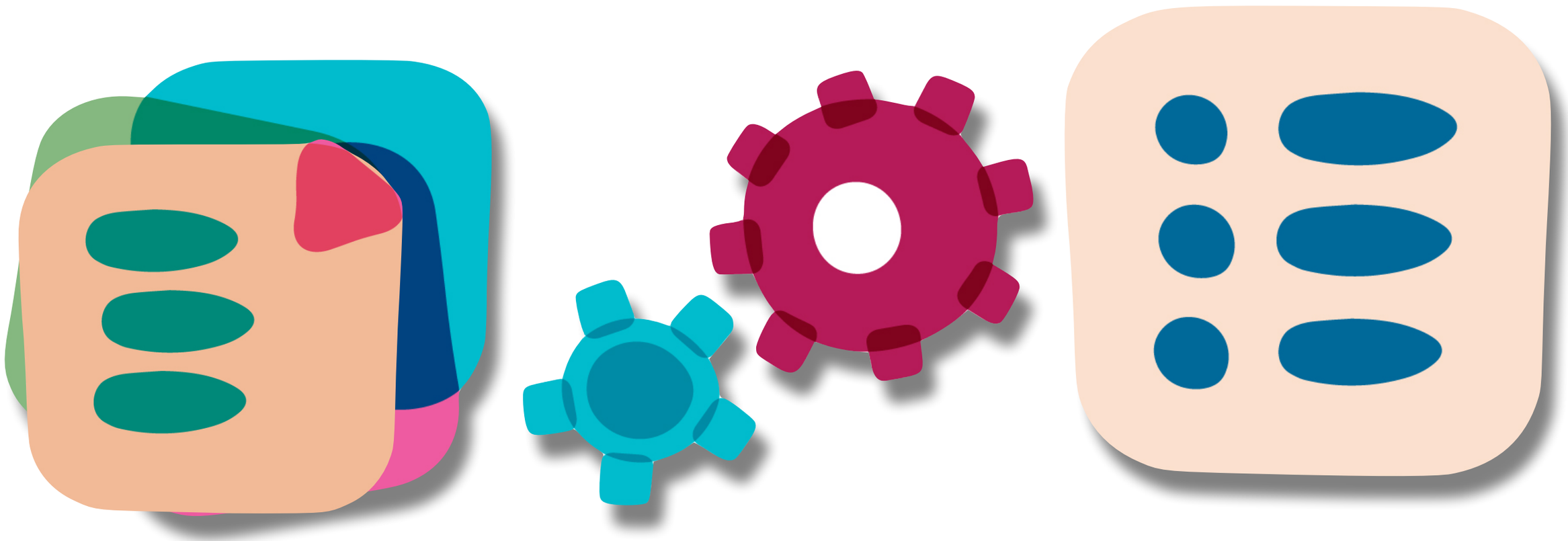
Penultimate

Legality of Open Source Libraries



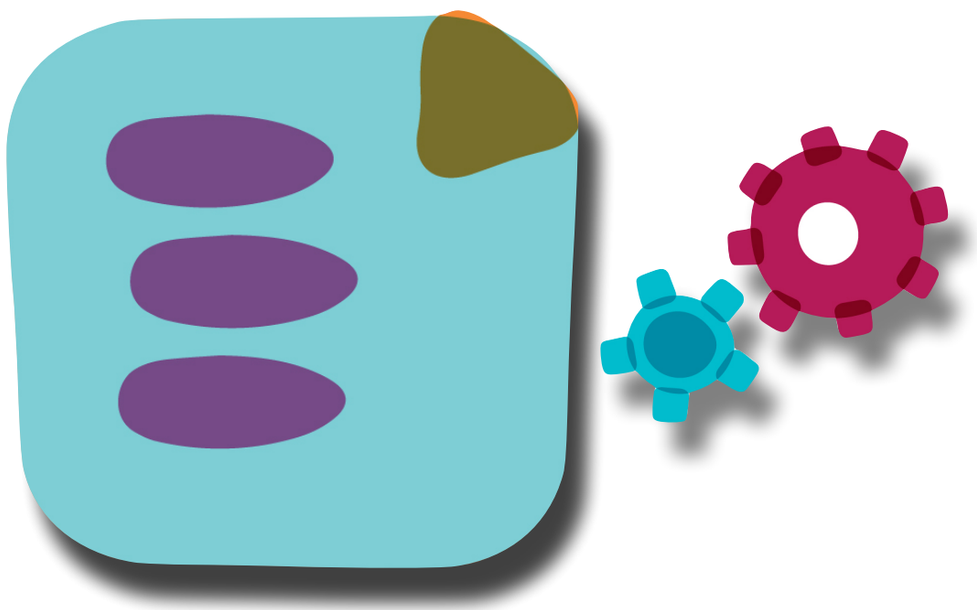
Penultimate

Legality of Open Source Libraries



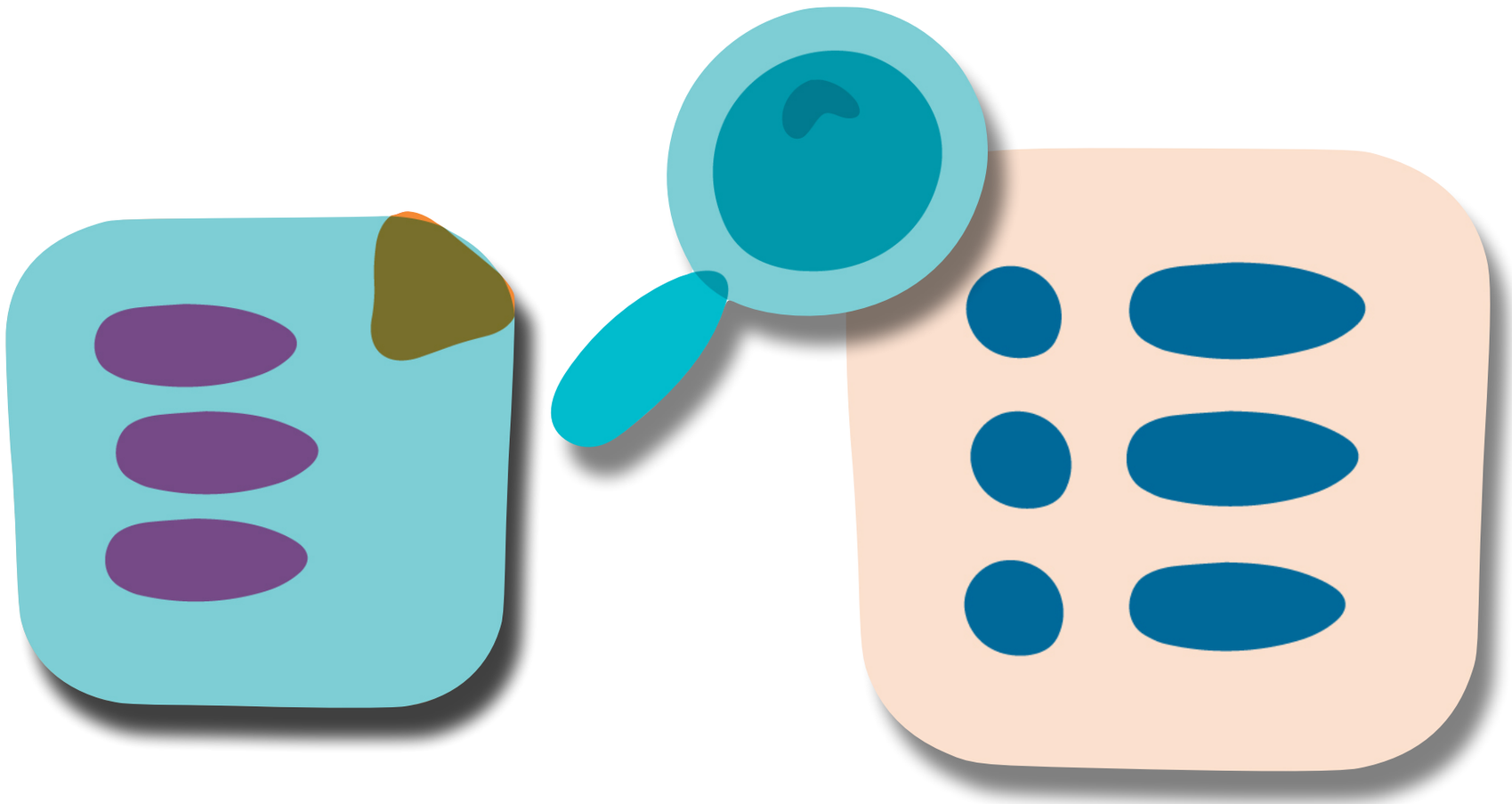
Penultimate

Legality of Open Source Libraries



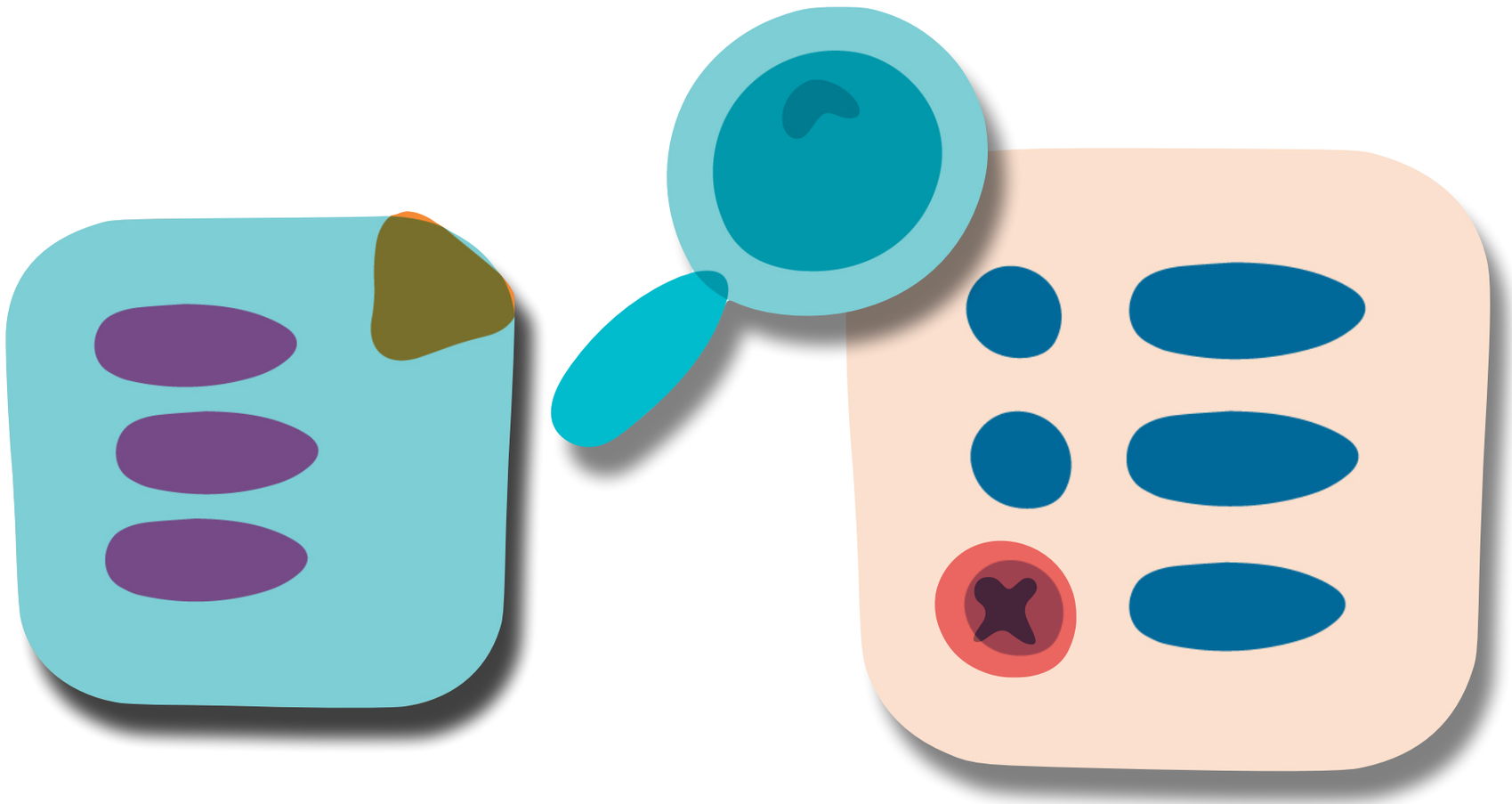
Penultimate

Legality of Open Source Libraries



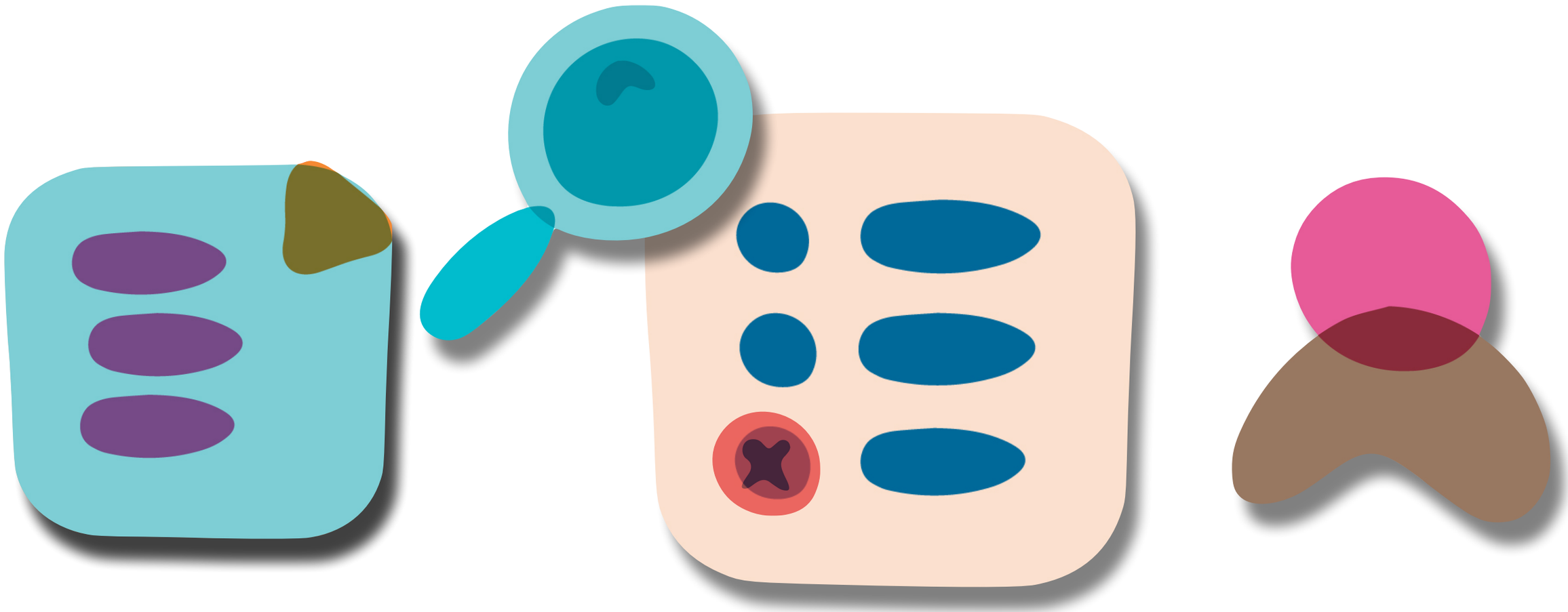
Penultimate

Legality of Open Source Libraries



Penultimate

Legality of Open Source Libraries



Penultimate

Fitness Function:

Configure Some of the Things All of the Time

PenultimateWidgets has been burned in the past by having hard-coded constants in several places in their code base, governing both business cases (thresholds, maximums, etc.) and operational concerns. Architects need a way to prevent developers from accidentally hard-coding critical values.

Fitness Function:

Configure Some of the Things All of the Time

- Upon check-in:
  - scan the list of dynamic values in the code base and see if any of them are assigned to (or their setter is called) or a new instance is instantiated via a constructor that provides values
  - ensure that all assignments for dynamic values are assigned via the API that handles configuration, not with hard-coded values
  - scan the definitions of infrastructure to look for dynamic values and ensure they are assigned properly before building the environment
  - scan the configuration file and ensure that all defined dynamic values are present and have values

Fitness Function:

Ally All the Things

PenultimateWidgets has decided that all new development of user interfaces must support accessibility, both web and desktop.

Fitness Function:

Ally All the Things

- add a fitness function to the deployment pipeline for web projects that runs pa11y (http://pa11y.org/) to ensure compliance
- for Java, use PMD to check code within the user interface and look for control instantiation
- check to make sure every control has the AccessibleName property set

Fitness Function:

No More View Models

PenultimateWidgets has a group of developers who cut their teeth on mainframe programming, where all work is done via a terminal. They have (mostly) switched to modern web development, but architects find that they often create models and workflows directly in web pages rather than support a cleaner MVC separation, and they would like to discourage this behavior.

Fitness Function:

No More View Models

- use PMD (https://pmd.github.io/) or Language server protocol (https://langserver.org/) to build a tool that scans which classes developers instantiate in view code
- build a list of restricted model and controller packages
- if any view code instantiates a class from the restricted list, fail the build

“Parser” Fitness Function

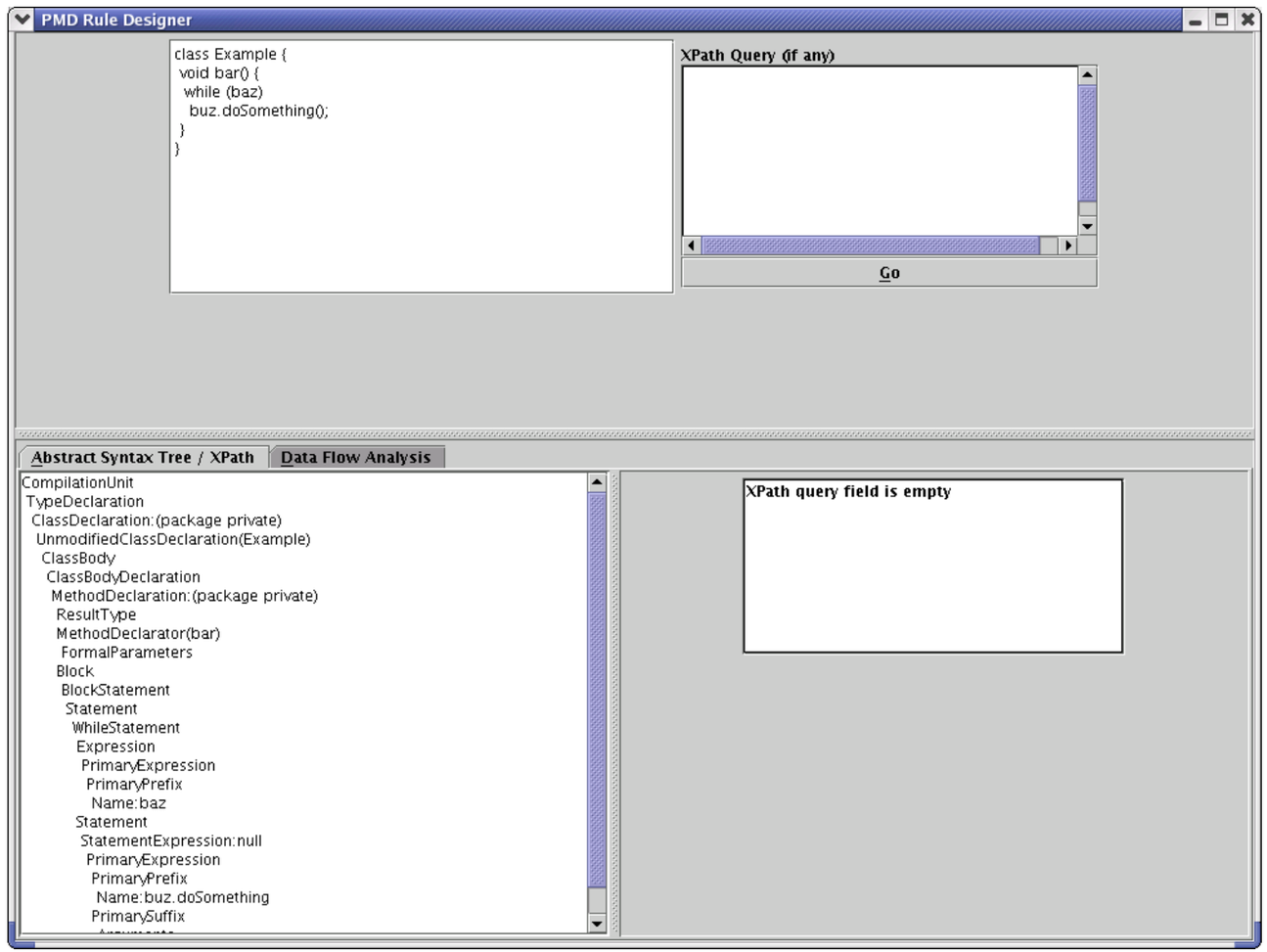
— 1. Full-blown testing library

— 2. Parser

— 3. Lexer

“Parser” Fitness Function

https://pmd.github.io/



“Parser” Fitness Function

https://pmd.github.io/

```
import net.sourceforge.pmd.lang.ast.*;
import net.sourceforge.pmd.lang.java.ast.*;
import net.sourceforge.pmd.lang.java.rule.*;

public class WhileLoopsMustUseBracesRule extends AbstractJavaRule {
    public Object visit(ASTWhileStatement node, Object data) {
        Node firstStmt = node.jjtGetChild(1);
        if (!hasBlockAsFirstChild(firstStmt)) {
            addViolation(data, node);
        }
        return super.visit(node,data);
    }
    private boolean hasBlockAsFirstChild(Node node) {
        return (node.jjtGetNumChildren() != 0 && (node.jjtGetChild(0) instanceof ASTBlock));
    }
}
```

“Parser” Fitness Function

— 1. Full-blown testing library

https://pmd.github.io/

— 2. Parser

— 3. Lexer

“Parser” Fitness Function

— 1. Full-blown testing library

— 2. Parser

www.antlr.org/

— 3. Lexer

https://pmd.github.io/

“Parser” Fitness Function

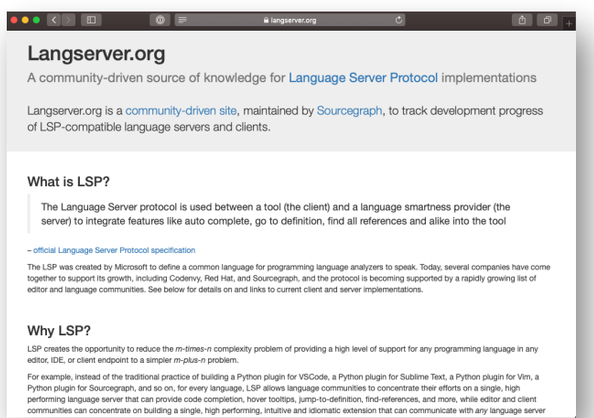
— 1. Full-blown testing library

— 2. Parser

www.antlr.org/

https://pmd.github.io/

— 3. Lexer



Fitness Function:

Break on Upgrade

PenultimateWidgets builds a number of applications in Ruby on Rails, which has an active release schedule. Occasionally, developers really want a feature that appears in version X+1, where PenultimateWidgets is currently on version X. To acquire the desired functionality, developers back-port the new features from X + 1 into X. However, when PenultimateWidgets finally does upgrade to X + 1, the back ported features are almost certain to break because of incompatibilities.

Fitness Function:

Break on Upgrade

- Add a unit test to the code base that checks the version number of the framework. If the framework version it higher than the expected one, fail the build with a warning that back-ported features will likely not work.
- Catalog all the back-ported features in the unit test to make it easy to identify them when it becomes time to replace them.

Fitness Function:

Upgrade All The Things

PenultimateWidgets has a bad habit of waiting too long to update core libraries and frameworks they depend upon for development. Waiting past several versions makes the eventual upgrade quite painful, and they have resolved to perform upgrades in a more timely manner.

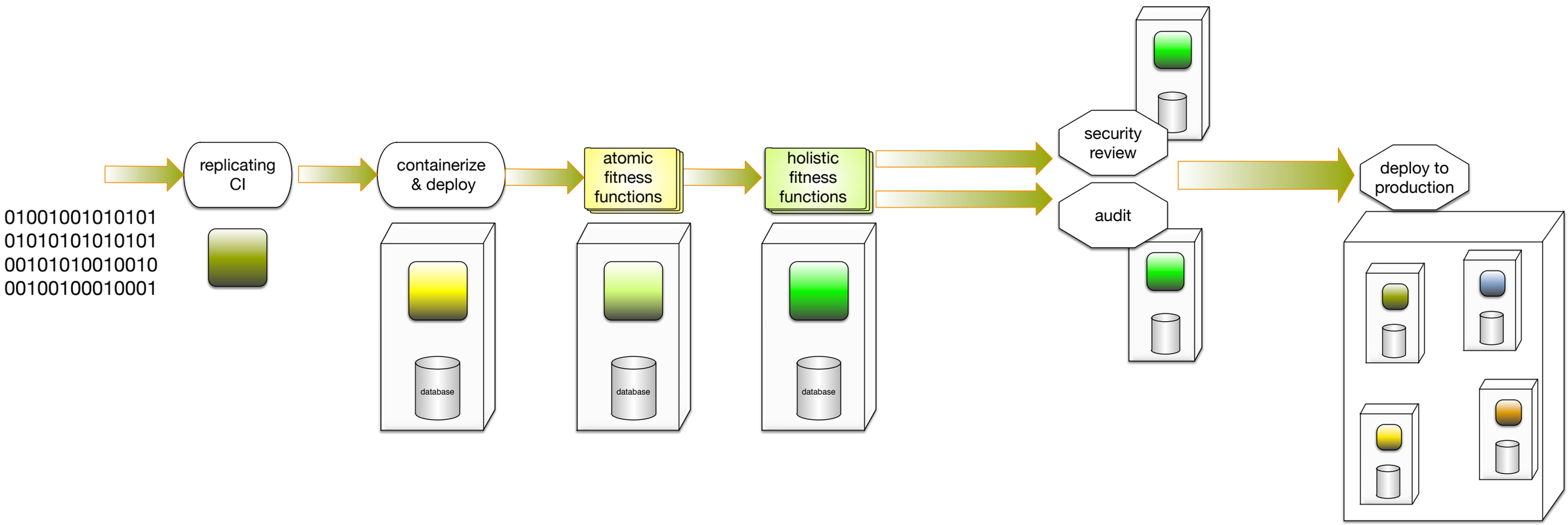
Fitness Function:

Upgrade All The Things

- Use Dependabot to flag new major versions of frameworks and libraries.
- When a library updates, make an entry in a build database and start a 3-month clock.
- If the library or framework in question hasn't been updated in 3 months, fail the build.

Penultimate

Deployment Pipeline



Fitness Function:

Zero Day Security Check

PenultimateWidgets management has become concerned about zero-day exploits in open source libraries and has tasked the security team with coming up with a way of ensuring that projects do not use known compromised versions.

Fitness Function:

Zero Day Security Check

- build a common fitness function into every deployment pipeline for the security team
- have the security fitness function check version numbers of deployed libraries
- if a tainted version is in use, fail the build and notify the team

Penultimate

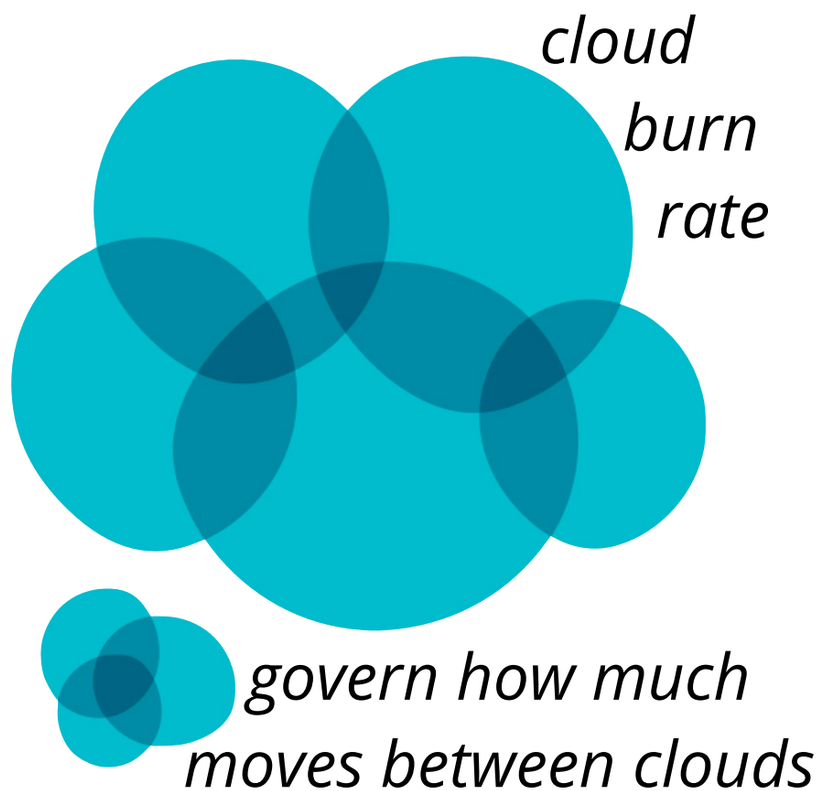
Fitness Function:

Cloudy Fitness Functions

Penultimate

Fitness Function:

Cloudy Fitness Functions



Penultimate

Fitness Function:

Deployment Pipelines

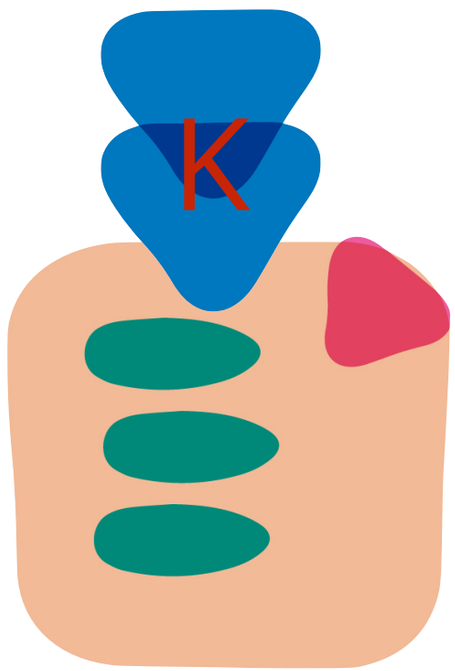
validate auto-generated deployment pipelines

Penultimate

Fitness Function:

K-Weight

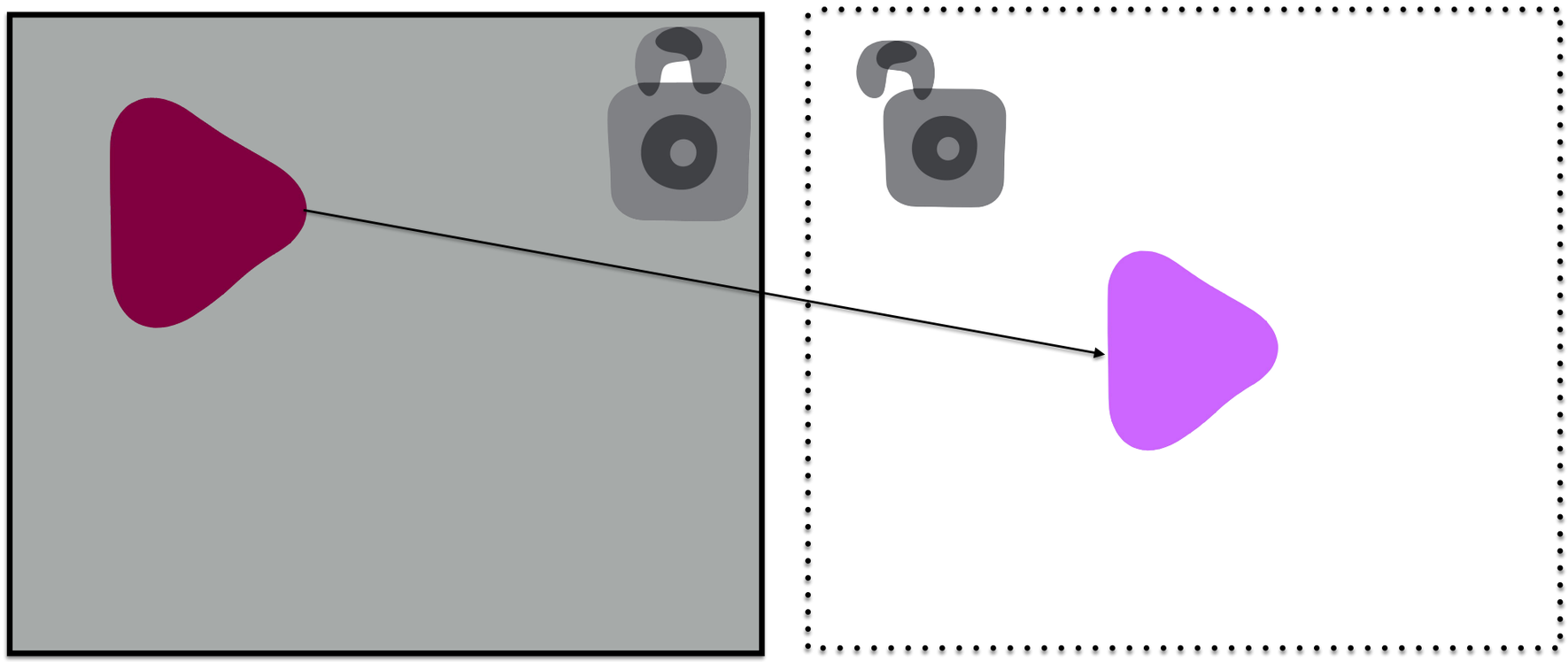
K-weight for javascript downloads



Penultimate

Fitness Function:

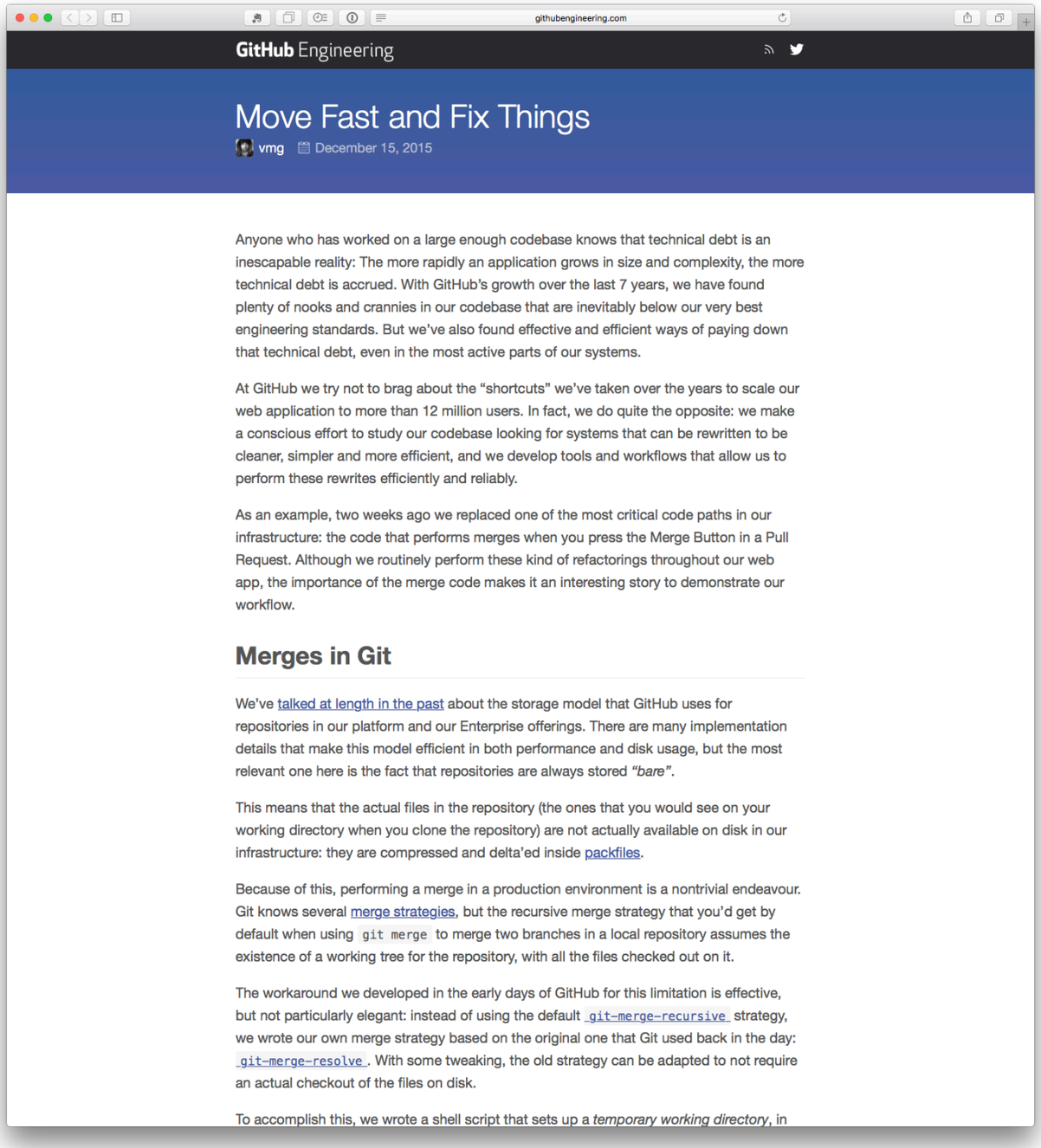
PCI+1 Calls



Fitness Function:

Replace the Crufty Core

PenultimateWidgets built a network scanning tool many, many years ago to look for suspicious activity at the network packet level. To achieve the proper performance, the code was written and has been maintained in increasingly complex C code. PenultimateWidgets needs to replace the code but is nervous about introducing bugs and/or harming performance or throughput.



Move Fast & Fix Things


GitHub Engineering

Move Fast and Fix Things

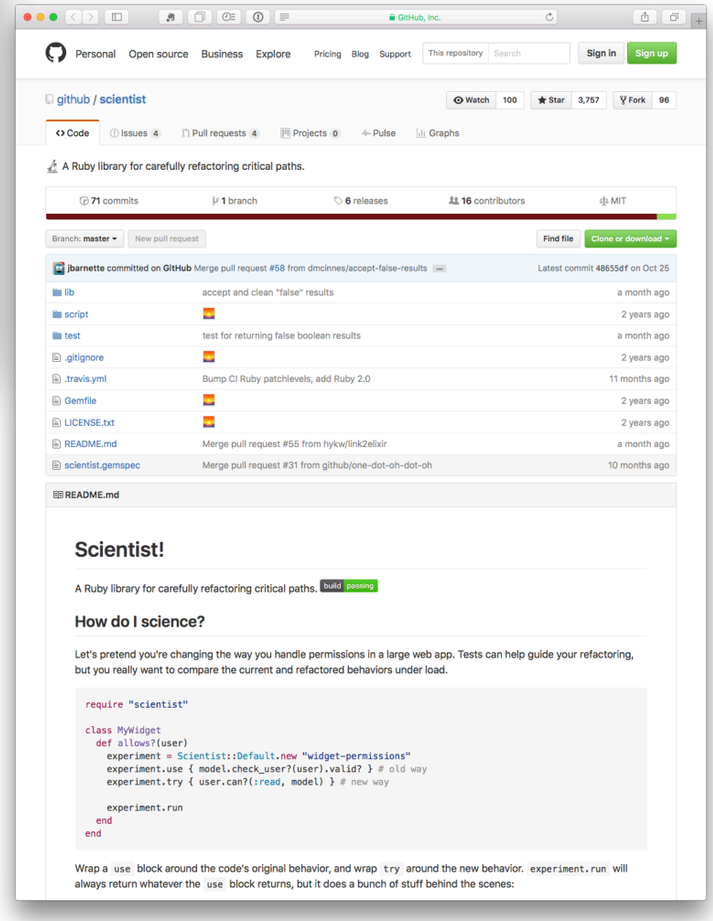
Merges in Git

```ruby
def create_merge_commit(base, head, author, commit_message)
  base = resolve_commit(base)
  head = resolve_commit(head)
  commit_message = Rugged.prettify_message(commit_message)

  merge_base = rugged.merge_base(base, head)
  return [nil, "already_merged"] if merge_base == head.oid

  ancestor_tree = merge_base && Rugged::Commit.lookup(rugged, merge_base).tree
  merge_options = {
    :fail_on_conflict => true,
    :skip_reuc => true,
    :no_recursive => true,
  }
  index = base.tree.merge(head.tree, ancestor_tree, merge_options)
  return [nil, "merge_conflict"] if (index.nil? || index.conflicts?)

  options = {
    :message    => commit_message,
    :committer  => author,
    :author     => author,
    :parents    => [base, head],
    :tree       => index.write_tree(rugged)
  }

  [Rugged::Commit.create(rugged, options), nil]
end
```

```ruby
def create_merge_commit(author, base, head, options = {})
  commit_message = options[:commit_message] || "Merge #{head} into #{base}"
  now = Time.current

  science "create_merge_commit" do |e|
    e.context :base => base.to_s, :head => head.to_s, :repo => repository.nwo
    e.use { create_merge_commit_git(author, now, base, head, commit_message) }
    e.try { create_merge_commit_rugged(author, now, base, head, commit_message)
  end
end
```


```ruby
def create_merge_commit(base, head, author, commit_message)
  base = resolve_commit(base)
  head = resolve_commit(head)
  commit_message = Rugged.prettify_message(commit_message)

  merge_base = rugged.merge_base(base, head)
  return [nil, "already_merged"] if merge_base == head.oid

  ancestor_tree = merge_base && Rugged::Commit.lookup(rugged, merge_base).tree
  merge_options = {
    :fail_on_conflict => true,
    :skip_reuc => true,
    :no_recursive => true,
  }
  index = base.tree.merge(head.tree, ancestor_tree, merge_options)
  return [nil, "merge_conflict"] if (index.nil? || index.conflicts?)

  options = {
    :message    => commit_message,
    :committer  => author,
    :author     => author,
    :parents    => [base, head],
    :tree       => index.write_tree(rugged)
  }

  [Rugged::Commit.create(rugged, options), nil]
end
```

```ruby
def create_merge_commit(author, base, head, options = {})
  commit_message = options[:commit_message] || "Merge #{head} into #{base}"
  now = Time.current

  science "create_merge_commit" do |e|
    e.context :base => base.to_s, :head => head.to_s, :repo => repository.nwo
    e.use { create_merge_commit_git(author, now, base, head, commit_message) }
    e.try { create_merge_commit_rugged(author, now, base, head, commit_message)
  end
end
```



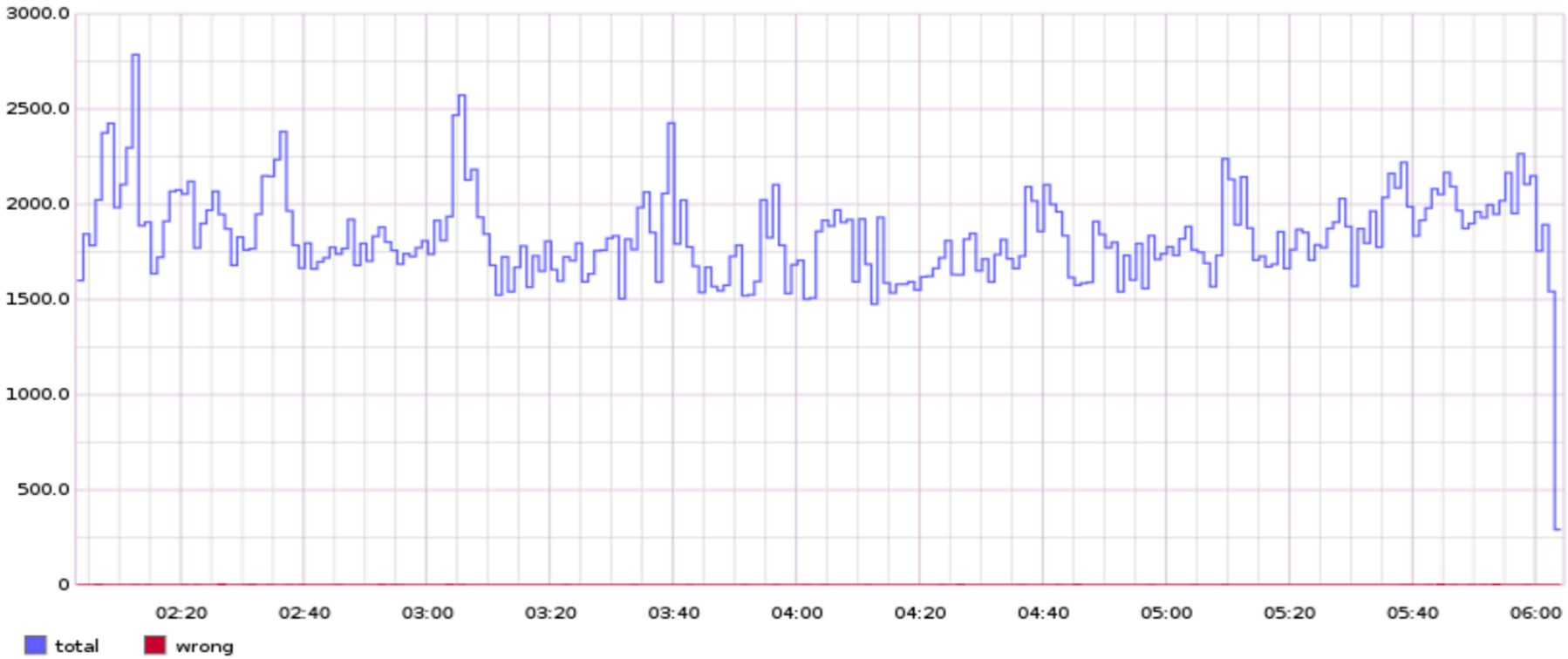
https://github.com/github/scientist


```ruby
require "scientist"

class MyWidget
  def allows?(user)
    experiment = Scientist::Default.new "widget-permissions"
    experiment.use { model.check_user?(user).valid? } # old way
    experiment.try { user.can?(:read, model) } # new way

    experiment.run
  end
end
```

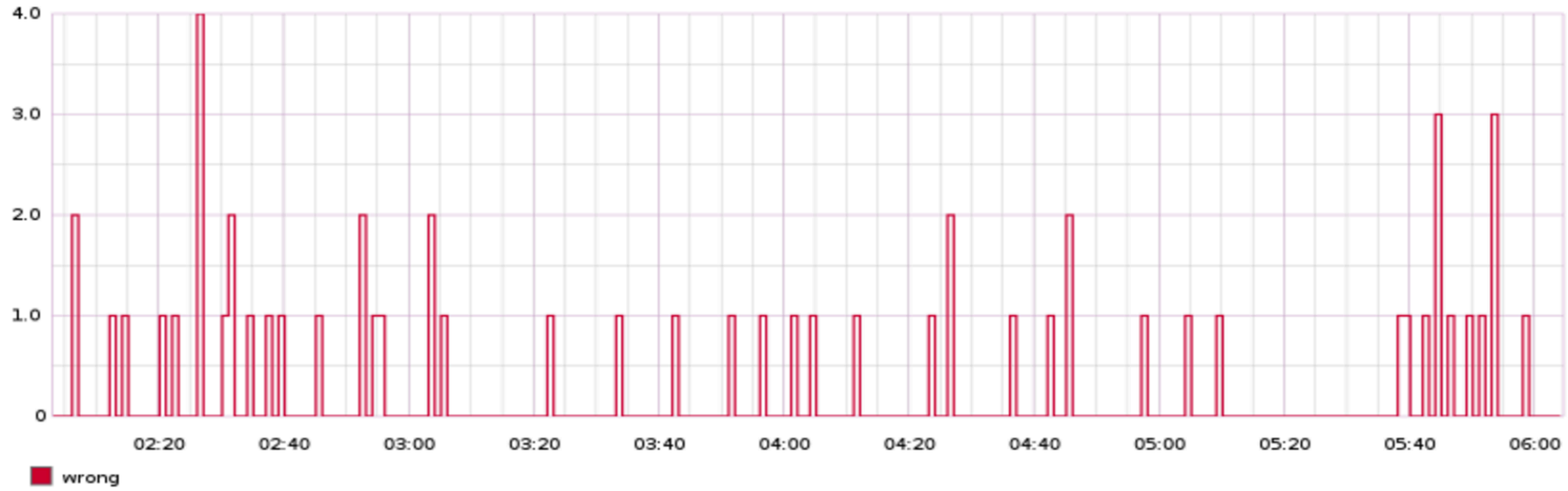
- It decides whether or not to run the try block,
- Randomizes the order in which use and try blocks are run,
- Measures the durations of all behaviors,
- Compares the result of try to the result of use,
- Swallows (but records) any exceptions raised in the try block
- Publishes all this information.

Accuracy

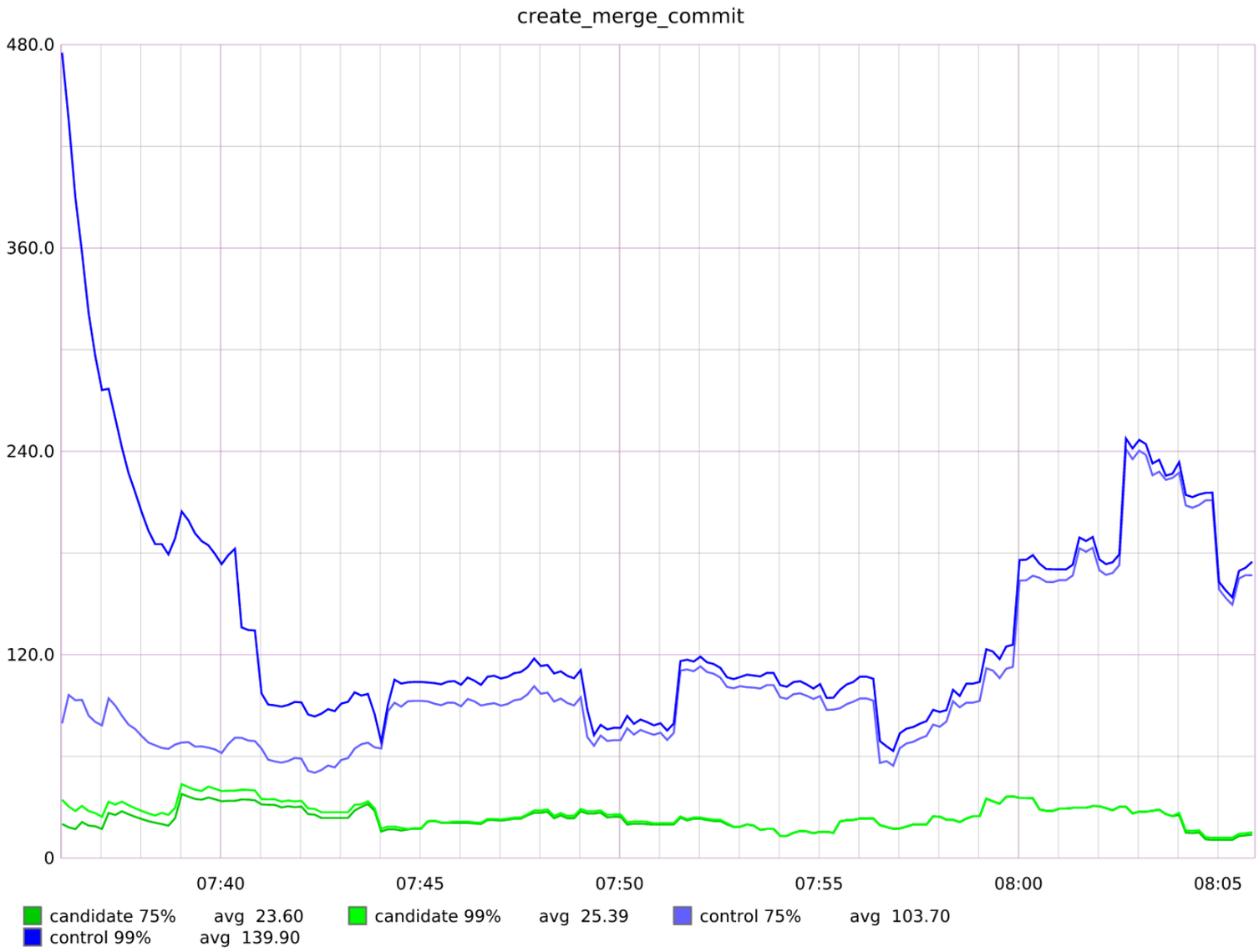
The number of times that the candidate and the control agree or disagree. View mismatches



The number of incorrect/ignored only.



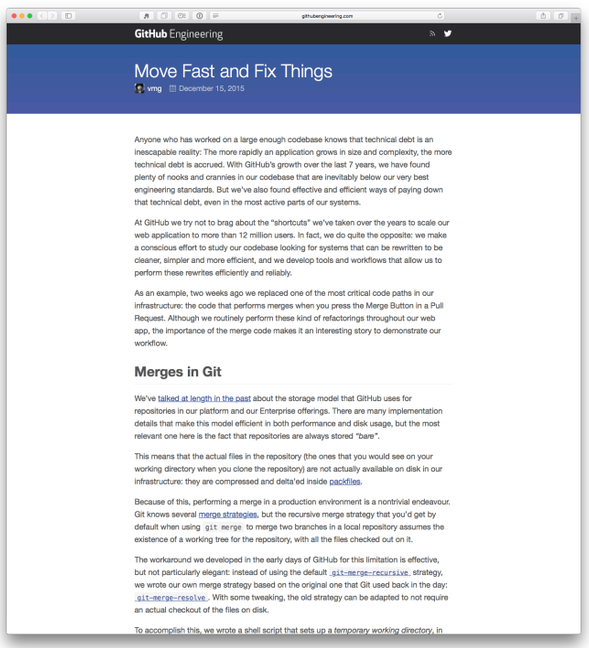

GitHub Engineering
Move Fast and Fix Things
Merges in Git


create_merge_commit
480.0
360.0
240.0
120.0
0
07:40
07:45
07:50
07:55
08:00
08:05
candidate 75% avg 23.60
candidate 99% avg 25.39
control 75% avg 103.70
control 99% avg 139.90

4 days

24 hours/
no mismatches or slow cases

> 10,000,000 comparisons


githubengineering.com

GitHub Engineering

Move Fast and Fix Things

vmg December 15, 2015

Anyone who has worked on a large enough codebase knows that technical debt is an inescapable reality: The more rapidly an application grows in size and complexity, the more technical debt is accrued. With GitHub's growth over the last 7 years, we have found plenty of nooks and crannies in our codebase that are inevitably below our very best engineering standards. But we've also found effective and efficient ways of paying down that technical debt, even in the most active parts of our systems.

At GitHub we try not to brag about the "shortcuts" we've taken over the years to scale our web application to more than 12 million users. In fact, we do quite the opposite: we make a conscious effort to study our codebase looking for systems that can be rewritten to be cleaner, simpler and more efficient, and we develop tools and workflows that allow us to perform these rewrites efficiently and reliably.

As an example, two weeks ago we replaced one of the most critical code paths in our infrastructure: the code that performs merges when you press the Merge Button in a Pull Request. Although we routinely perform these kind of refactorings throughout our web app, the importance of the merge code makes it an interesting story to demonstrate our workflow.

Merges in Git

We've talked at length in the past about the storage model that GitHub uses for repositories in our platform and our Enterprise offerings. There are many implementation details that make this model efficient in both performance and disk usage, but the most relevant one here is the fact that repositories are always stored *"bare"*.

This means that the actual files in the repository (the ones that you would see on your working directory when you clone the repository) are not actually available on disk in our infrastructure: they are compressed and delta'ed inside packfiles.

Because of this, performing a merge in a production environment is a nontrivial endeavour. Git knows several merge strategies, but the recursive merge strategy that you'd get by default when using `git merge` to merge two branches in a local repository assumes the existence of a working tree for the repository, with all the files checked out on it.

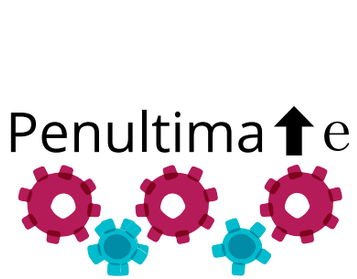
The workaround we developed in the early days of GitHub for this limitation is effective, but not particularly elegant: instead of using the default `git-merge-recursive` strategy, we wrote our own merge strategy based on the original one that Git used back in the day: `git-merge-resolve`. With some tweaking, the old strategy can be adapted to not require an actual checkout of the files on disk.

To accomplish this, we wrote a shell script that sets up a *temporary working directory*, in

Fitness Function:

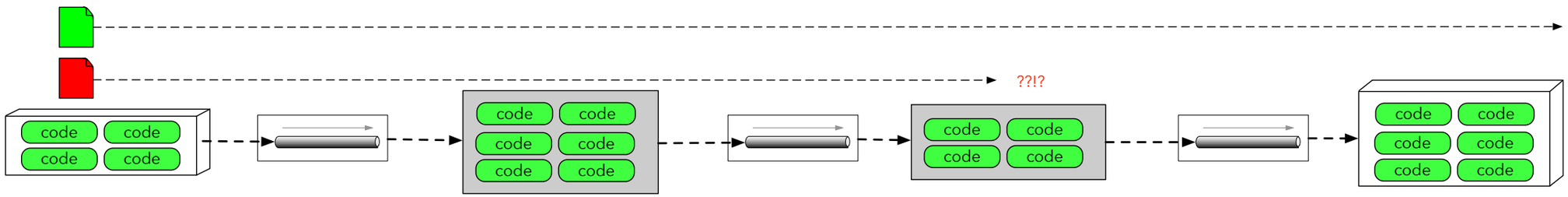
Replace the Crufty Core

- Built fitness functions around the existing C code to ensure performance and throughput values.
- Build a spike solution in both Java and Go (and perhaps others) and run them against the fitness functions to see if they can meet the threshold
- Use Scientist (https://github.com/github/scientist) port Scientist4J (https://github.com/rawls238/Scientist4J) to verify the new code against the old code in production



fitness function:

Missing Messages

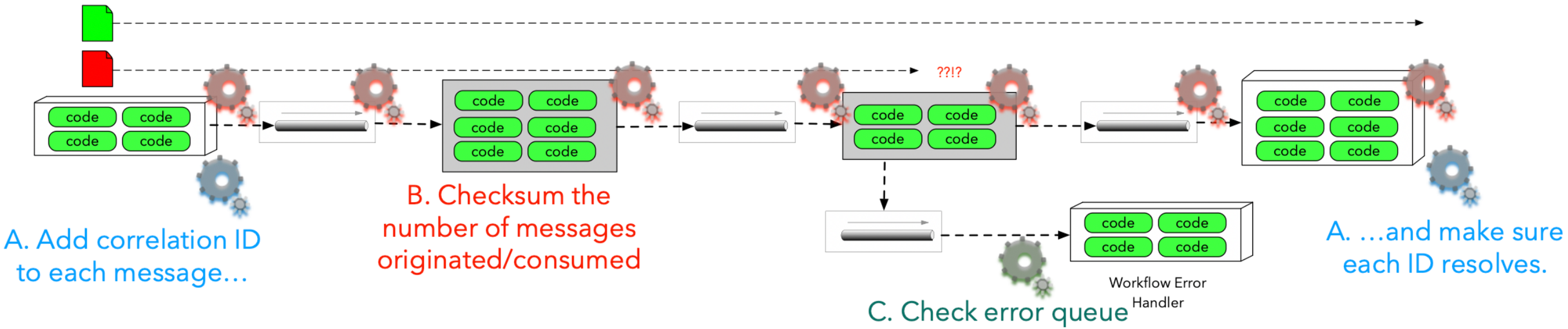


Penultimate Widgets has had a problem with lost / mis-routed tickets, and the architects have narrowed it down to lost messages in a message queue because of a hard to reproduce error.

Penultimate

fitness function:

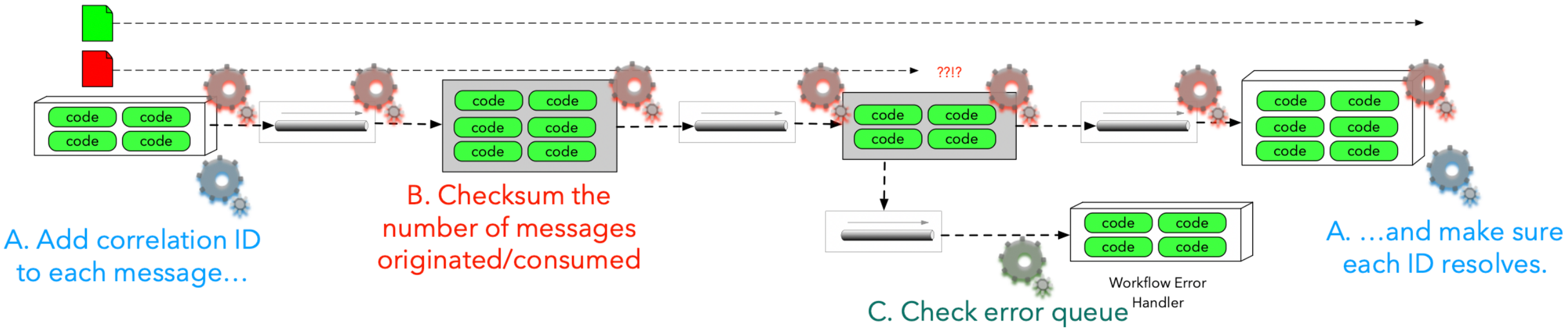
Missing Messages



Penultimate

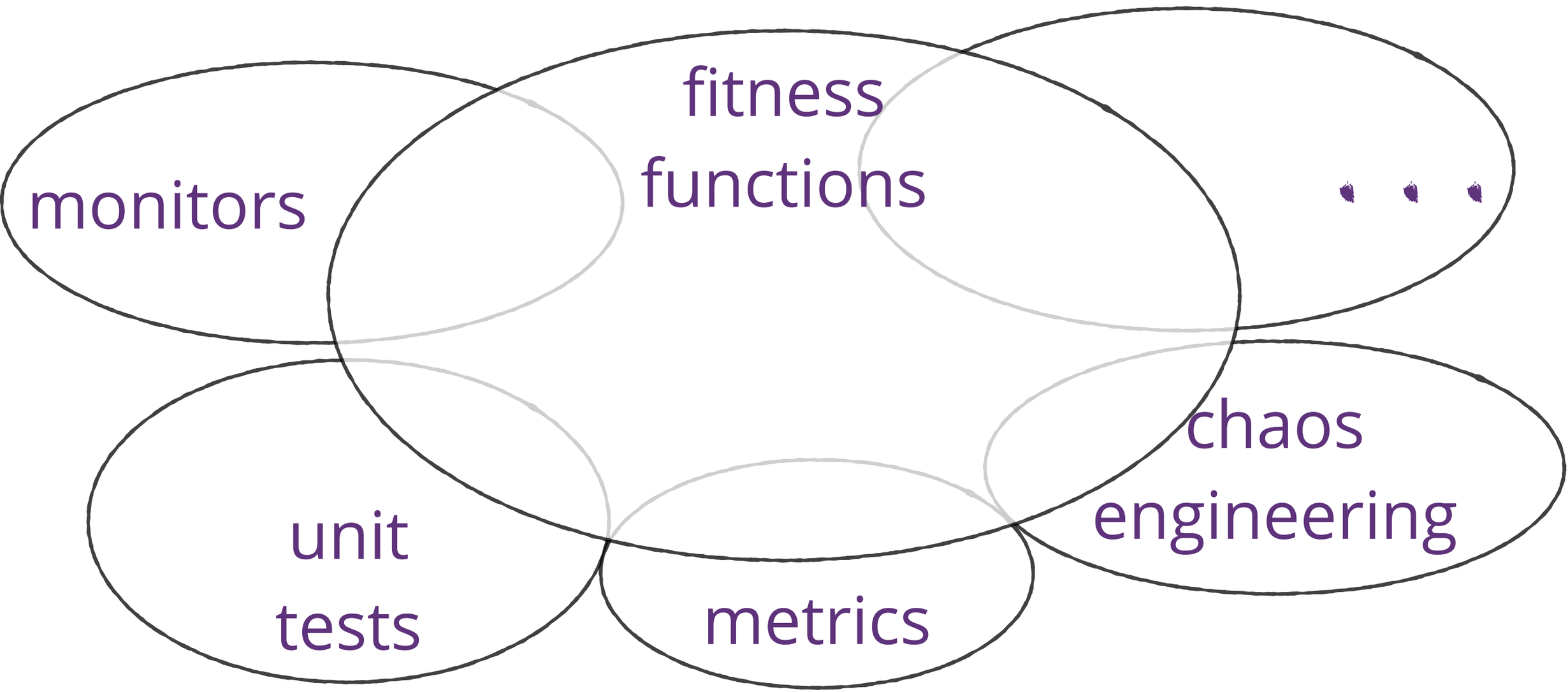
fitness function:

Missing Messages



Any of the above!

Fitness Functions



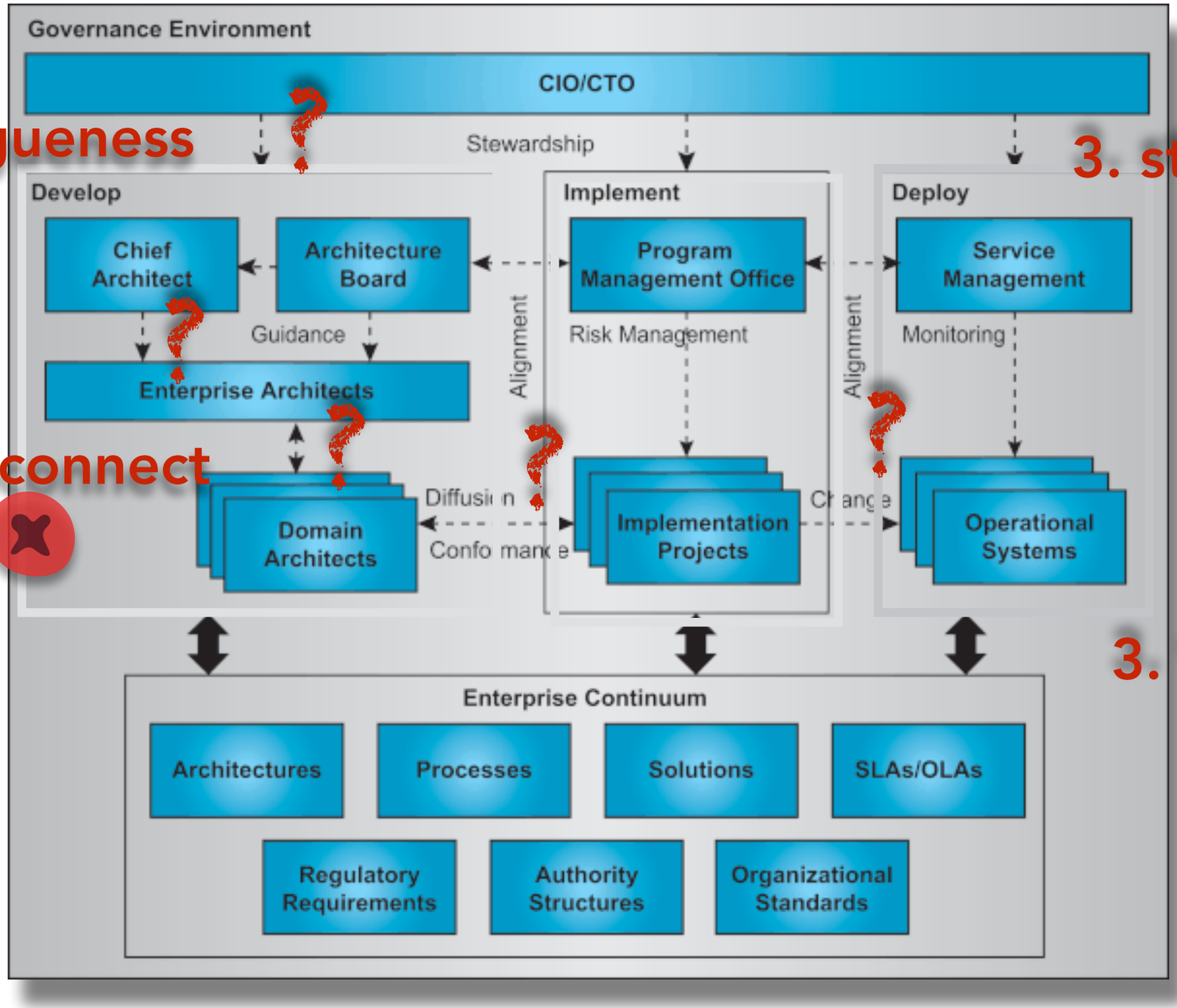
Defining Governance

2. vagueness

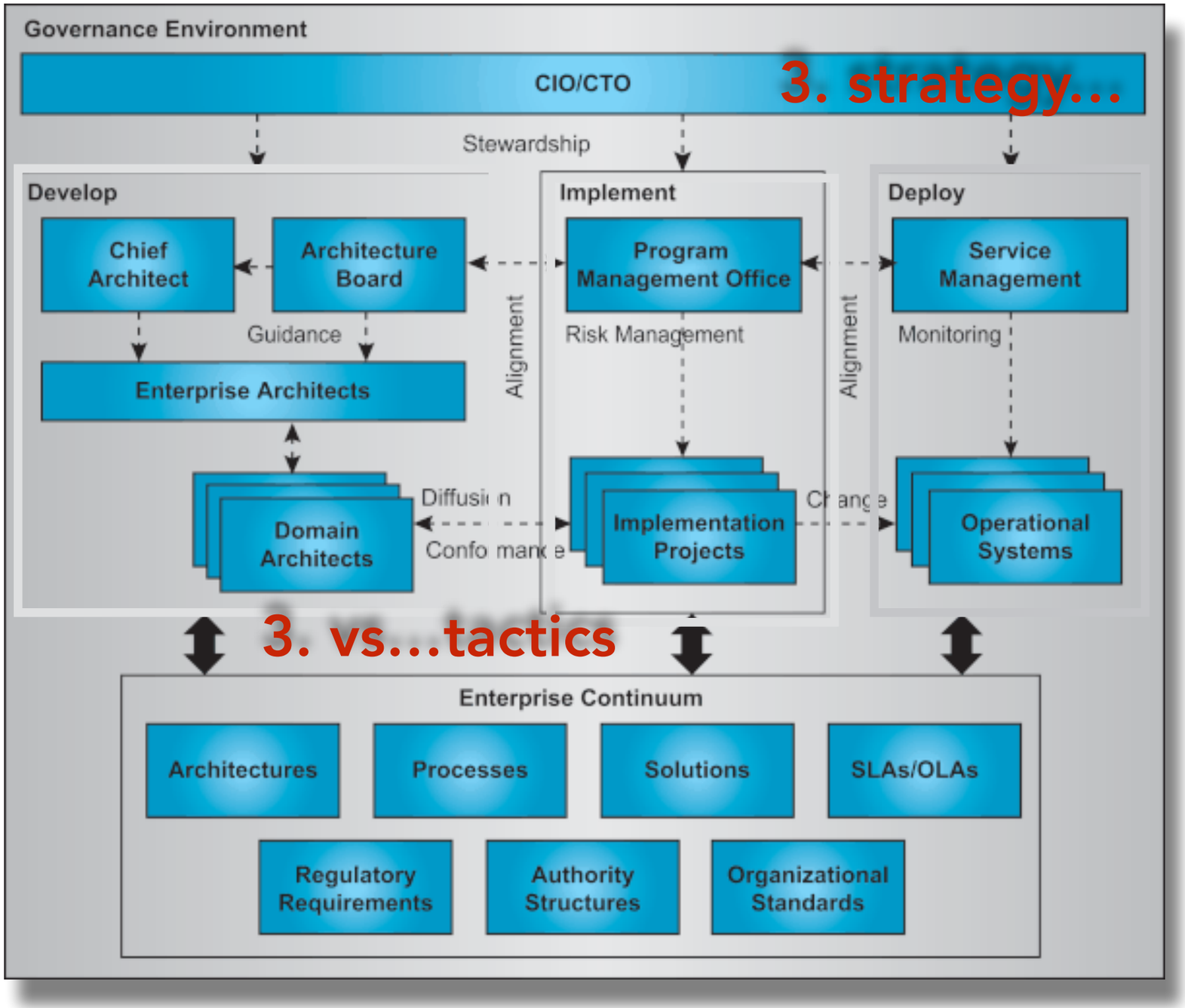
3. strategy...

1. disconnect

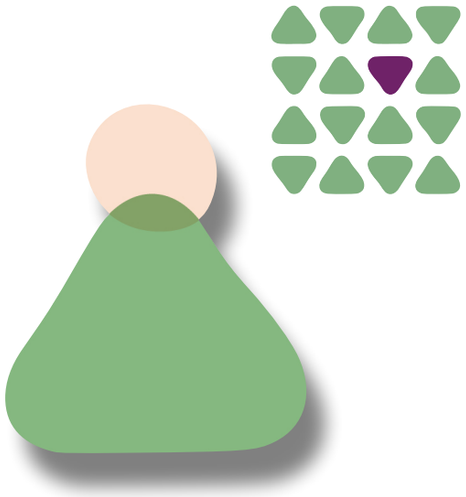
3. vs...tactics



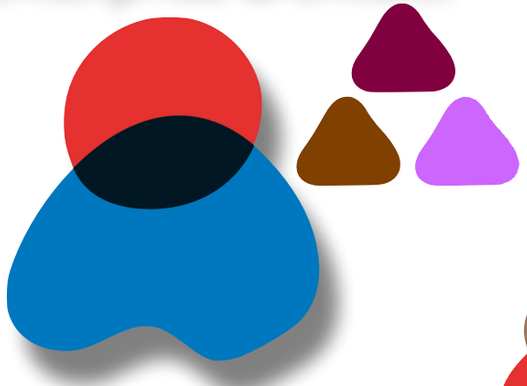
Defining Governance



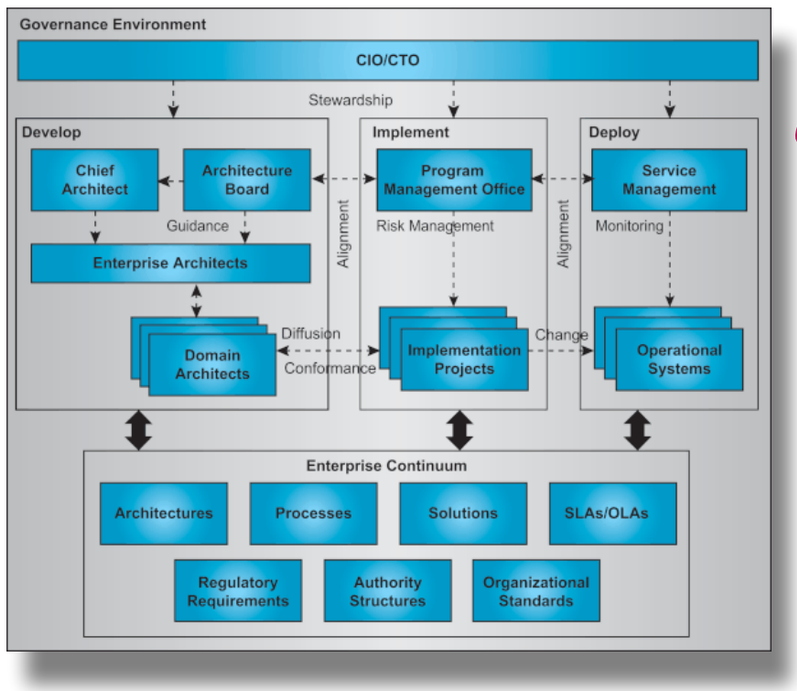
Defining Governance

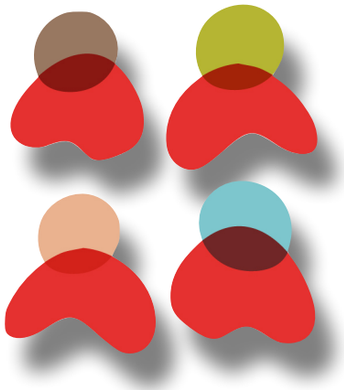


enterprise architect

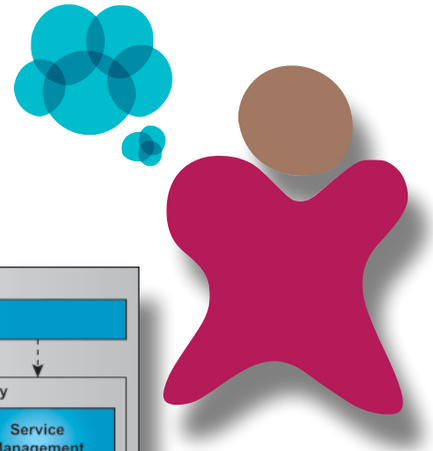


domain architect





developers



CTO

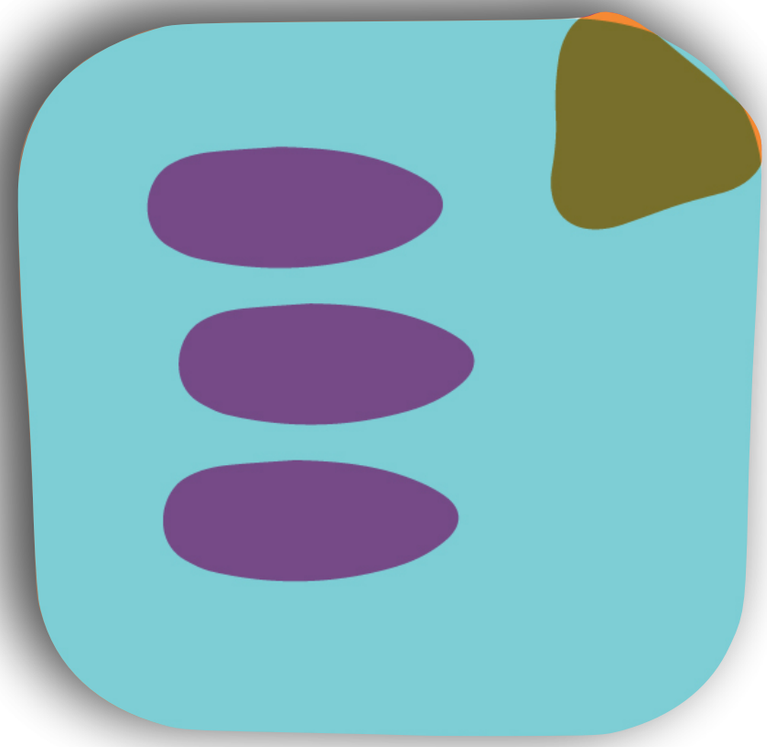
3. strategy...

objective outcomes

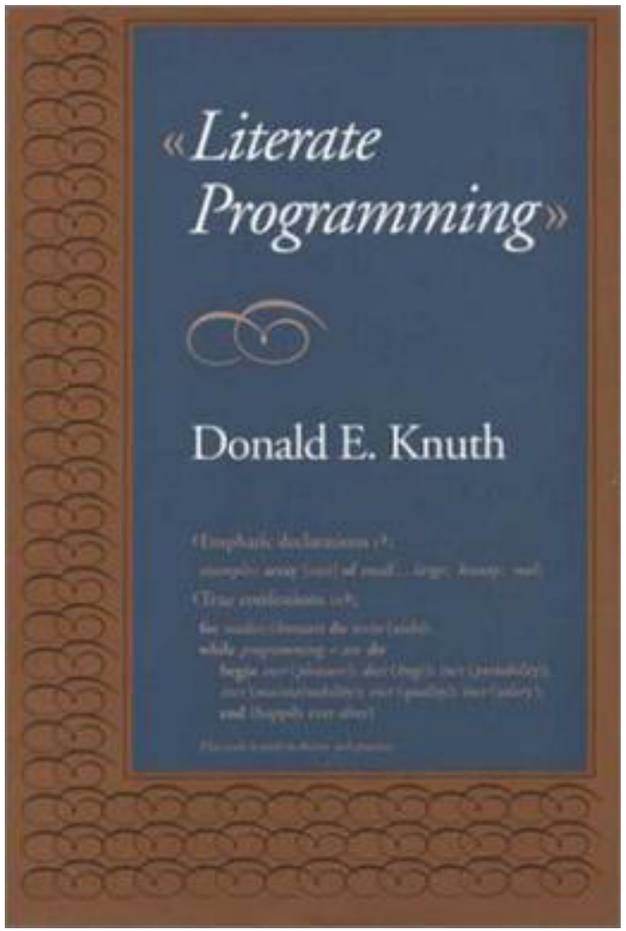
3. vs...tactics

Documenting Governance

— fitness functions
— ATAM Utility trees
— Jupyter notebooks
— org-babel



Literate Programming



cweb

```
Here, then, is an overview of the file wc.c that is defined by the noweb program wc.nw:
    <<*>>=
    <<Header files to include>>
    <<Definitions>>
    <<Global variables>>
    <<Functions>>
    <<The main program>>
    @

We must include the standard I/O definitions, since we want to send formatted output
to stdout and stderr.
    <<Header files to include>>=
    #include <stdio.h>
    @
```

Here, then, is an overview of the file wc.c that is defined by the noweb program wc.nw:

```
<<*>>=
<<Header files to include>>
<<Definitions>>
<<Global variables>>
<<Functions>>
<<The main program>>
@
```

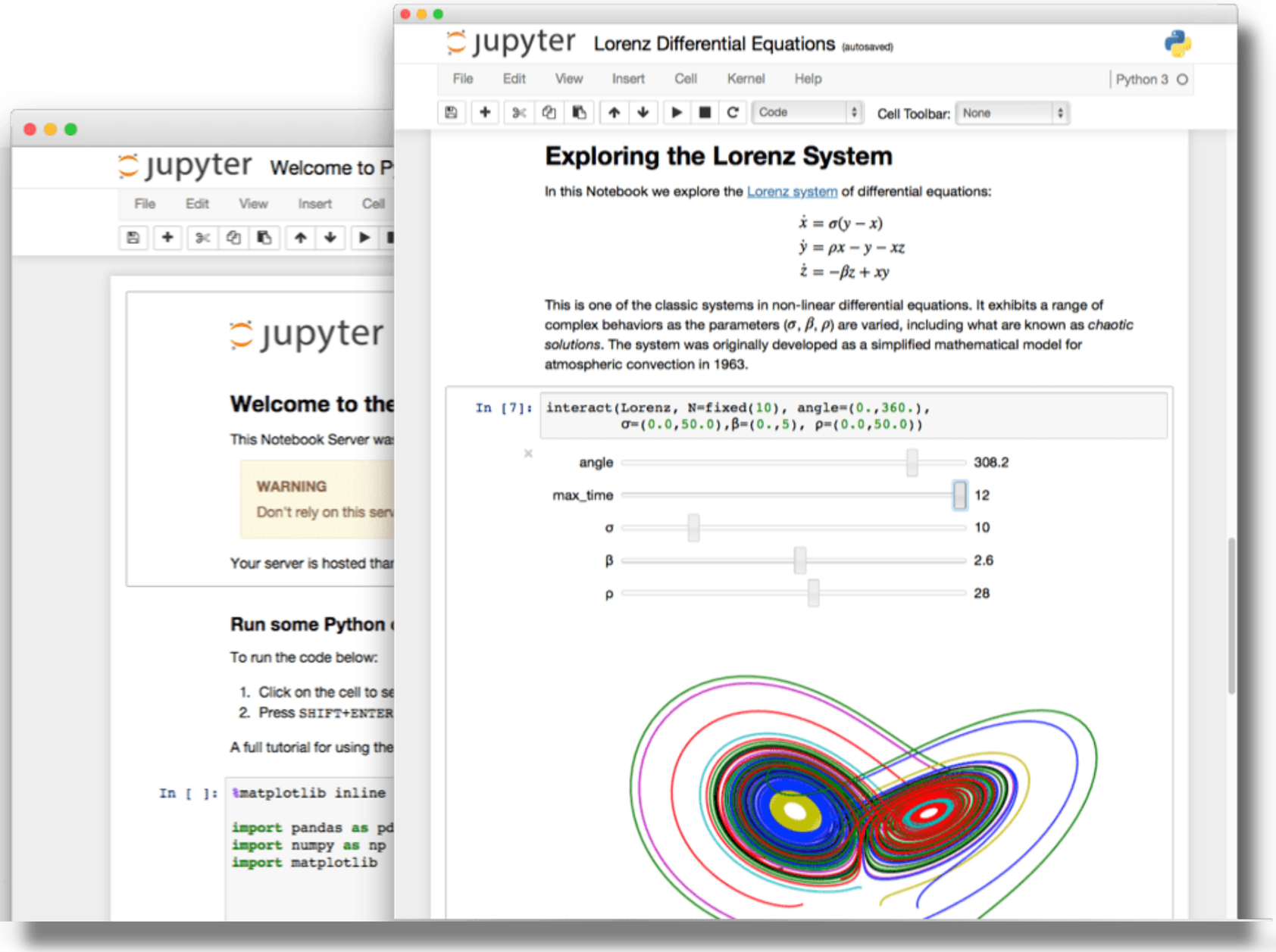
We must include the standard I/O routines for access to stdout and stderr.

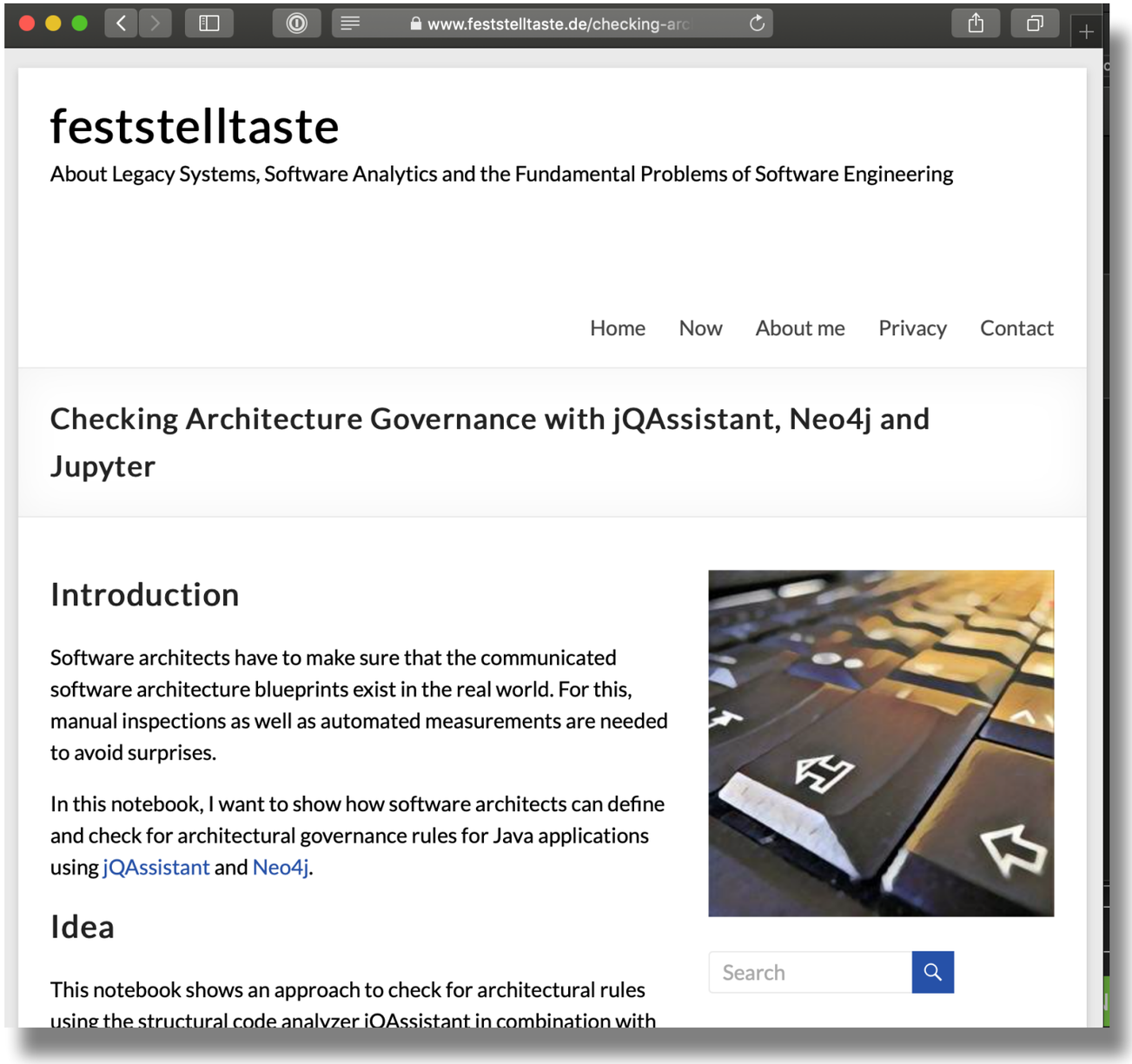
```
<<Header files to include>>
#include <stdio.h>
@
```

The present chunk, which does the counting, was actually one of the simplest to write. We look at each character and change state if it begins or ends a word.

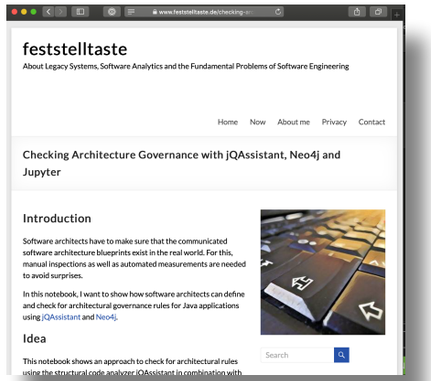
```
<<Scan file>>=
while (1) {
  <<Fill buffer if it is empty; break at end of file>>
  c = *ptr++;
  if (c > ' ' && c < 0177) {
    /* visible ASCII codes */
    if (!in_word) {
      word_count++;
      in_word = 1;
    }
    continue;
  }
  if (c == '\n') line_count++;
  else if (c != ' ' && c != '\t') continue;
  in_word = 0;
    /* c is newline, space, or tab */
}
@
```

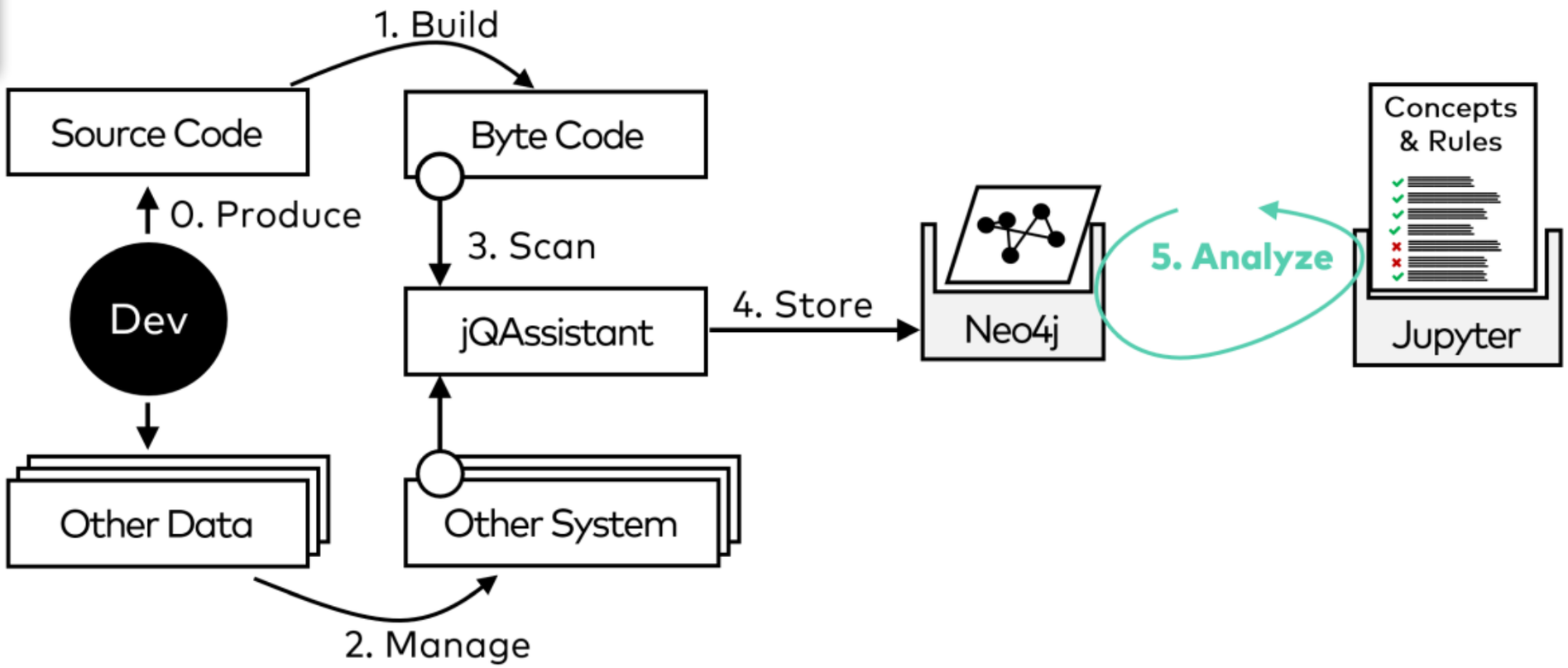
Jupyter Notebooks



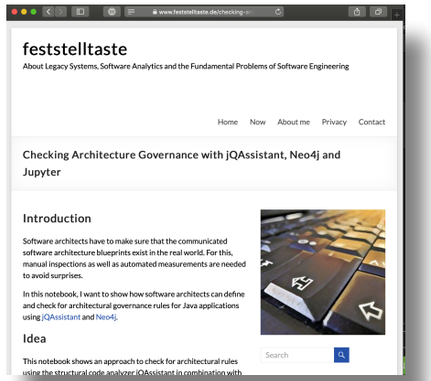


https://www.feststelltaste.de/checking-architecture-governance-with-jqassistant-neo4j-and-jupyter/





https://www.feststelltaste.de/checking-architecture-governance-with-jqassistant-neo4j-and-jupyter/



Rule Definition

Because we want to enable developers to quickly identify Entities in the application, we use the Java package names to define were all the Entities are placed. Thus, Entities must be placed in a package named `model`.

The following query lists all existing Entities in the scanned codebase:

In [3]:
```
%%cypher
MATCH (e:Entity)<-[:CONTAINS]-(p:Package)
WHERE p.name = "model"
RETURN e.fqn as Entity, p.name as Package
```

```
5 rows affected.
```

Out[3]:

Entity	Package
org.springframework.samples.petclinic.model.Owner	model
org.springframework.samples.petclinic.model.Pet	model
org.springframework.samples.petclinic.model.Visit	model
org.springframework.samples.petclinic.model.Specialty	model
org.springframework.samples.petclinic.model.Vet	model

Rule Violations

The following query lists all Enties that doesn't comply to the rule above:

In [4]:
```
%%cypher
MATCH (e:Entity)<-[:CONTAINS]-(p:Package)
WHERE p.name <> "model"
RETURN e.fqn as MisplacedEntity, p.name as WrongPackage
```

```
1 rows affected.
```

Out[4]:

MisplacedEntity	WrongPackage

org-babel

```
#+TITLE: fitness function demo
#+AUTHOR: Neal Ford
#+STARTUP: showall indent
#+OPTIONS: author:t num:nil toc:nil

#+NAME: build-path
#+BEGIN_SRC emacs-lisp
"~/work/ArchUnit/"
#+END_SRC


* structural fitness function
  * component cycles
      :PROPERTIES:
      :cycles-permitted: 3
      :END:
    #+NAME: check-cycles
    #+BEGIN_SRC emacs-lisp :var props=(org-entry-properties) :var path=build-path :exports both
    (defmacro get-prop (name) (cdr (assoc (upcase name) props)))
    (message (concat"checking for " (get-prop "cycles-permitted") " cycles"))
    (shell-command (concat "cd " path) ";./gradlew clean build -P cycle=" (get-prop "cycles-permitted"))
    #+END_SRC

    #+RESULTS: check-cycles
    : fail
```

Analyzing Tradeoffs

ATAM
Architecture tradeoff analysis method

– A risk mitigation process developed by the Software Engineering Institute at the Carnegie Mellon University.

– Its purpose is to help choose a suitable architecture for a software system by discovering trade-offs and sensitivity points.

– ATAM is most beneficial when done early in the software development life-cycle, when the cost of changing architectures is minimal.

Steps in the ATM Process

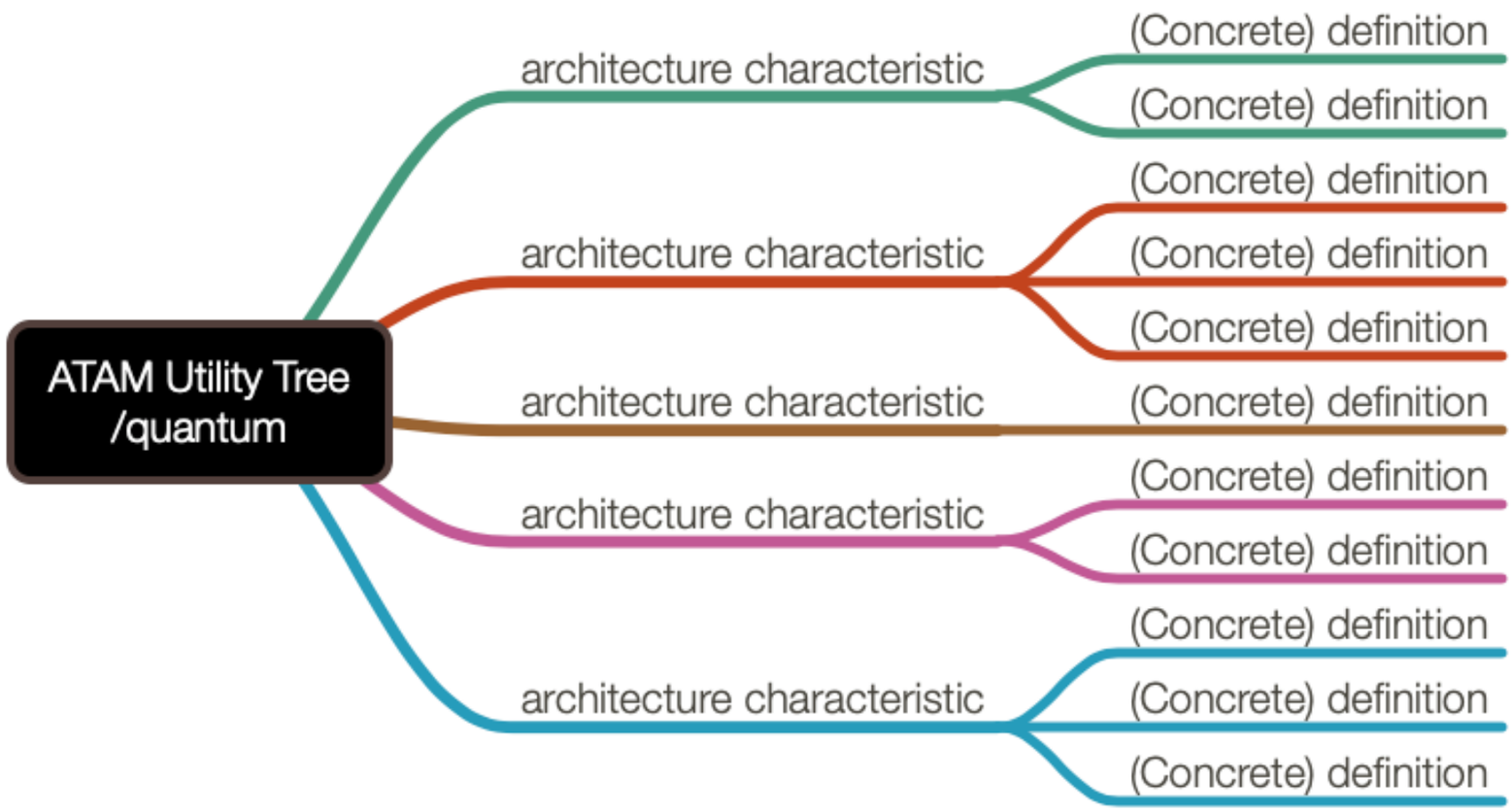
- Present ATAM – Present the concept of ATAM to the stakeholders, and answer any questions about the process.
- Present business drivers – everyone in the process presents and evaluates the business drivers for the system in question.
- Present the architecture – the architect presents the high level architecture to the team, with an 'appropriate level of detail'
- Identify architectural approaches – different architectural approaches to the system are presented by the team, and discussed.
- Generate quality attribute utility tree – define the core business and technical requirements of the system, and map them to an appropriate architectural property. Present a scenario for this given requirement.
- Analyze architectural approaches – Analyze each scenario, rating them by priority. The architecture is then evaluated against each scenario.
- Brainstorm and prioritize scenarios – among the larger stakeholder group, present the current scenarios, and expand.
- Analyze architectural approaches – Perform step 6 again with the added knowledge of the larger stakeholder community.
- Present results – provide all documentation to the stakeholders.

Steps in the ATM Process

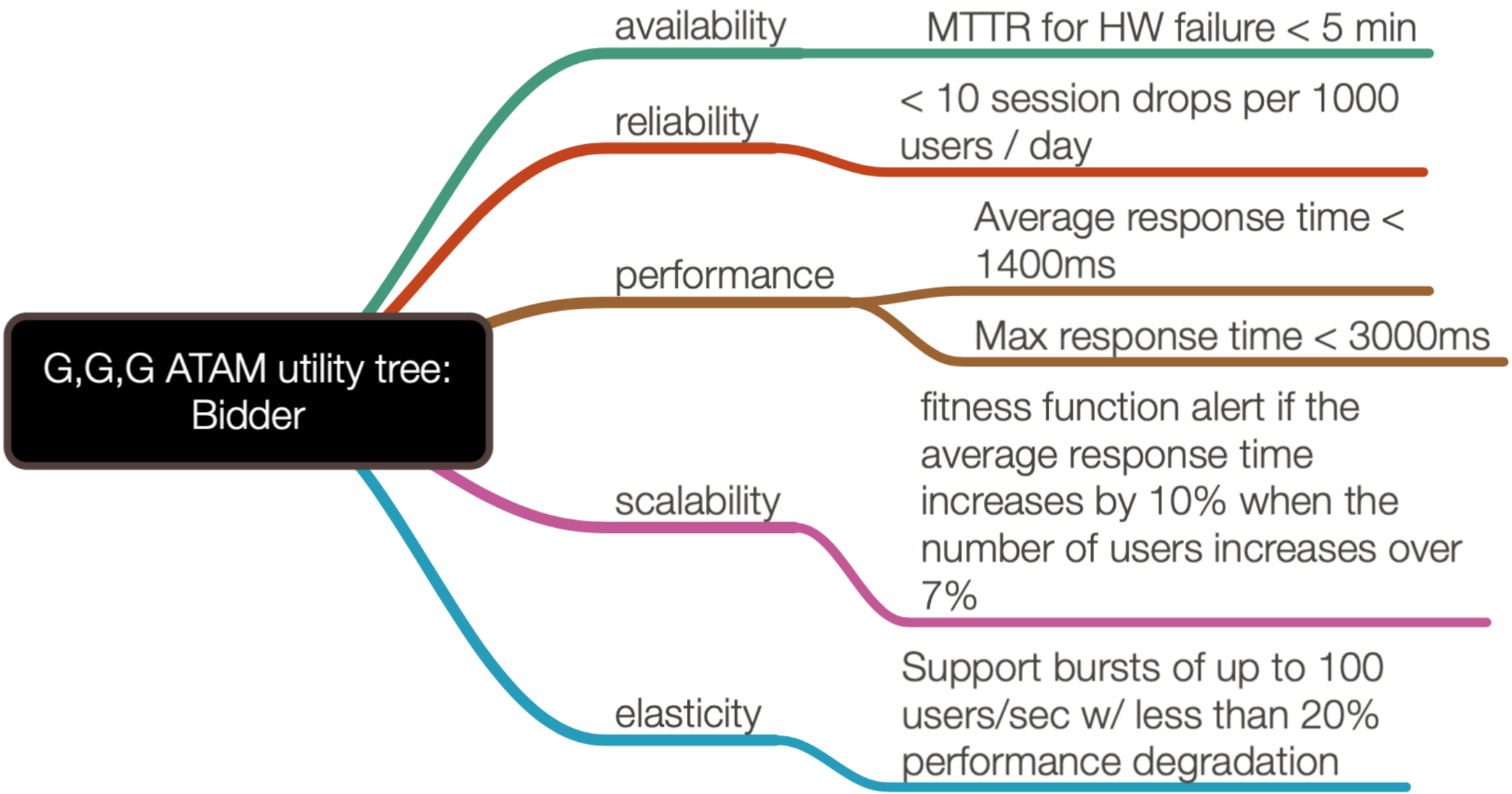
Useful

- ~~Present ATAM – Present the concept of ATAM to the stakeholders, and answer any questions about the process.~~
- ~~Present business drivers – everyone in the process presents and evaluates the business drivers for the system in question.~~
- ~~Present the architecture – the architect presents the high level architecture to the team, with an 'appropriate level of detail'~~
- ~~Identify architectural approaches – different architectural approaches to the system are presented by the team, and discussed.~~
- Generate quality attribute utility tree – define the core business and technical requirements of the system, and map them to an appropriate architectural property. Present a scenario for this given requirement.
- ~~Analyze architectural approaches – Analyze each scenario, rating them by priority. The architecture is then evaluated against each scenario.~~
- ~~Brainstorm and prioritize scenarios – among the larger stakeholder group, present the current scenarios, and expand.~~
- ~~Analyze architectural approaches – Perform step 6 again with the added knowledge of the larger stakeholder community.~~
- ~~Present results – provide all documentation to the stakeholders.~~

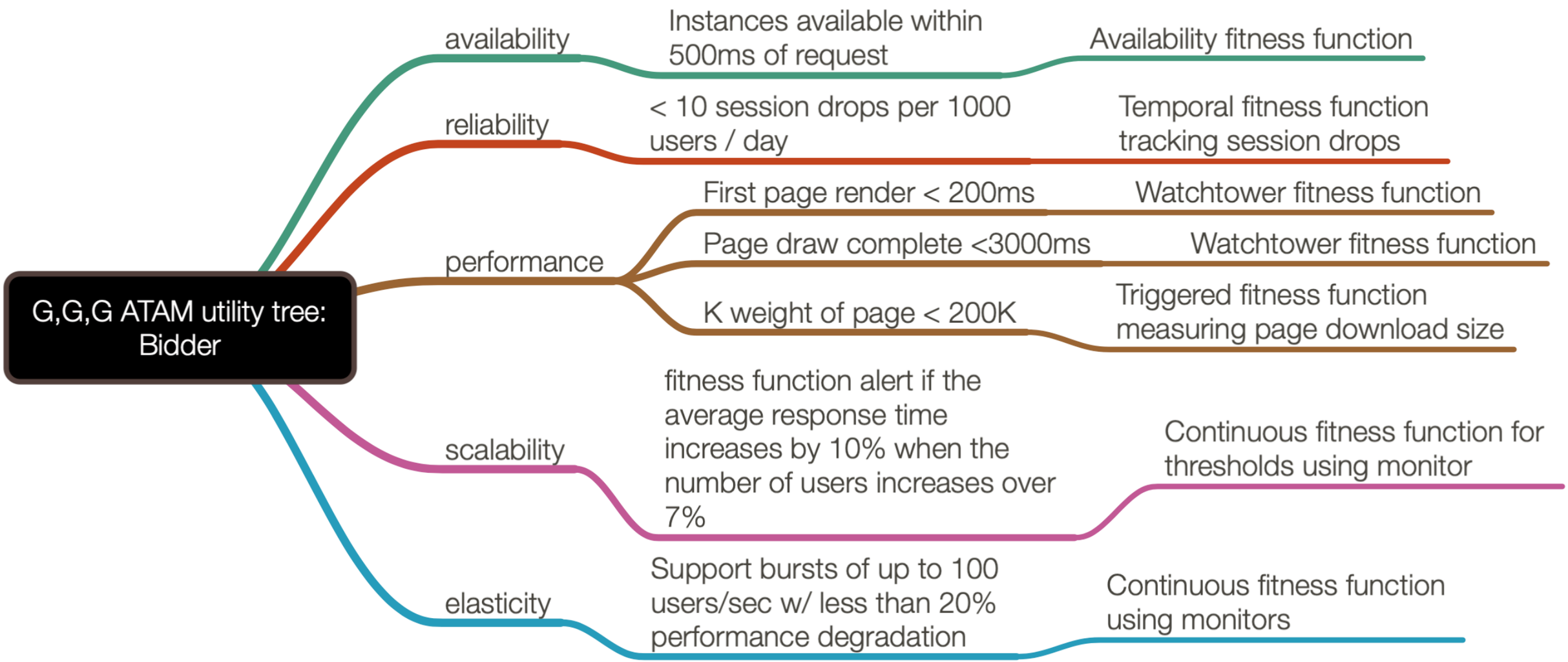
ATAM Utility Trees



Bidder Quantum ATAM Utility Tree



Bidder Quantum ATAM Utility Tree Tradeoff Analysis



Automating Architectural Governance

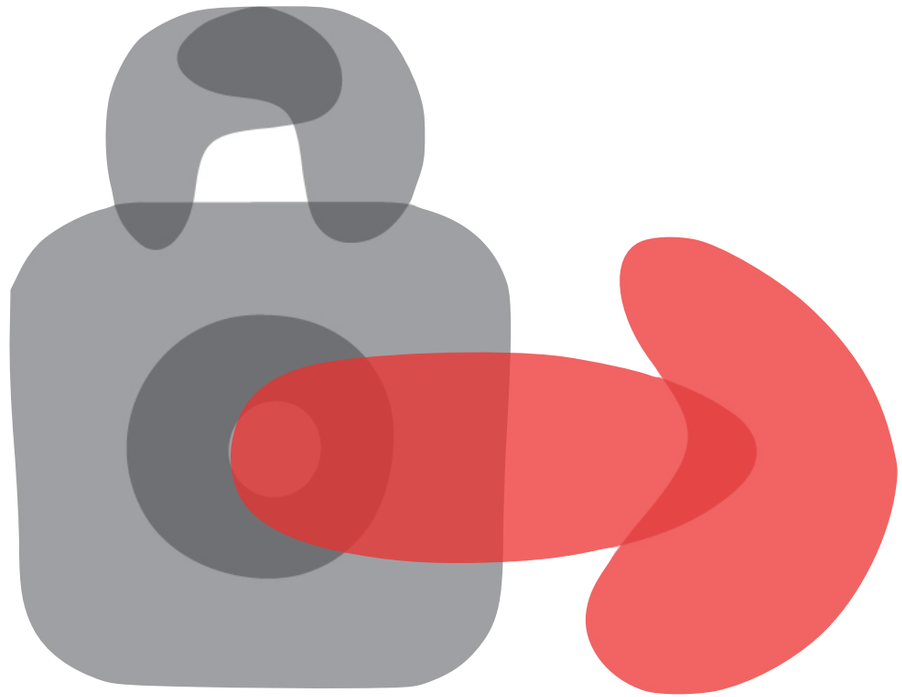
ThoughtWorks®

NEAL FORD

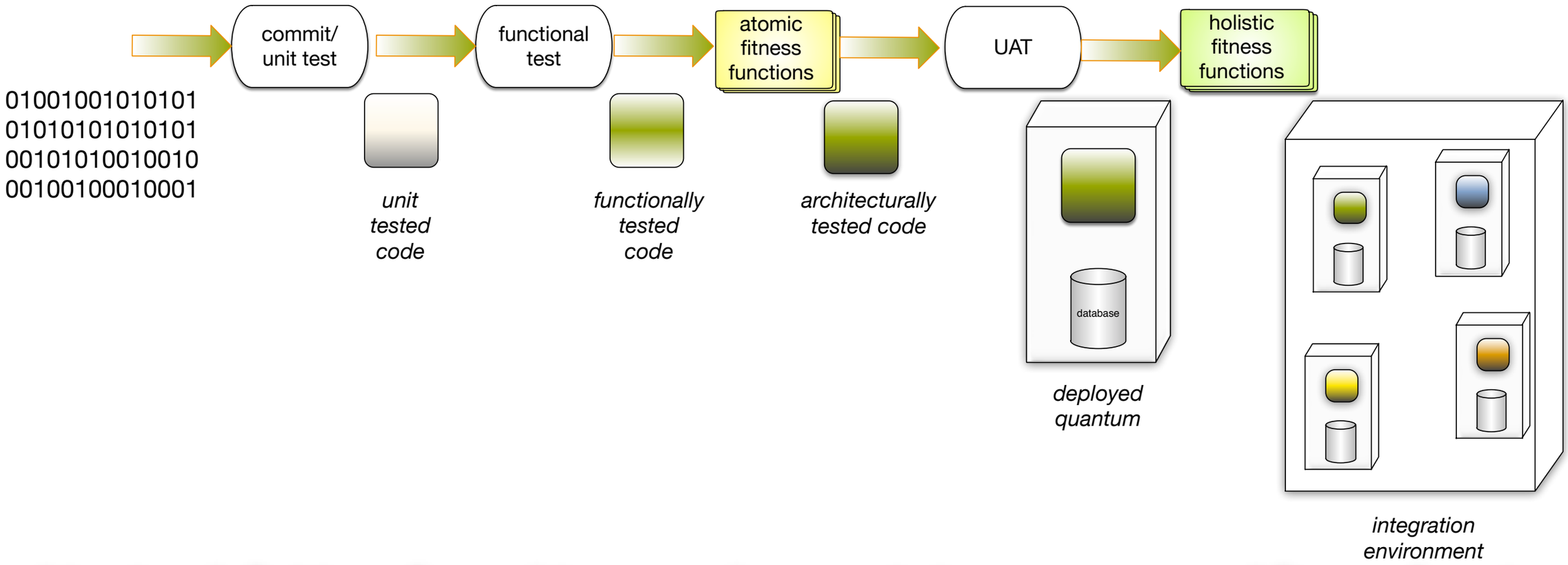
Director / Software Architect / Meme Wrangler

@neal4d

http://nealford.com

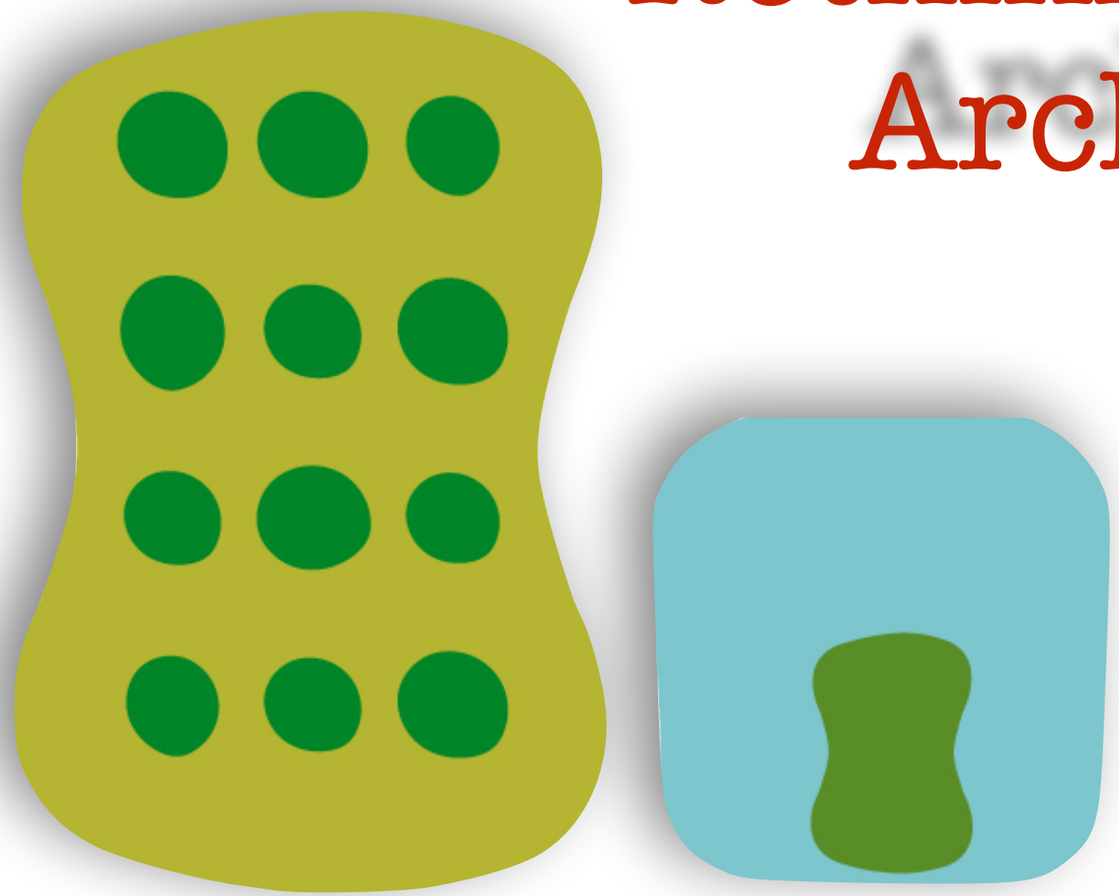


Automating

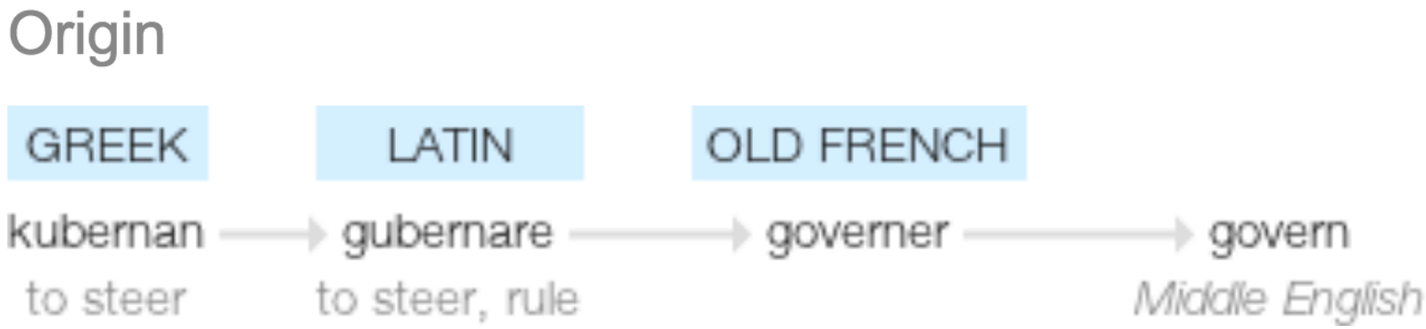


objective definitions for architecture characteristics => automated fitness functions

Rethink Enterprise Architecture



govern

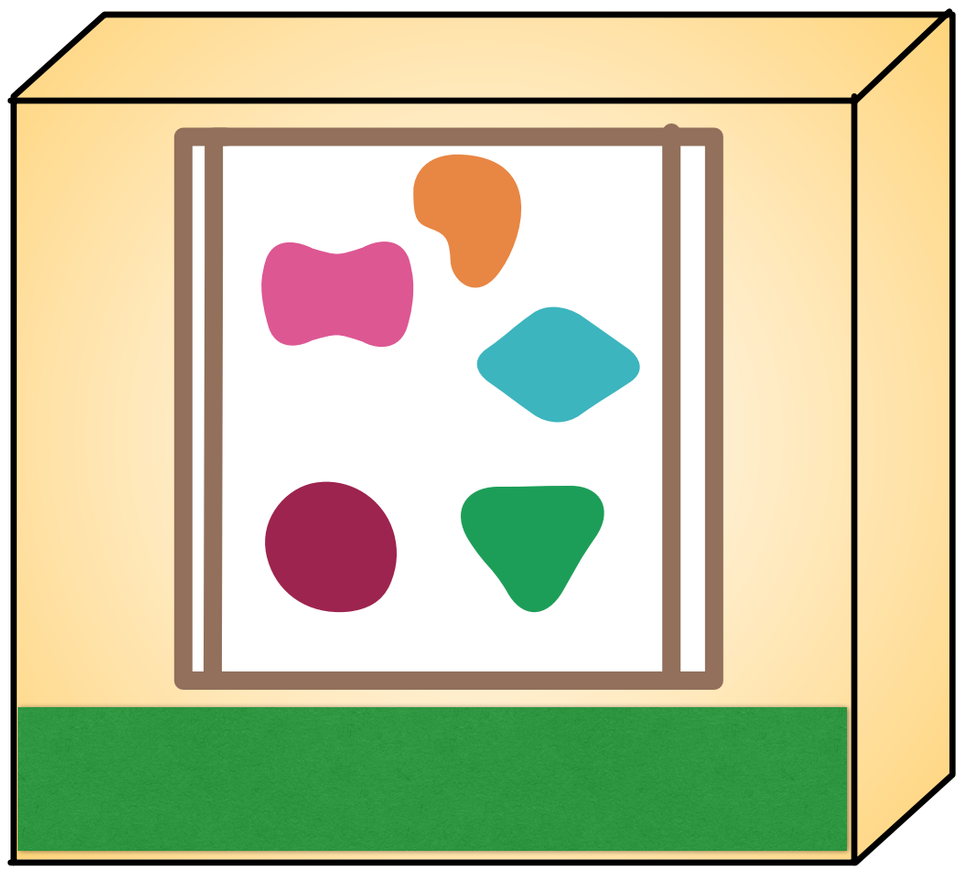


— to control, direct, or strongly influence the actions and conduct of

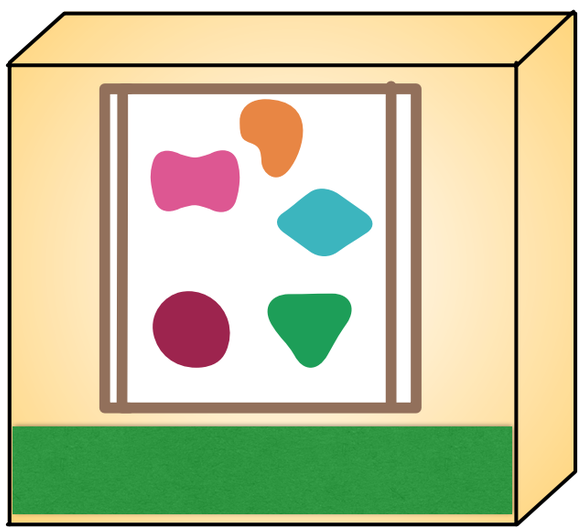
— to exert a determining or guiding influence in or over

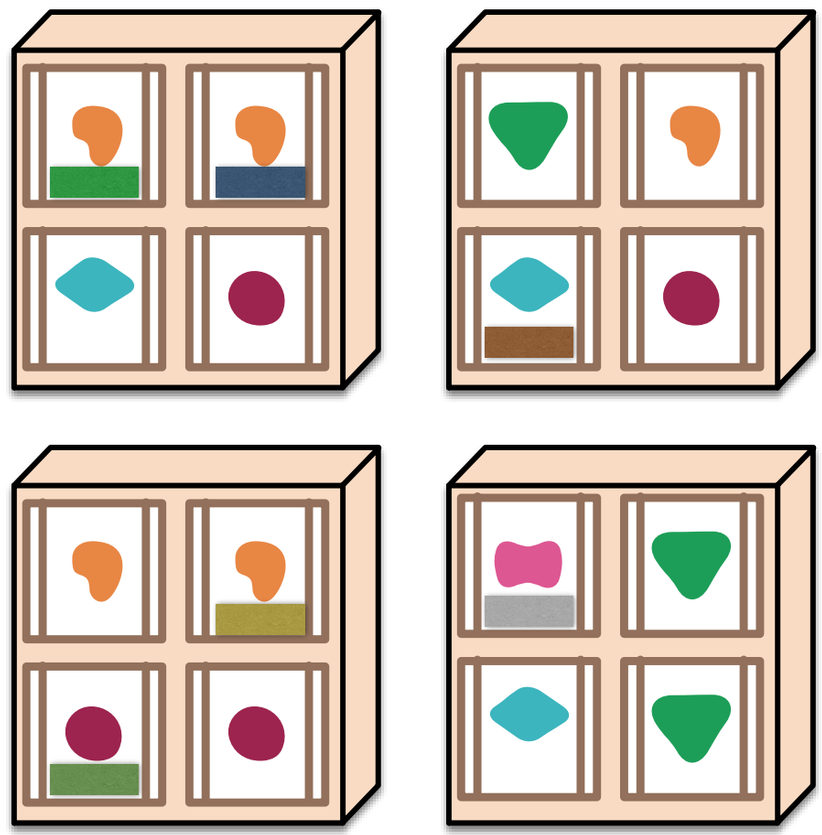
— to hold in check

Monolithic Governance



Decentralized Governance





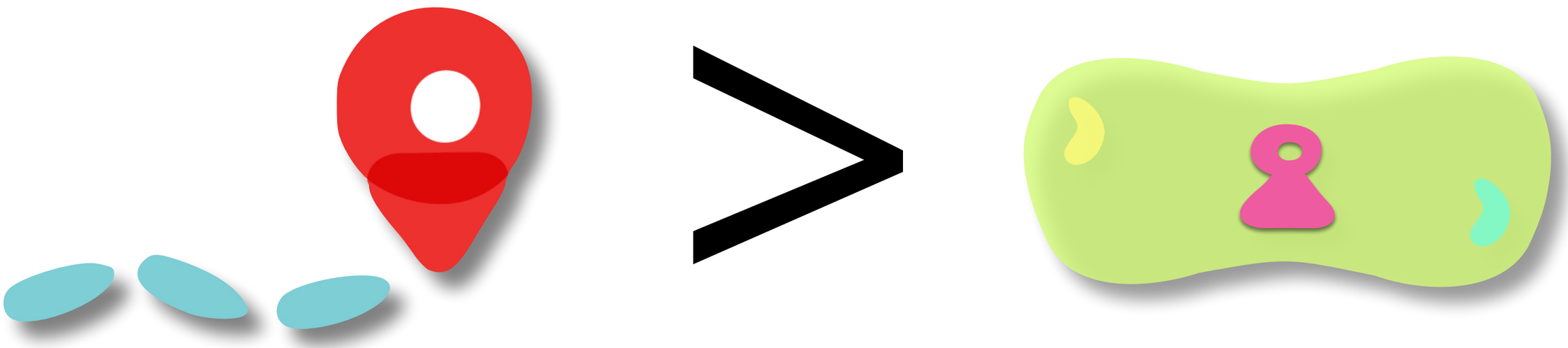
govern

Origin

— to serve as a precedent or deciding principle for

20th Century Governance

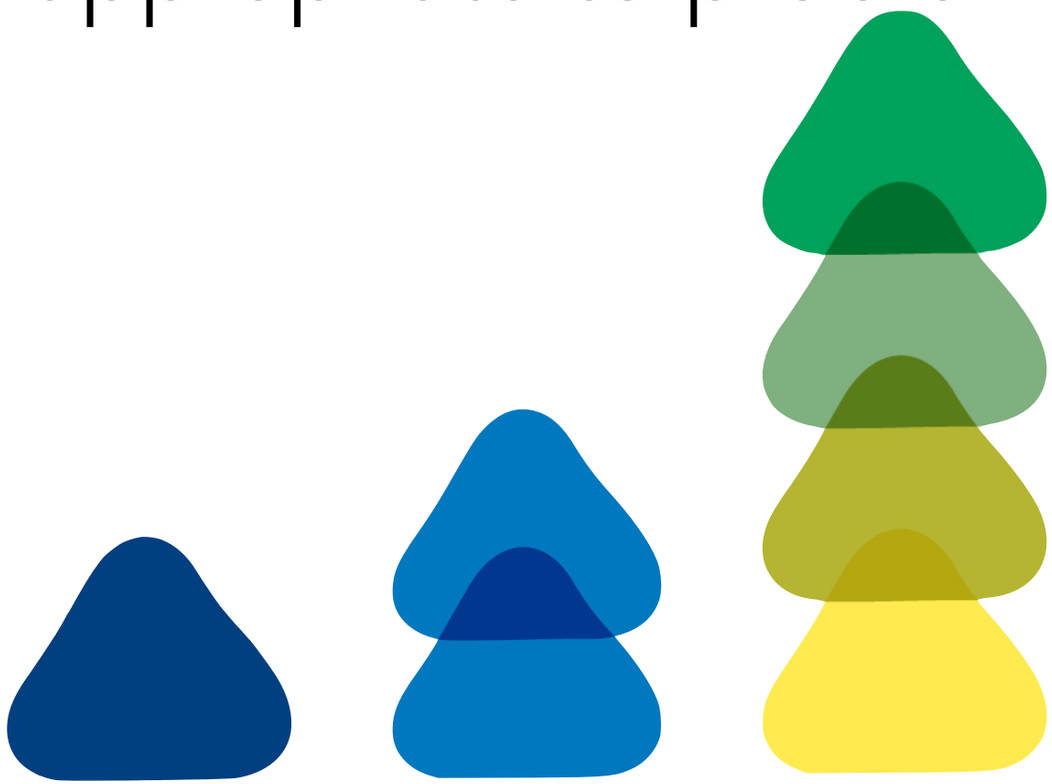
21st Century Governance

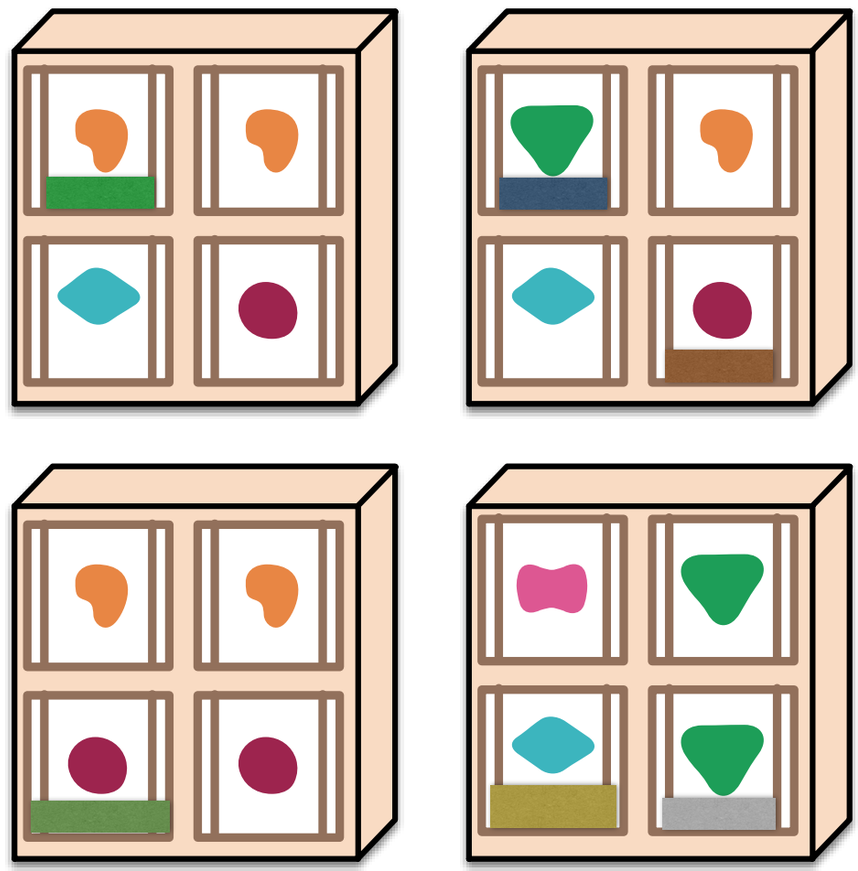


guidance over cost savings

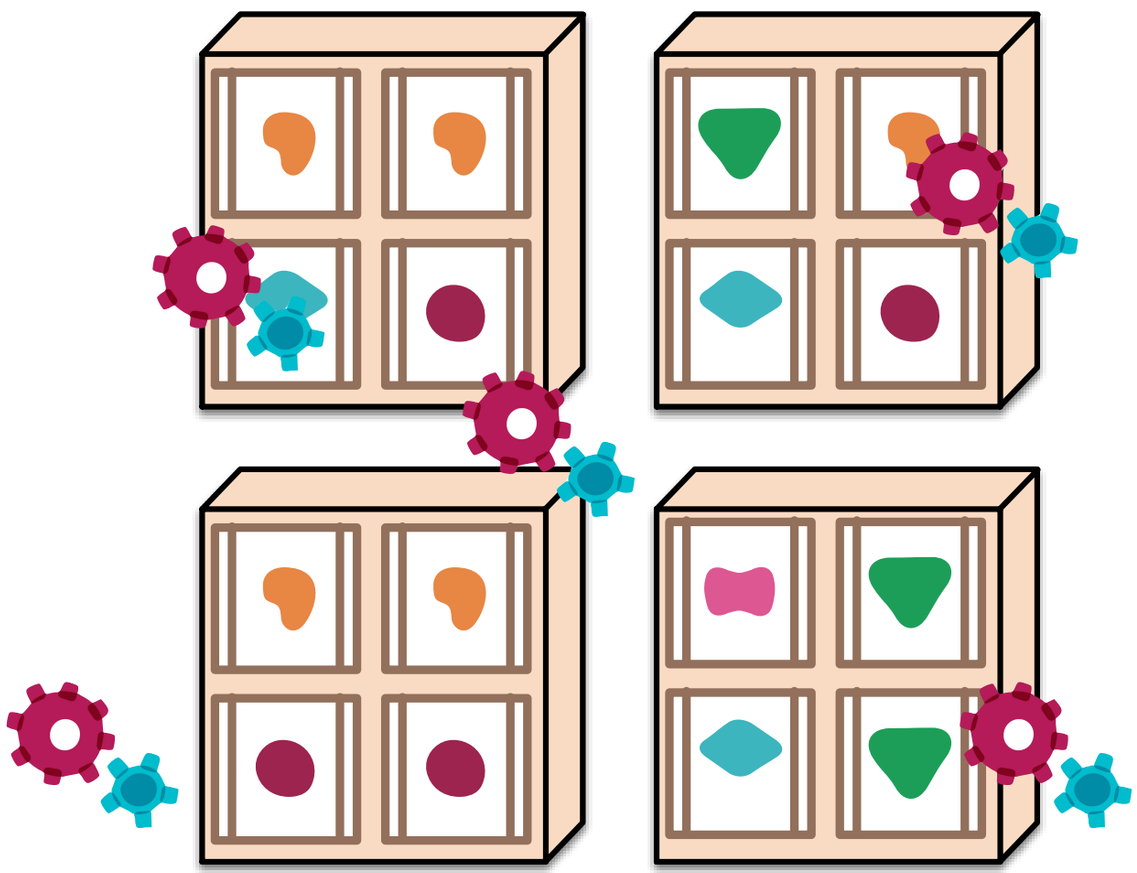
“Goldilocks” Governance

Choose technology stacks appropriate to problem scale.





Protection via Fitness Functions



Fitness Function Guidelines

most fitness function are local

Fitness Function Guidelines

most fitness function are local

global only when universally applied.

Fitness Function Guidelines

most fitness function are local

global only when universally applied.

No EA  states!

Fitness Function Guidelines

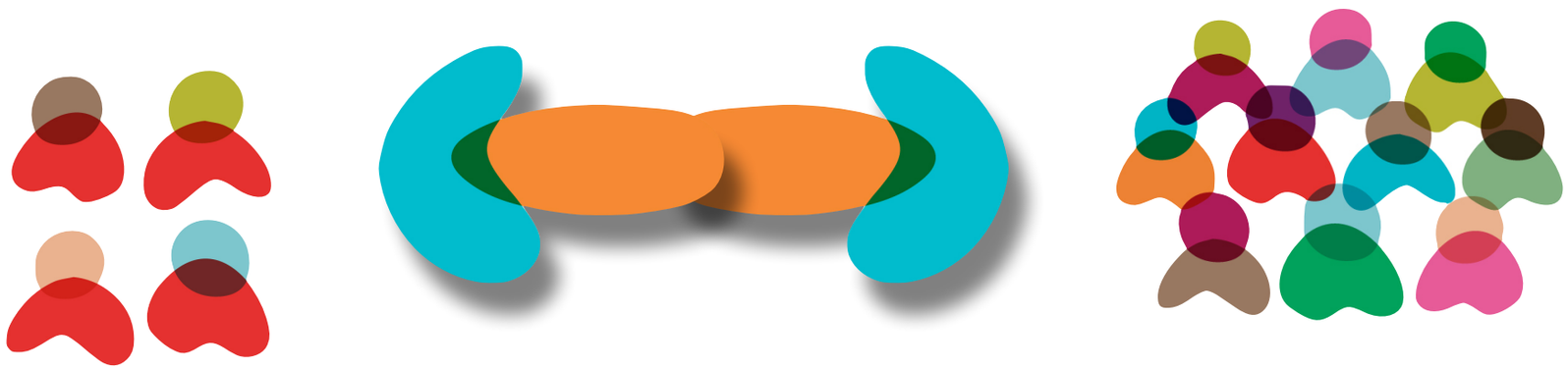
developers must collaborate

enterprise architects

developers

Fitness Function Guidelines

developers must collaborate



No EA  states!

Fitness Function Guidelines

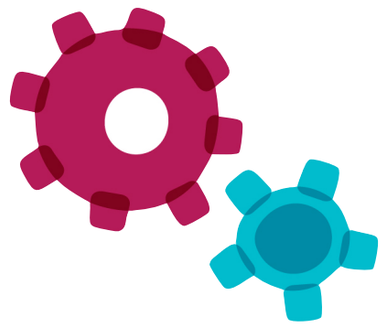
prefer automation but rely on manual when needed

Fitness Function Guidelines

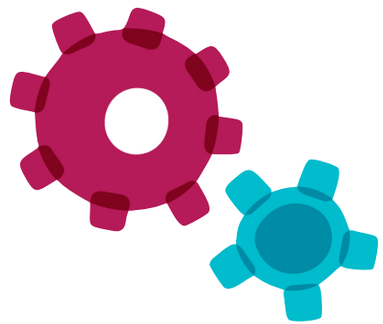
prefer automation but rely on manual when needed

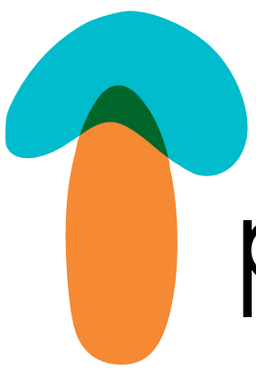
automation =  when needed

Architecture Defined

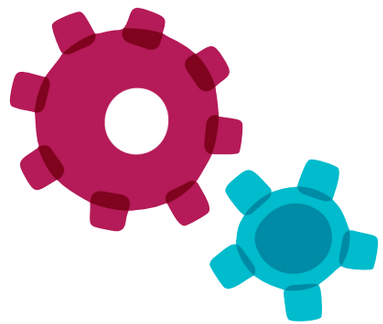
architecture =  + 

Architecture Defined

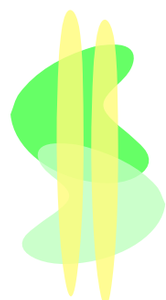
architecture =  + 

 prioritization

Architecture Defined

architecture =  + 

 prioritization

 cost

fitness functions = the mechanism of governance

Automating Architectural Governance

ThoughtWorks®

NEAL FORD

Director / Software Architect / Meme Wrangler

@neal4d

http://nealford.com

?'S