Evolving mutation from objects to the cloud

MUTATION workshop, Berlin, March 2011

Benoit Baudry

INSTITUT NATIONAL
DE RECHERCHE
EN INFORMATIQUE
ET EN AUTOMATIQUE

INRIA

centre de recherche RENNES - BRETAGNE ATLANTIQUE

Outline

- A perspective on testing in evolving software construction paradigms
- A methodological pattern: question-learn-test-feedback
- An illustration: aspect-oriented programming

1.

The scope of software testing practice and research is broadening

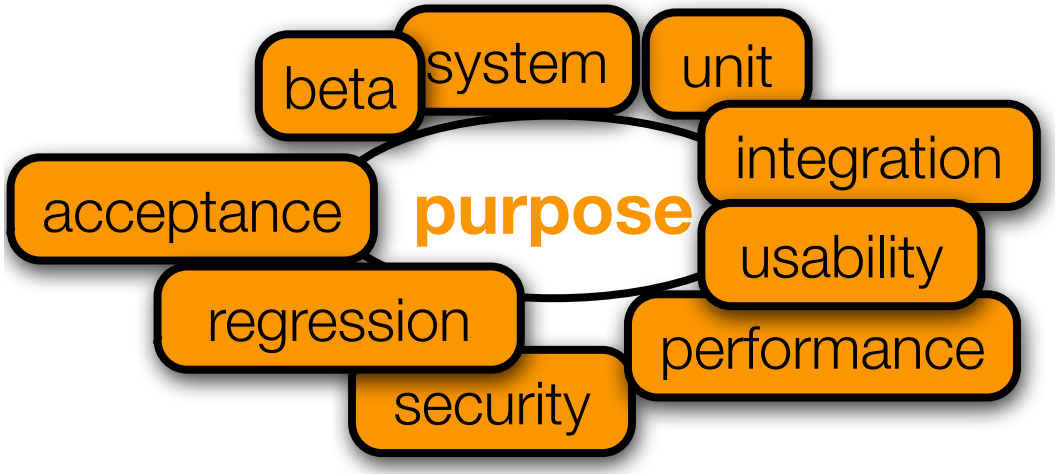
1.

The scope of software testing practice and research is widening

purpose

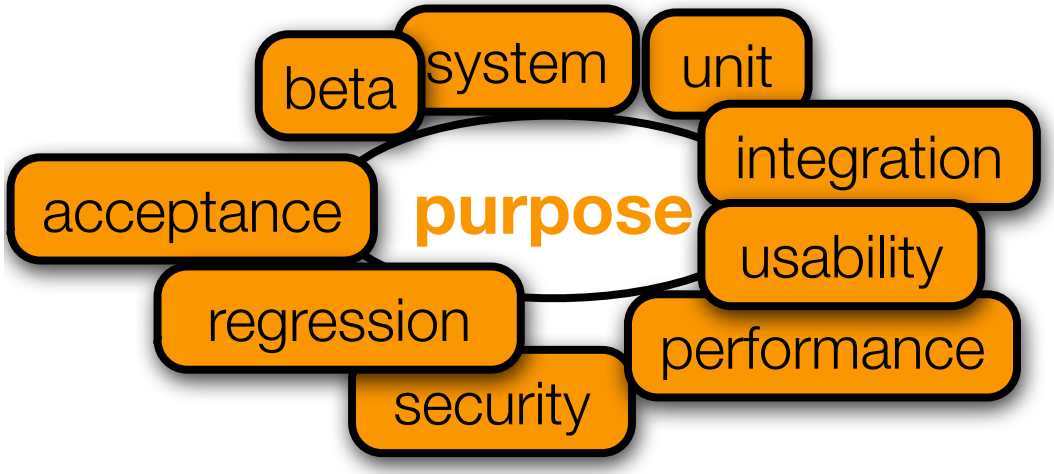
1.

The scope of software testing practice and research is widening



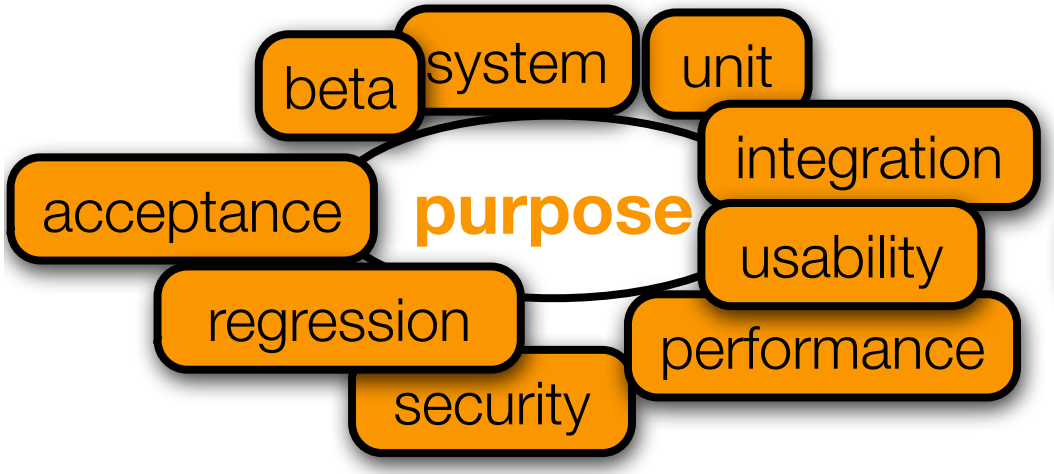
1.

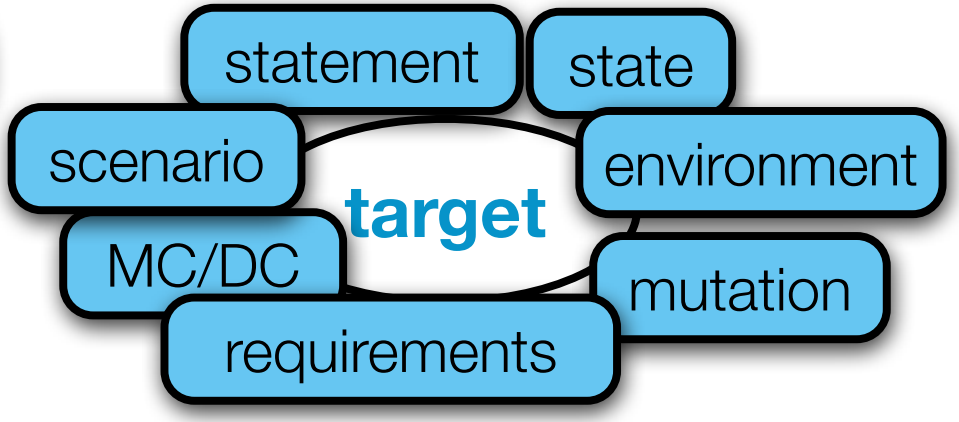
The scope of software testing practice and research is widening



1.

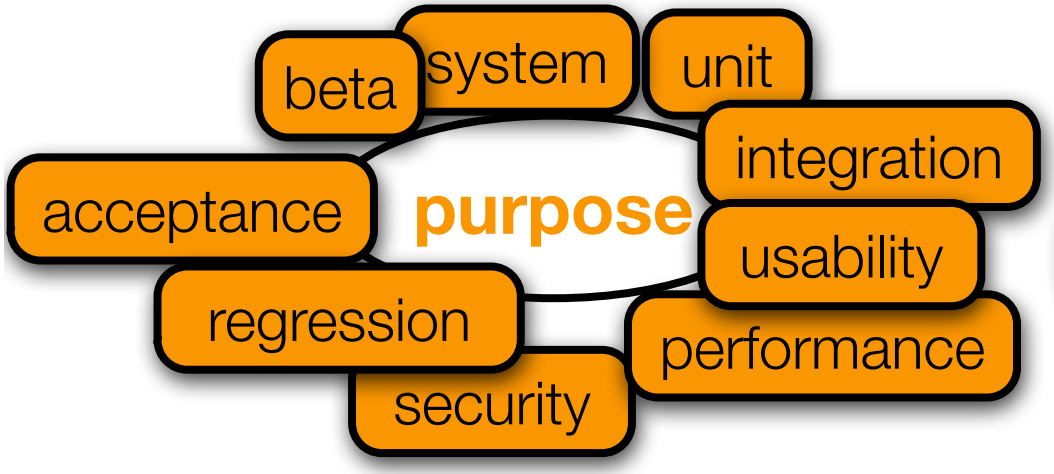
The scope of software testing practice and research is widening

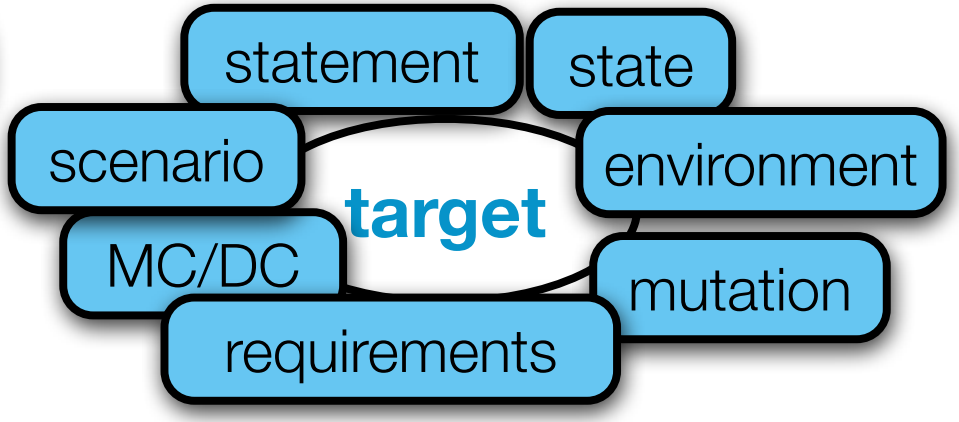




1.

The scope of software testing practice and research is widening

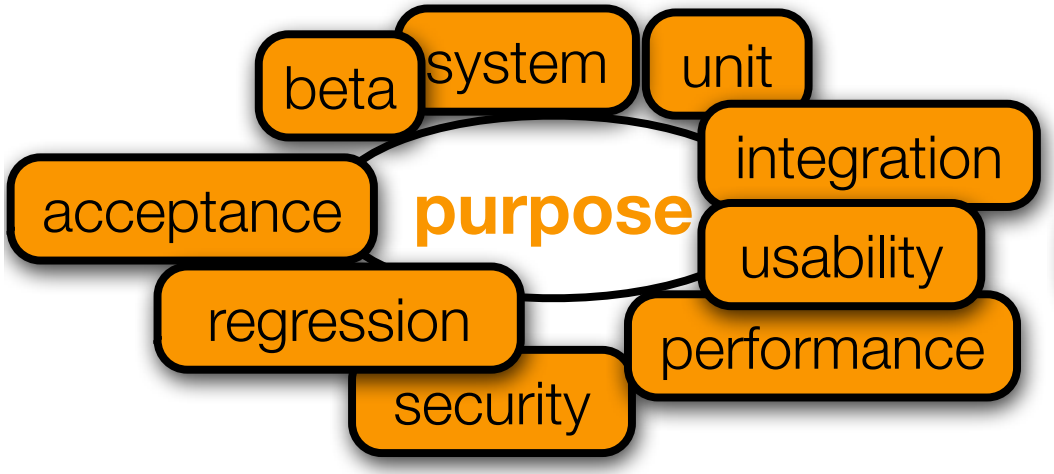


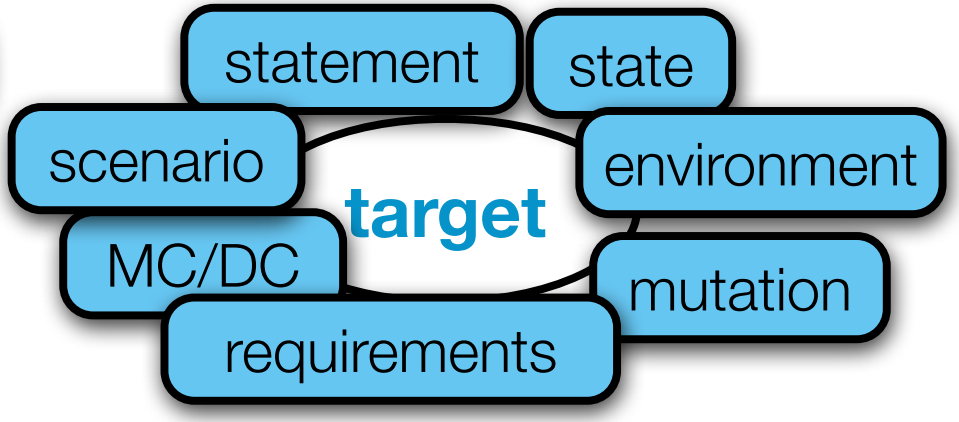


1.

The scope of software testing practice and research is widening

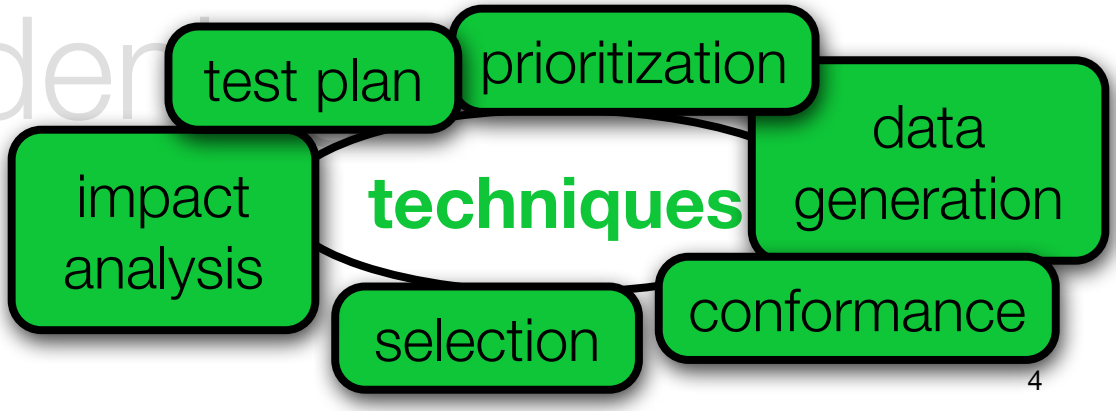
techniques

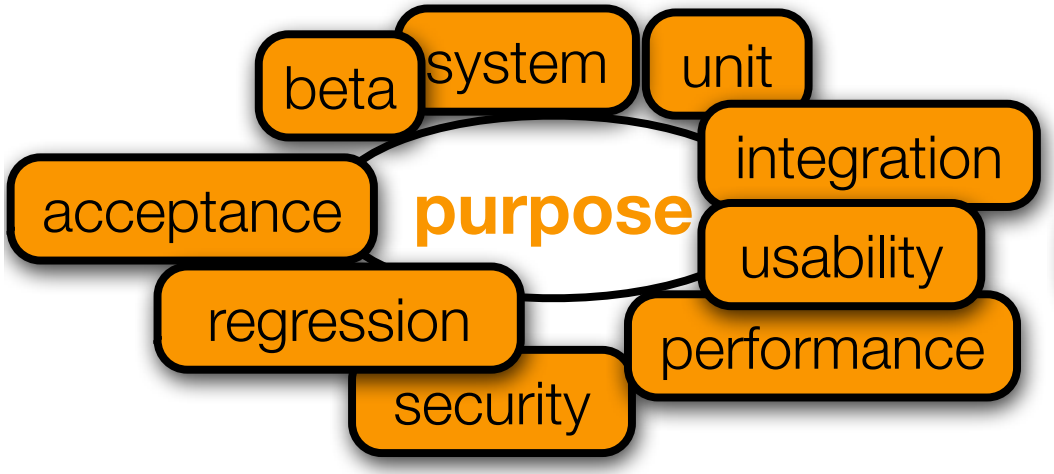


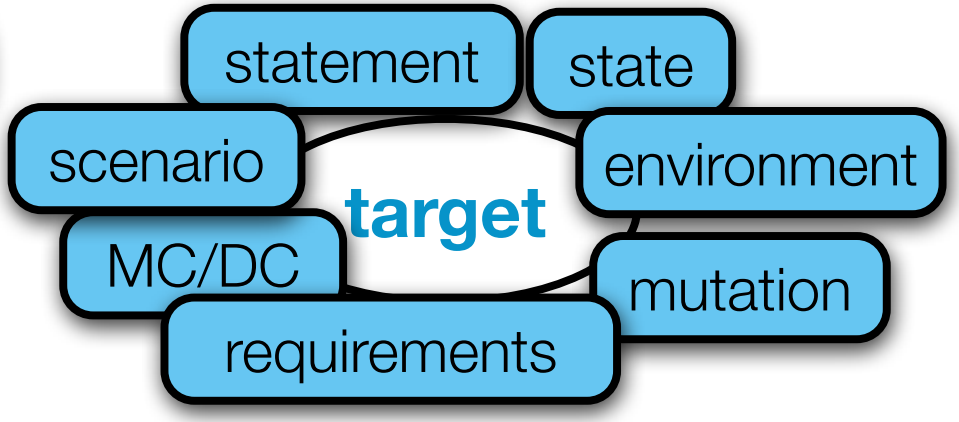


1.

The scope of software testing practice and research is widen

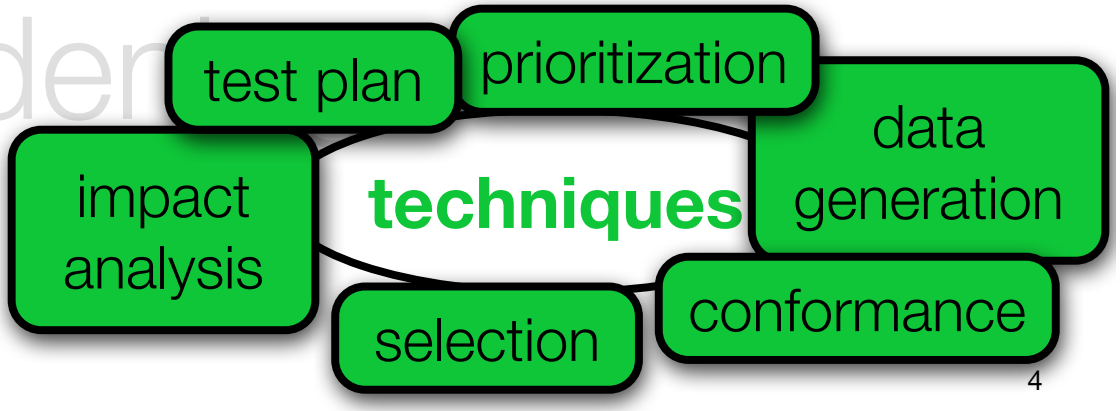






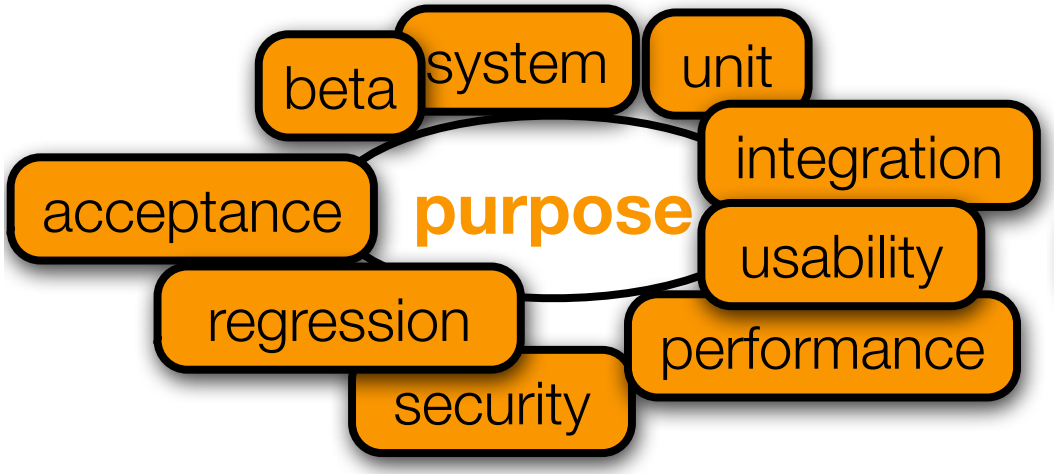
1.

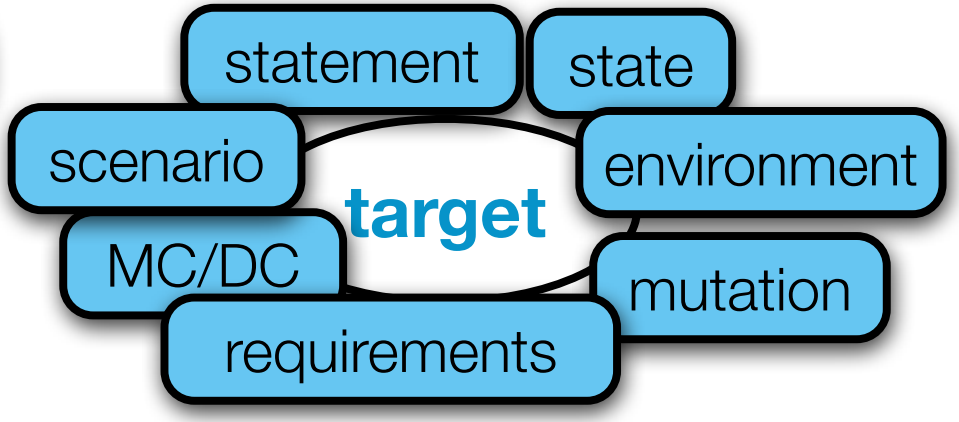
The scope of software testing practice and research is widen

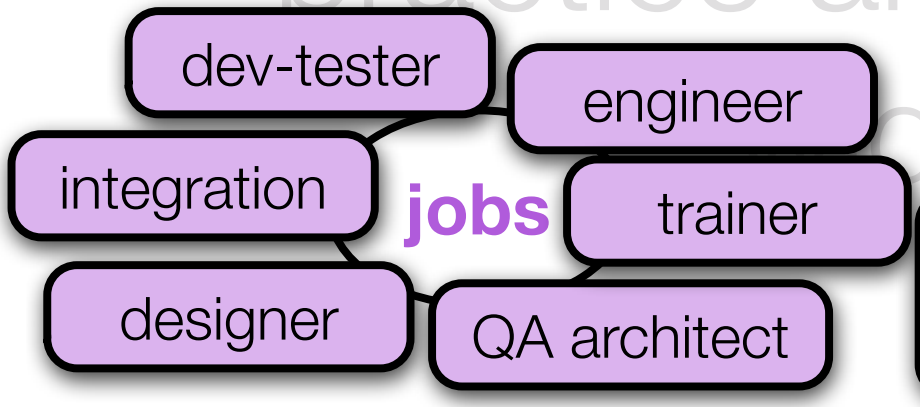


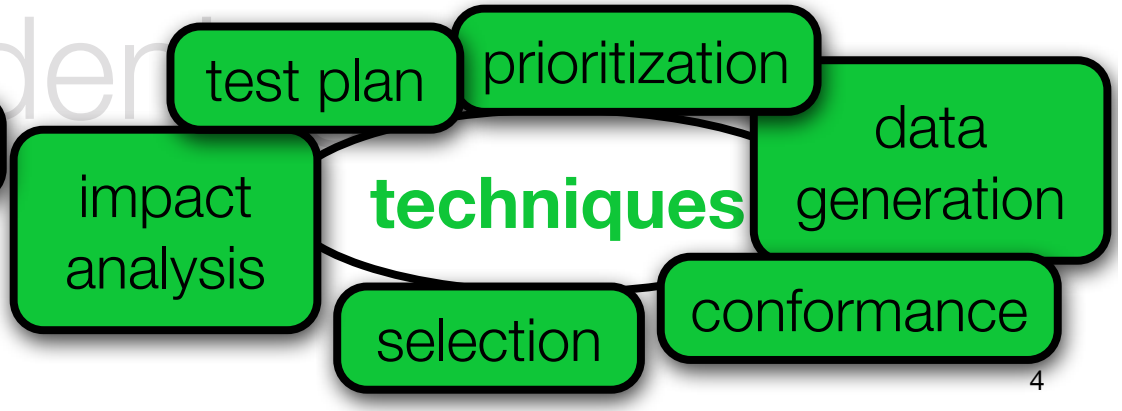
1.

The scope of software testing practice and research is enormous

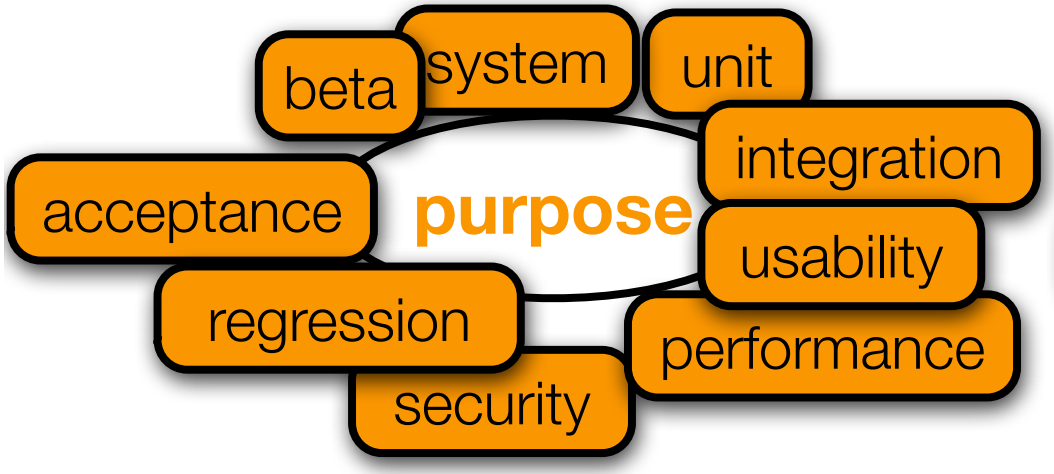


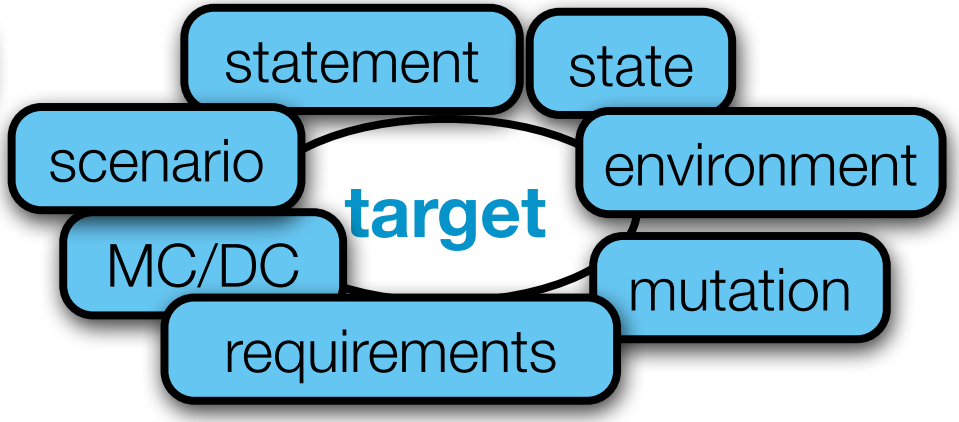


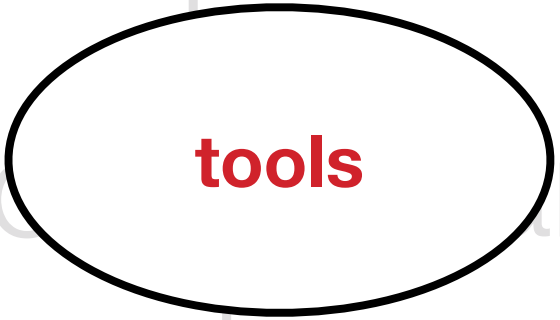


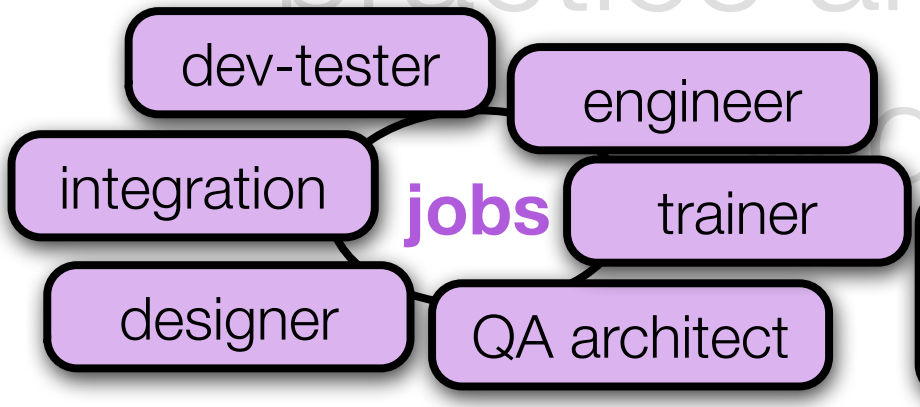


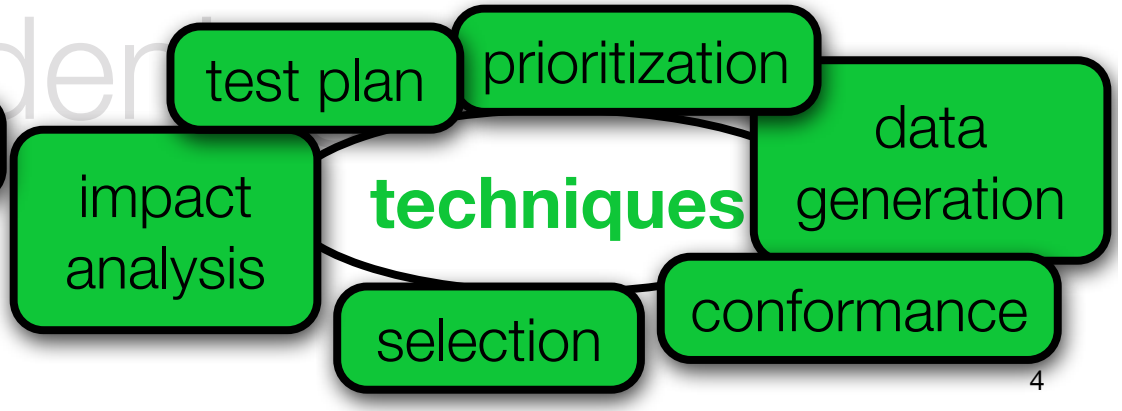
The scope of software testing practice and research is enormous!

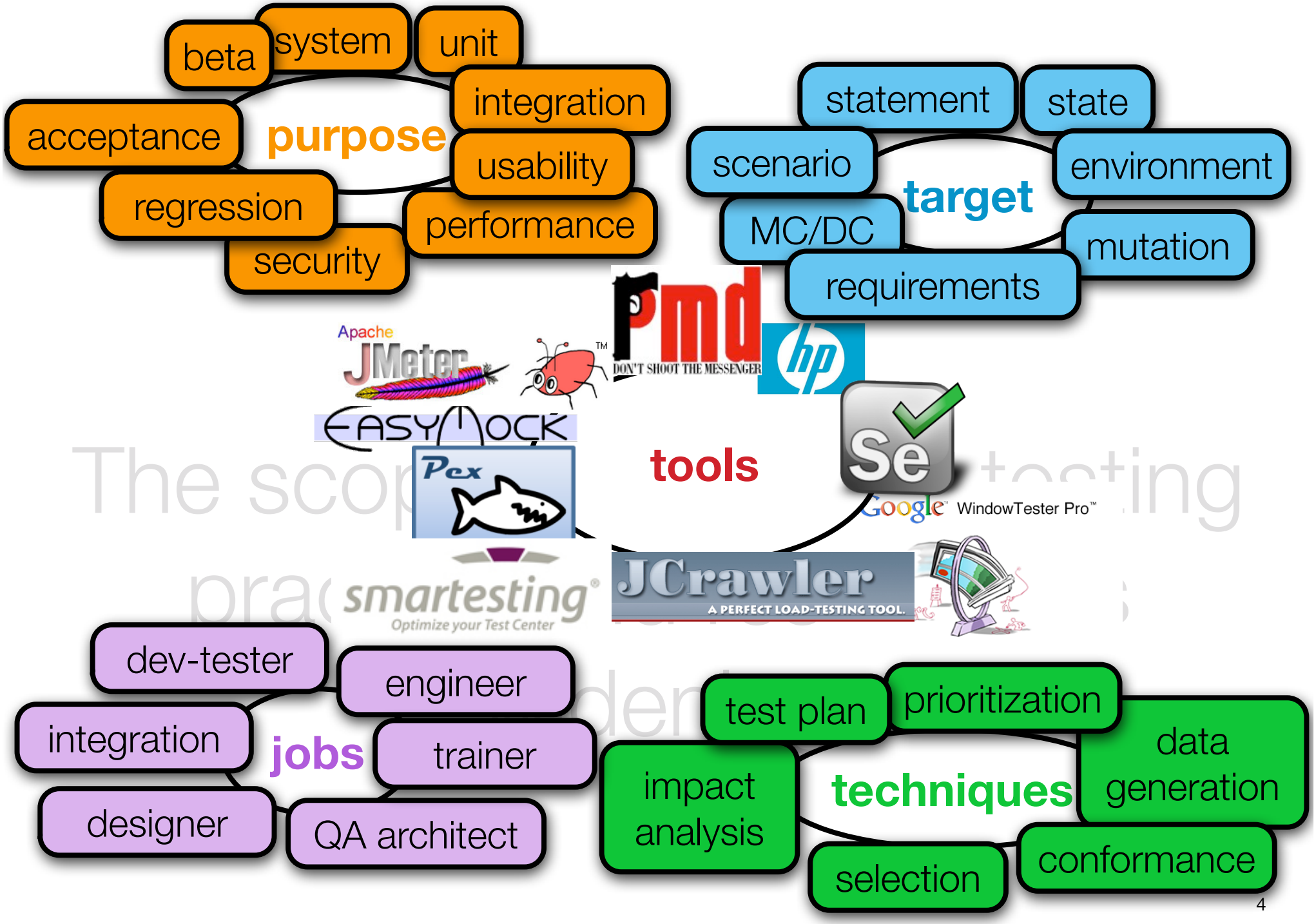











purpose
beta
system
unit
integration
acceptance
usability
regression
performance
security
target
statement
state
scenario
environment
MC/DC
mutation
requirements
tools
Apache JMeter
pmd
DON'T SHOOT THE MESSENGER
hp
EASYMOCK
Pex
Se
Google WindowTester Pro
smartesting
Optimize your Test Center
JCrawler
A PERFECT LOAD-TESTING TOOL.
jobs
dev-tester
engineer
integration
trainer
designer
QA architect
techniques
test plan
prioritization
impact analysis
data generation
selection
conformance

purpose

beta
system
unit
integration
acceptance
usability
regression
performance
security

target

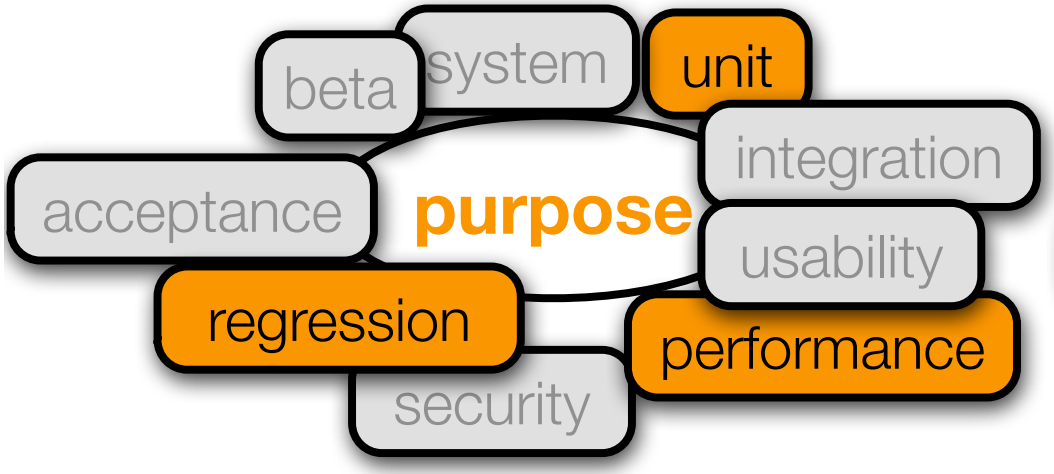
statement
state
scenario
environment
MC/DC
mutation
requirements

jobs

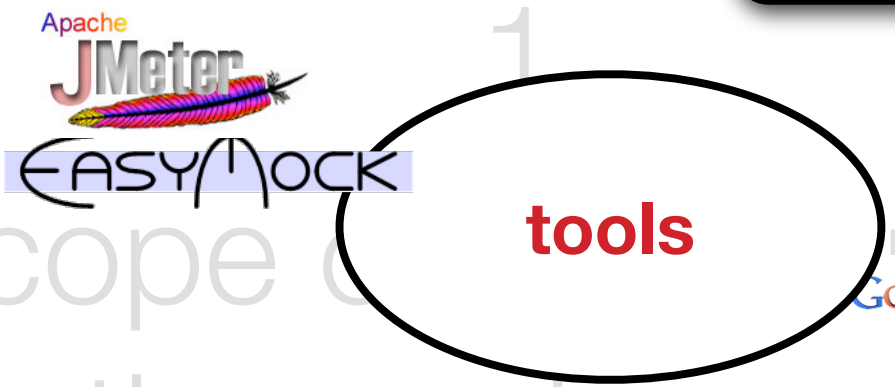
dev-tester
engineer
integration
trainer
designer
QA architect

techniques

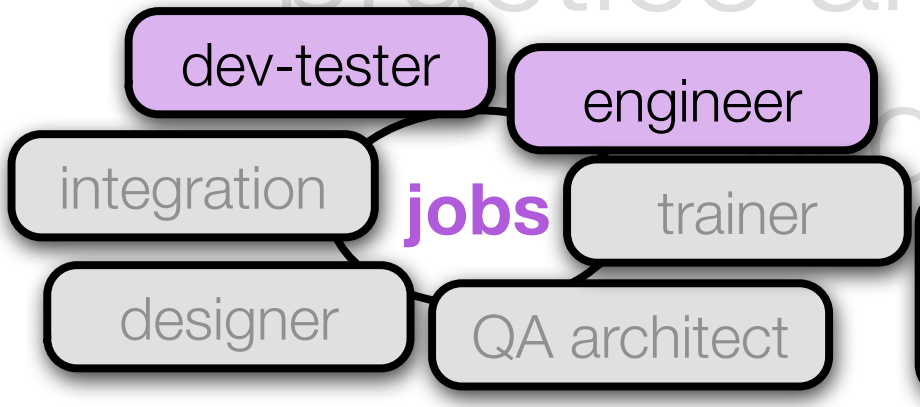
test plan
prioritization
impact analysis
data generation
selection
conformance

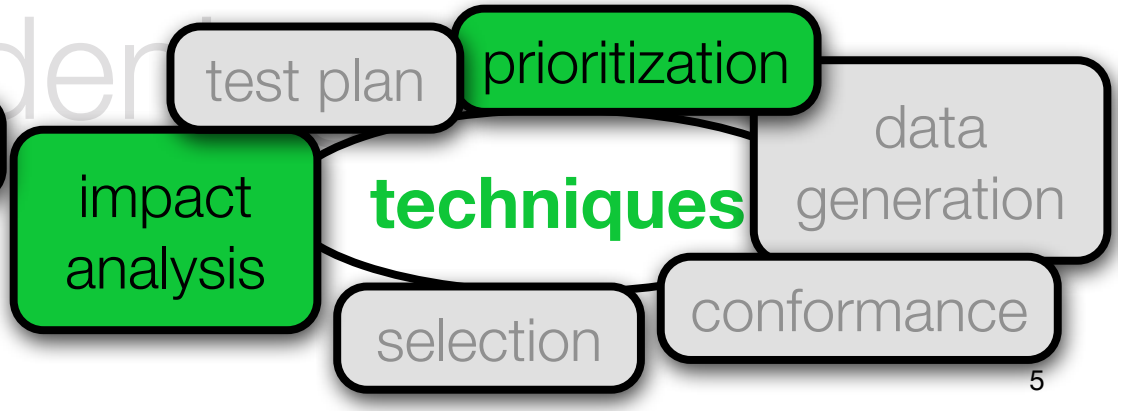

purpose
beta
system
unit
integration
acceptance
usability
regression
performance
security


target
statement
state
scenario
environment
MC/DC
mutation
requirements


Apache
JMeter
EASYMOCK
tools


Google
WindowTester Pro™


jobs
dev-tester
engineer
integration
trainer
designer
QA architect


techniques
test plan
prioritization
impact analysis
data generation
selection
conformance

The scope of software testing practice and research is enormous

purpose

- system
- unit
- integration
- usability
- performance
- security
- regression
- acceptance
- beta

target

- statement
- state
- environment
- mutation
- requirements
- MC/DC
- scenario

tools

jobs

- engineer
- trainer
- QA architect
- designer
- integration
- dev-tester

techniques

- test plan
- prioritization
- data generation
- conformance
- selection
- impact analysis

2.

because of (i) new ways of building software systems to (ii) face the growing diversity of applications and requirements for software systems

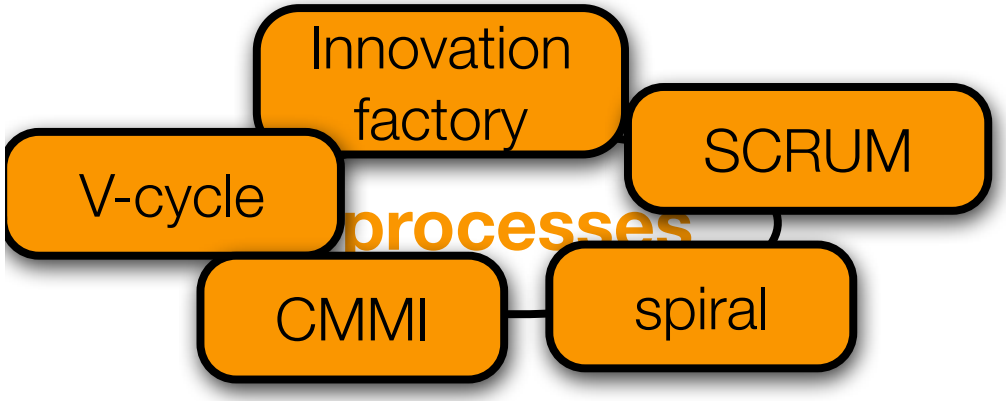
2.

because of (i) new ways of building software systems to

processes

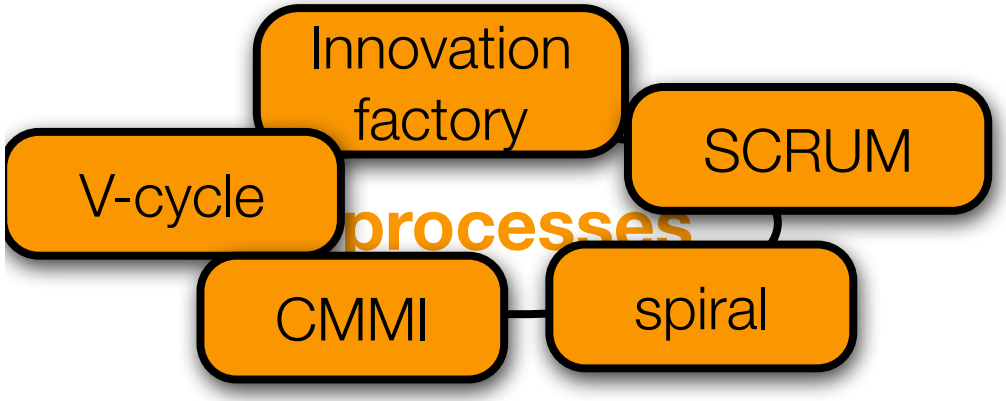
2.

because of (i) new ways of building software systems to



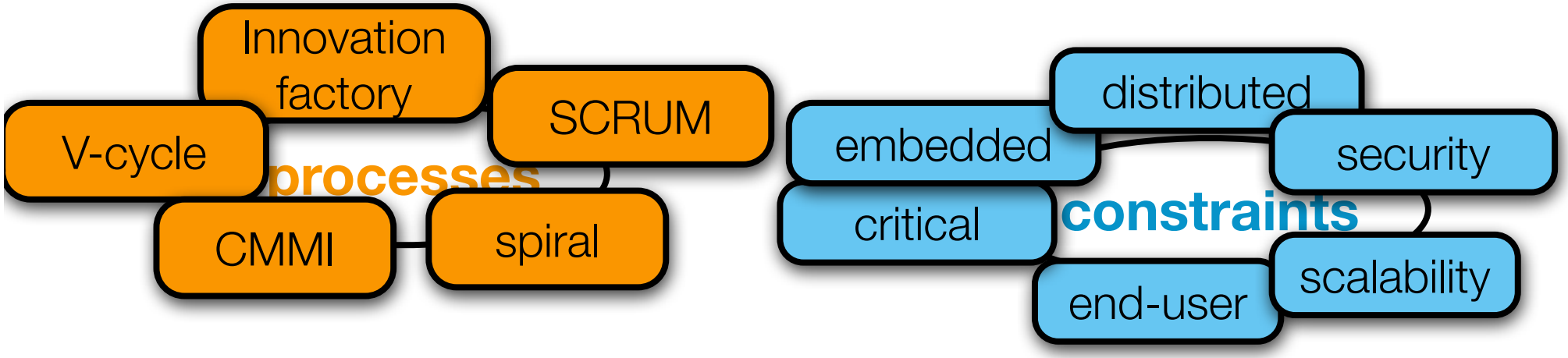
2.

because of (i) new ways of building software systems to



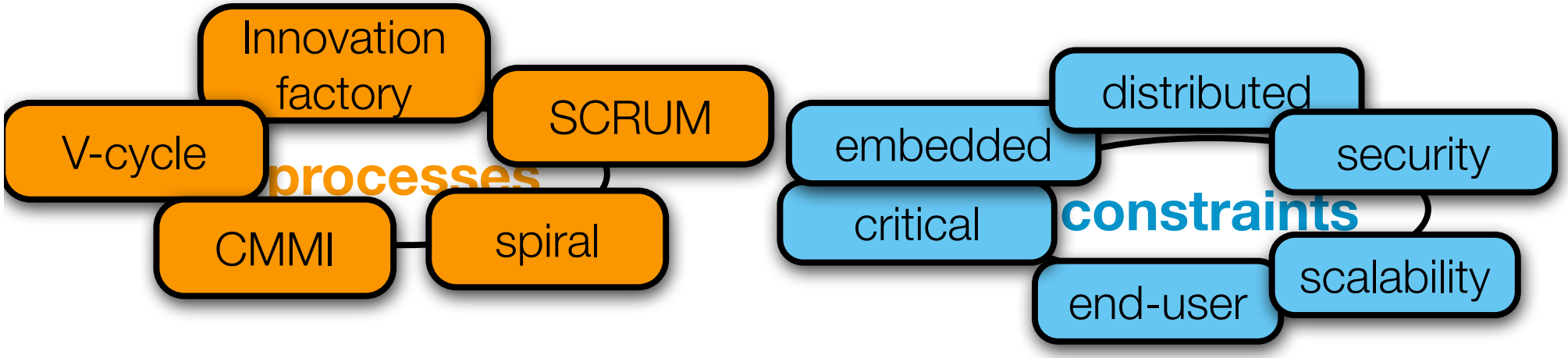
2.

because of (i) new ways of building software systems to



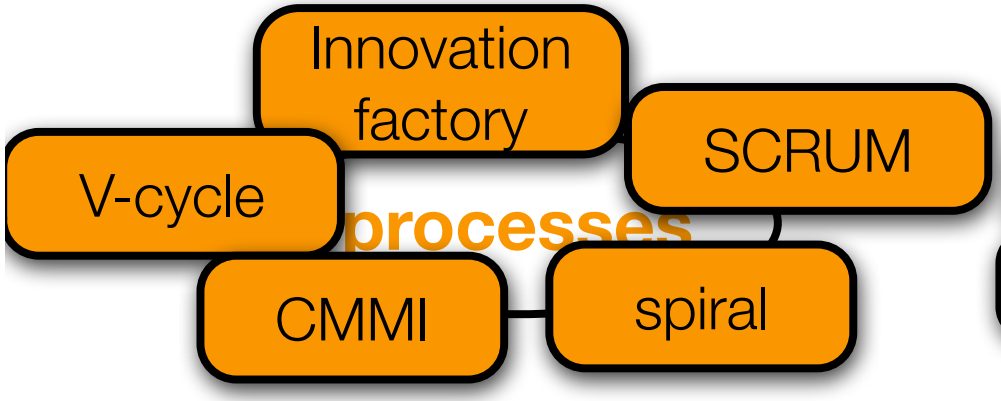
2.

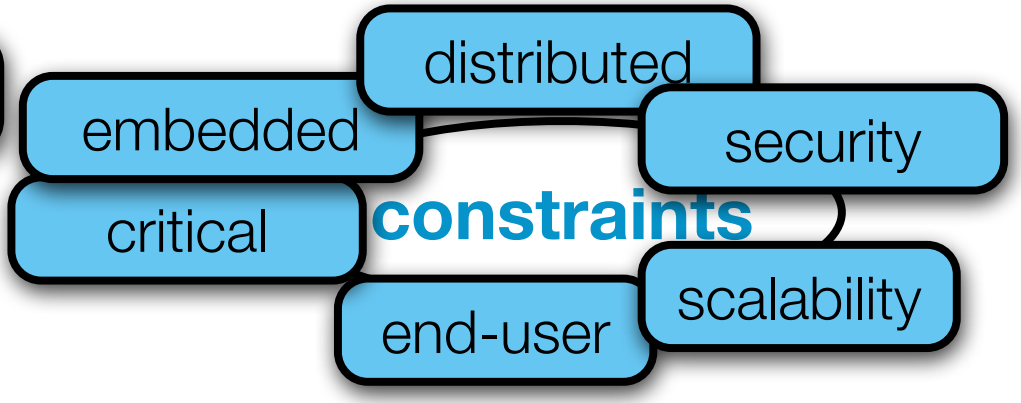
because of (i) new ways of building software systems to



2.

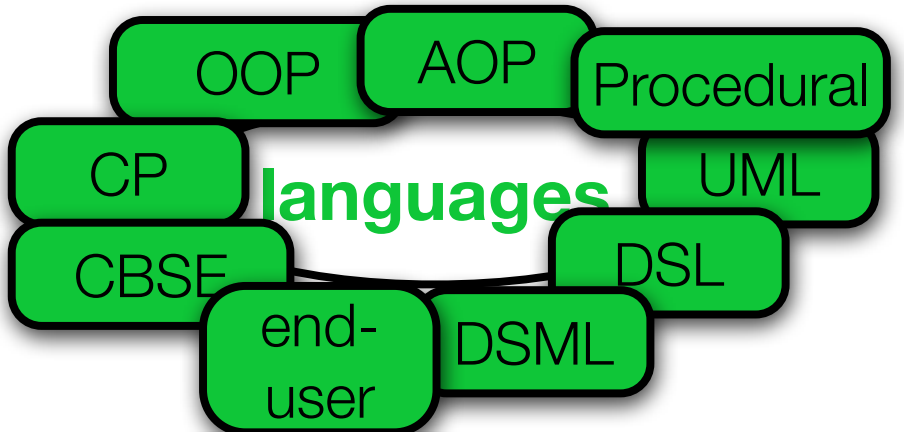
because of (i) new ways of building software systems to

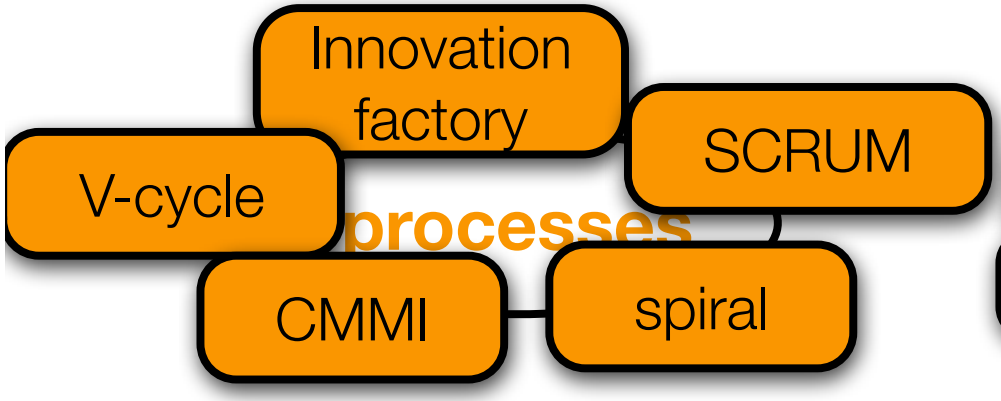


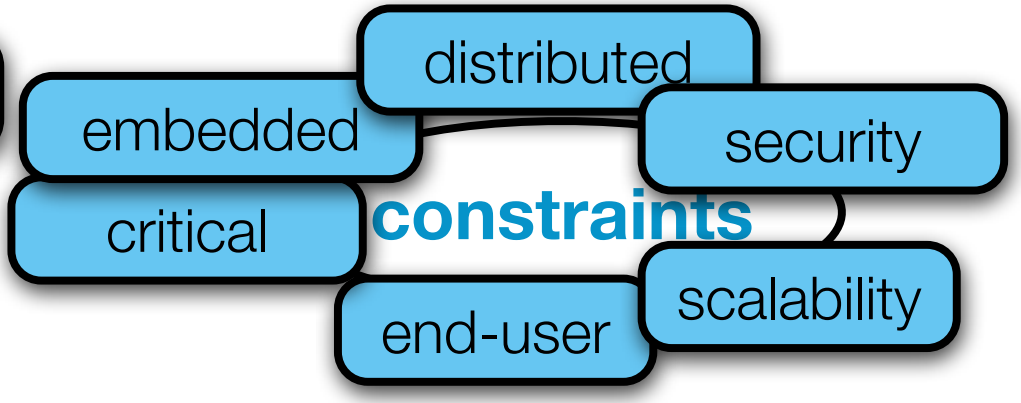


2.

because of (i) new ways of building software systems to

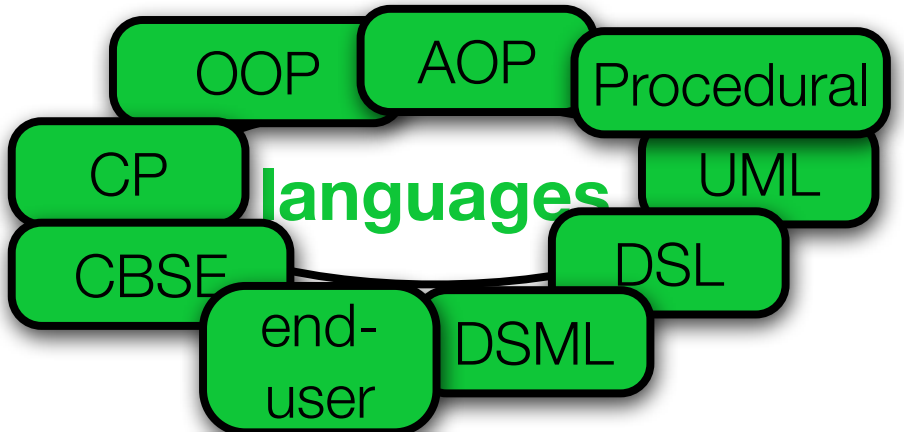


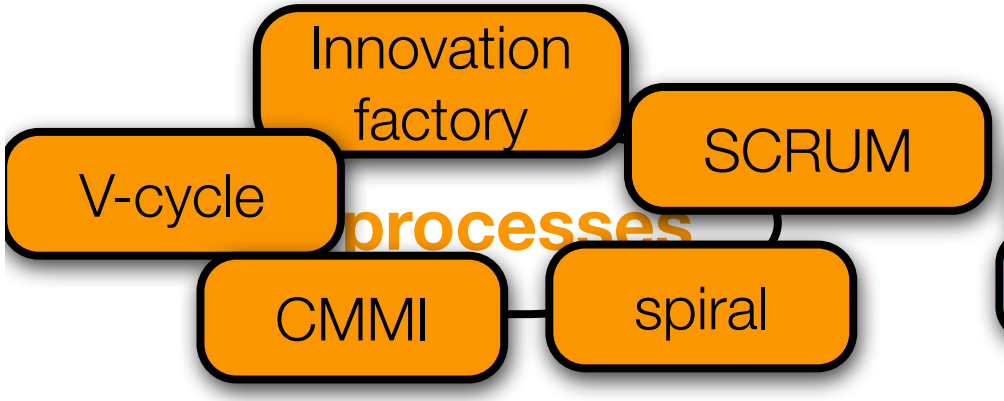


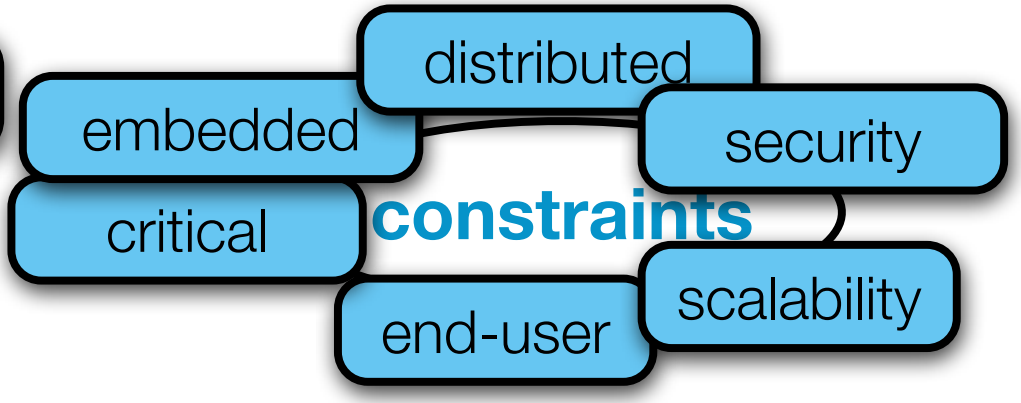


2.

because of (i) new ways of building software systems to

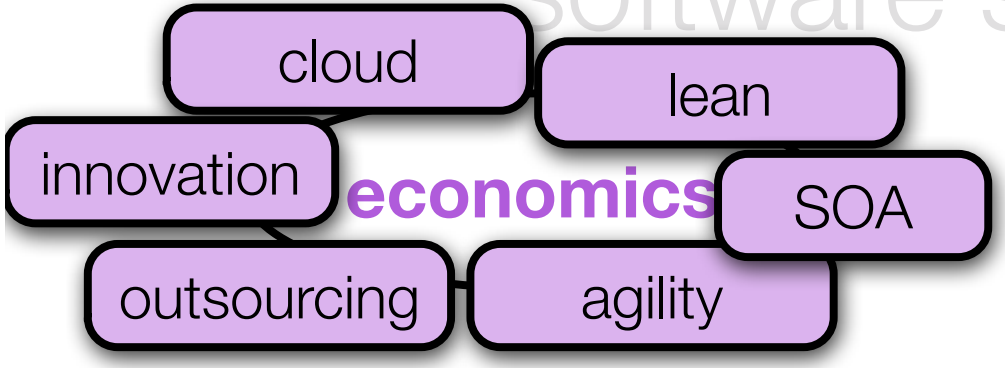


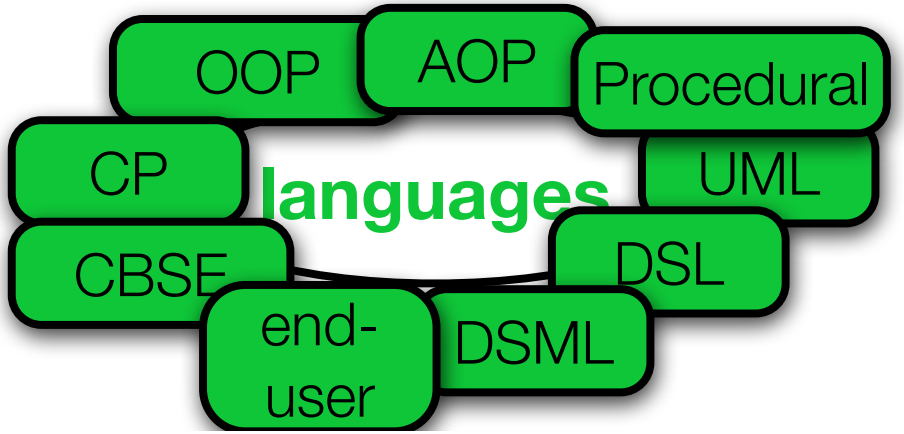


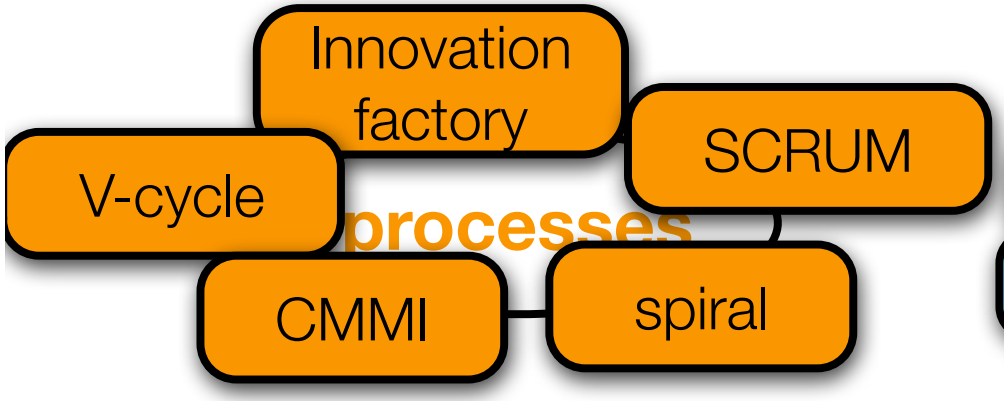


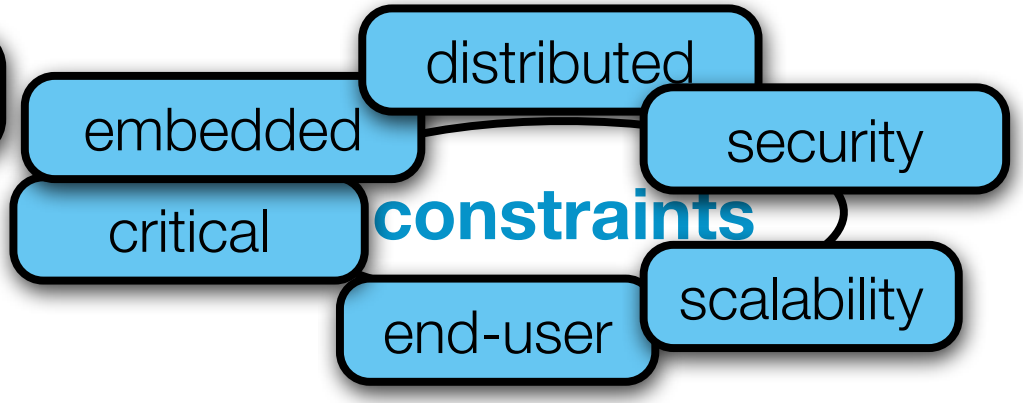
2.

because of (i) new ways of building software systems to

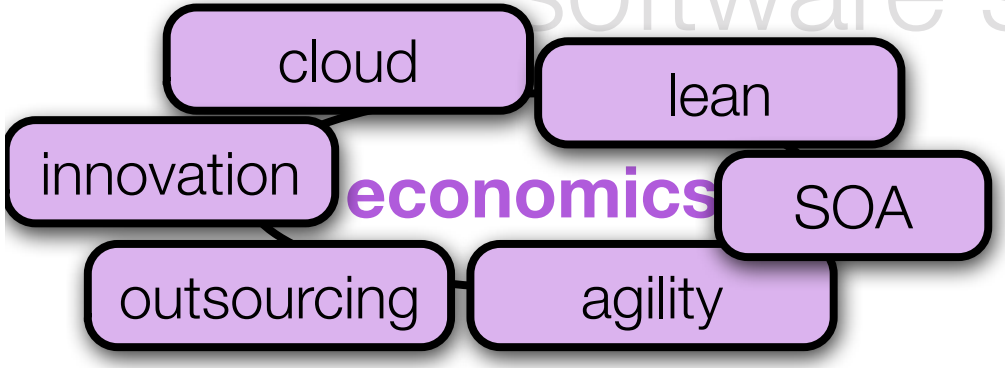


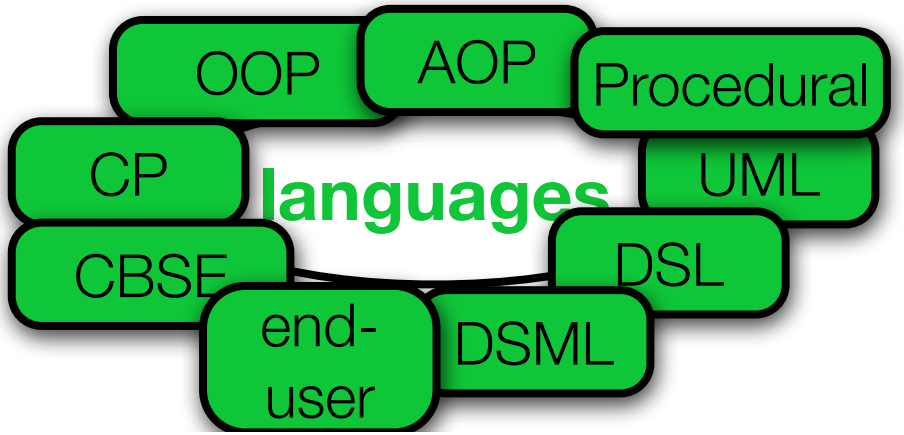






because of (i) new ways of building software systems to





because our job is building software systems to

processes

Innovation factory
SCRUM
V-cycle
CMMI
spiral

constraints

distributed
embedded
security
critical
scalability
end-user

environments

J2EE Java 2 Platform Enterprice Edition
eclipse 2.1 The Eclipse Project
spring source
GWT
maven
ESTEREL Technologies
sonar
Cradle From concept to creation
MATLAB SIMULINK

economics

cloud
lean
innovation
SOA
outsourcing
agility

languages

OOP
AOP
Procedural
CP
UML
CBSE
DSL
end-user
DSML

Innovation factory

V-cycle

SCRUM

processes

CMMI

spiral

distributed

embedded

security

critical

constraints

end-user

scalability

lean

innovation

economics

SOA

outsourcing

agility

OOP

AOP

Procedural

CP

languages

UML

CBSE

DSL

end-user

DSML

(ii) face the growing diversity of applications and requirements for software systems

(ii) face the of
application or
so

(ii) face th of
application or
so

(ii) face th
application
so

of
or


12:00
Jeu 09/12/2010
Arthur H
Confessions Nocturnes
04:17
4/10
LOUD
AUTO
A/C
20.5
AUTO

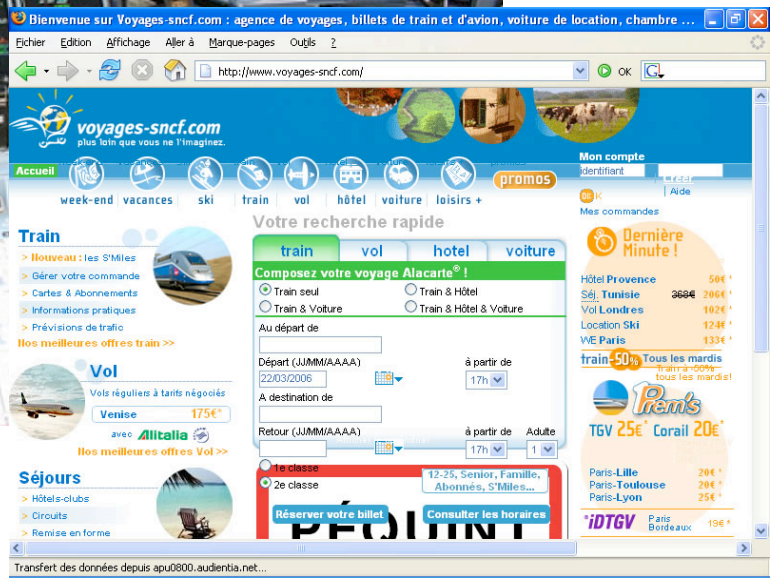

NASDAQ
NASDAQ
NASDAQ
NASDAQ
NASDAQ
NASDAQ
BUY ORDERS
19.12
+1.10
+1.0%
+1.2%
+0.27
17.15
+0.17
30.89

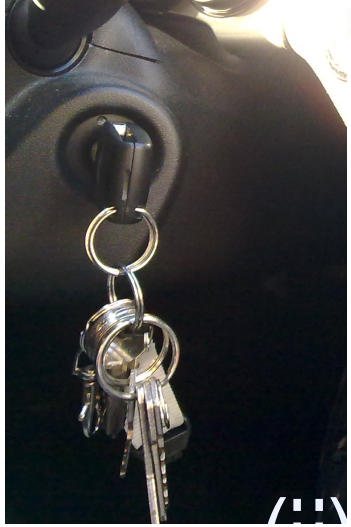

AUTO CURTAIN
HOME ENTERTAINMENT
COMMUNICATION
MOODS
HOME AUTOMATION
GARAGE
AUTO SPRINKLERS
ON
OFF
AUTO LIGHTING
HOME BACK UP SYSTEM
SECURITY
eCAM


12:00
Jeu 09/12/2010
Arthur II
Confessions Noctu
04:17
4/10
LOUD
A/C
AUTO
20.5


NASDAQ
NASDAQ
NASDAQ
NASDAQ
NASDAQ


AUTO CURTAIN
HOME ENTERTAINMENT
COMMUNICATION
MOODS
HOME AUTOMATION
GARAGE
AUTO SPRINKLERS
AUTO LIGHTING
HOME BACK UP SYSTEM
SECURITY
eCAM

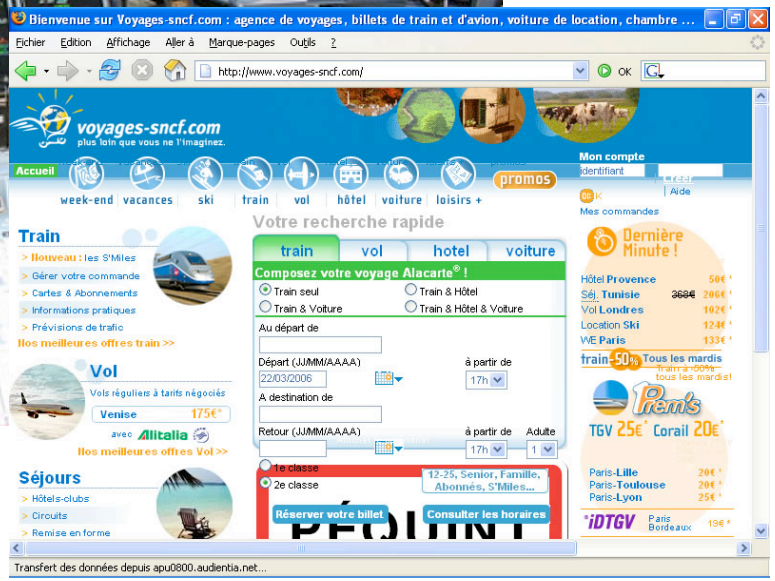

Bienvenue sur Voyages-sncf.com : agence de voyages, billets de train et d'avion, voiture de location, chambre ...
Fichier Edition Affichage Aller à Marque-pages Outils ?
http://www.voyages-sncf.com/
voyages-sncf.com
plus loin que vous ne l'imaginez.
Accueil
week-end vacances ski train vol hôtel voiture loisirs +
promos
Mon compte
Mes commandes
Train
Votre recherche rapide
train vol hotel voiture
Composez votre voyage Alacarte® !
Train seul
Train & Hôtel
Train & Voiture
Train & Hôtel & Voiture
Au départ de
Départ (JJ/MM/AAAA)
à partir de
A destination de
Retour (JJ/MM/AAAA)
Adulte
Vol
Vols réguliers à tarifs négociés
Venise 175€
avec Alitalia
Nos meilleures offres Vol >>
Séjours
Hôtels-clubs
Circuits
Remise en forme
1e classe
2e classe
12-25, Senior, Famille, Abonnés, S'Miles...
Réserver votre billet
Consulter les horaires
Dernière Minute !
Hôtel Provence 50€
Séj. Tunisie 200€
Vol Londres 102€
Location Ski 124€
WE Paris 133€
train -50% Tous les mardis
Proms
TGV 25€ Corail 20€
Paris-Lille 20€
Paris-Toulouse 20€
Paris-Lyon 25€
iDTGV Paris Bordeaux 19€
Transfert des données depuis apu0800.audientia.net...




WEB 2.0 Landscape
WEB APPLICATION
Widget/ component
Aggregation/ recombination
CONTENT SHARING
RECOMMENDATIONS/ FILTERING
Rating/ Tagging
Collaborative filtering
SOCIAL NETWORK
FEUTURE EXPLORATION NETWORK
www.futureexploration.net


NASDAQ


AUTO CURTAIN
HOME ENTERTAINMENT
COMMUNICATION
MOODS
HOME AUTOMATION
GARAGE
AUTO SPRINKLERS
AUTO LIGHTING
HOME BACK UP SYSTEM
SECURITY
eCAM

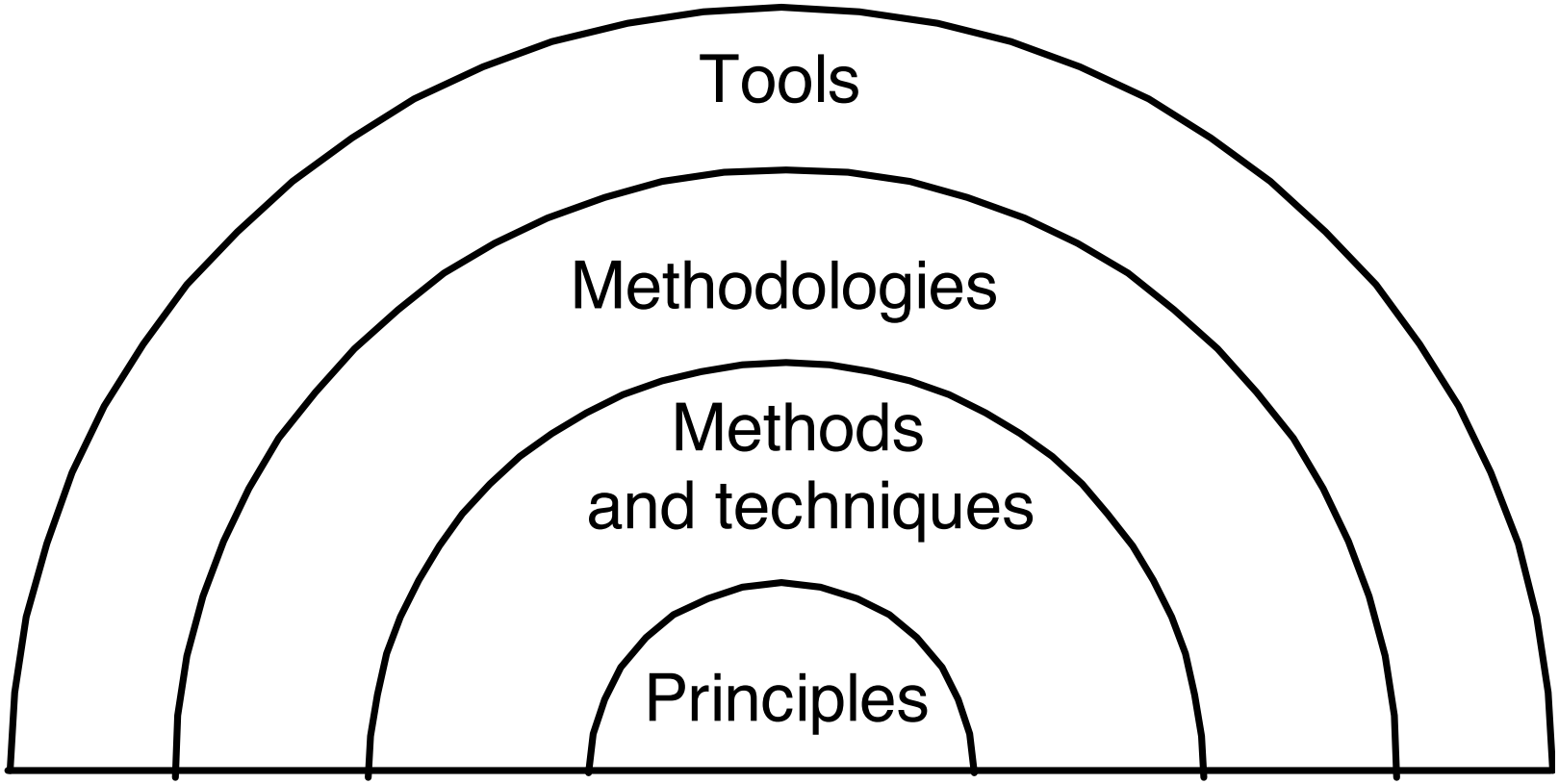

Bienvenue sur Voyages-sncf.com : agence de voyages, billets de train et d'avion, voiture de location, chambre ...
Fichier Edition Affichage Aller à Marque-pages Outils ?
http://www.voyages-sncf.com/
voyages-sncf.com
plus loin que vous ne l'imaginez.
Accueil
week-end vacances ski train vol hôtel voiture loisirs +
promos
Mon compte
Mes commandes
Votre recherche rapide
train vol hotel voiture
Composez votre voyage Alacarte® !
Train seul
Train & Hôtel
Train & Voiture
Train & Hôtel & Voiture
Au départ de
Départ (JJ/MM/AAAA)
22/03/2006
à partir de
17h
A destination de
Retour (JJ/MM/AAAA)
Adulte
1e classe
2e classe
12-25, Senior, Famille, Abonnés, S'Miles...
Réserver votre billet
Consulter les horaires
Train
Nouveau : les S'Miles
Gérer votre commande
Cartes & Abonnements
Informations pratiques
Prévisions de trafic
Les meilleures offres train >>
Vol
Vols réguliers à tarifs négociés
Venise 175€
avec Alitalia
Les meilleures offres Vol >>
Séjours
Hôtels-clubs
Circuits
Remise en forme
Dernière Minute !
Hôtel Provence 50€
Séj. Tunisie 200€
Vol Londres 102€
Location Ski 124€
WE Paris 133€
train -50% Tous les mardis
Proms
TGV 25€ Corail 20€
Paris-Lille 20€
Paris-Toulouse 20€
Paris-Lyon 25€
iDTGV Paris Bordeaux 19€
Transfert des données depuis apu0800.audientia.net...

3.

The identification of core principles underlying this apparent seething landscape is a major challenge for research in software engineering.

A visual representation

Carlo Ghezzi, Mehdi Jazayeri, and Dino Mandrioli.
Fundamentals of Software Engineering, 2nd edition. 2002.



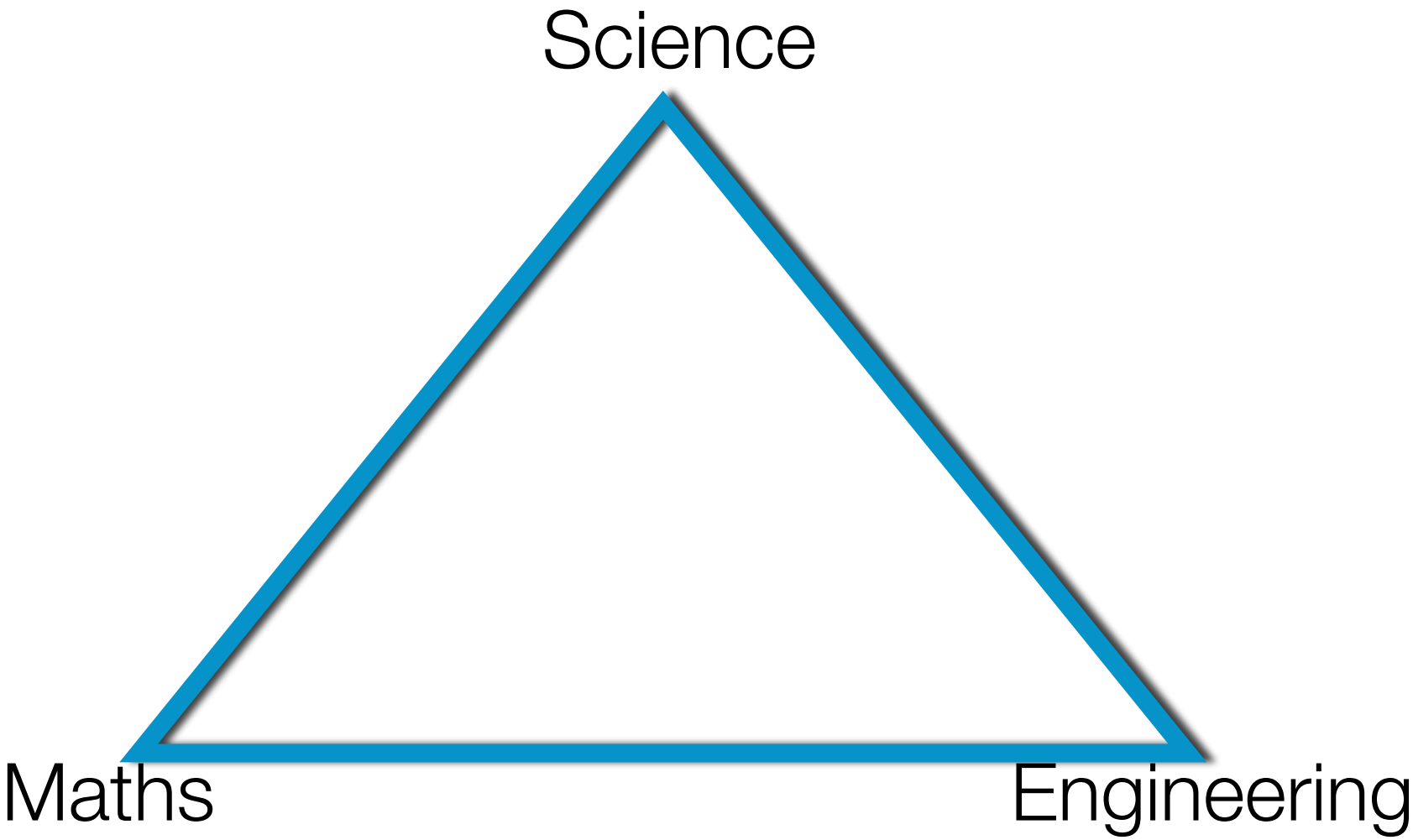
Key principles

Carlo Ghezzi, Mehdi Jazayeri, and Dino Mandrioli.
Fundamentals of Software Engineering, 2nd edition. 2002.

- Rigor and formality
- Separation of concerns
- Modularity
- Abstraction
- Anticipation of change
- Generality
- Incrementality

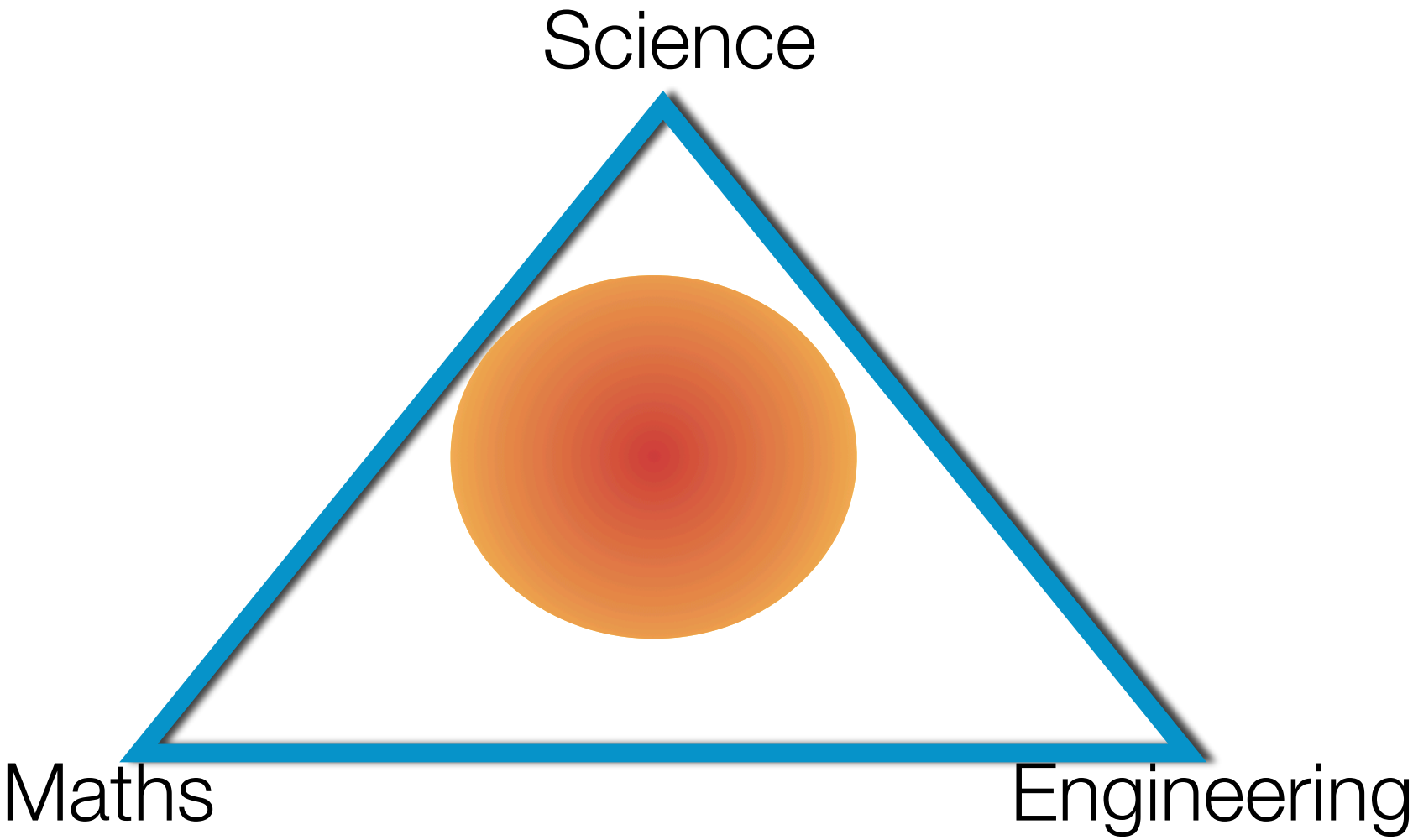
Research in Software Engineering

Peter J. Denning: Is computer science science?. *Communications of the ACM, 2005.* ***48****(4): p. 27 – 31.*



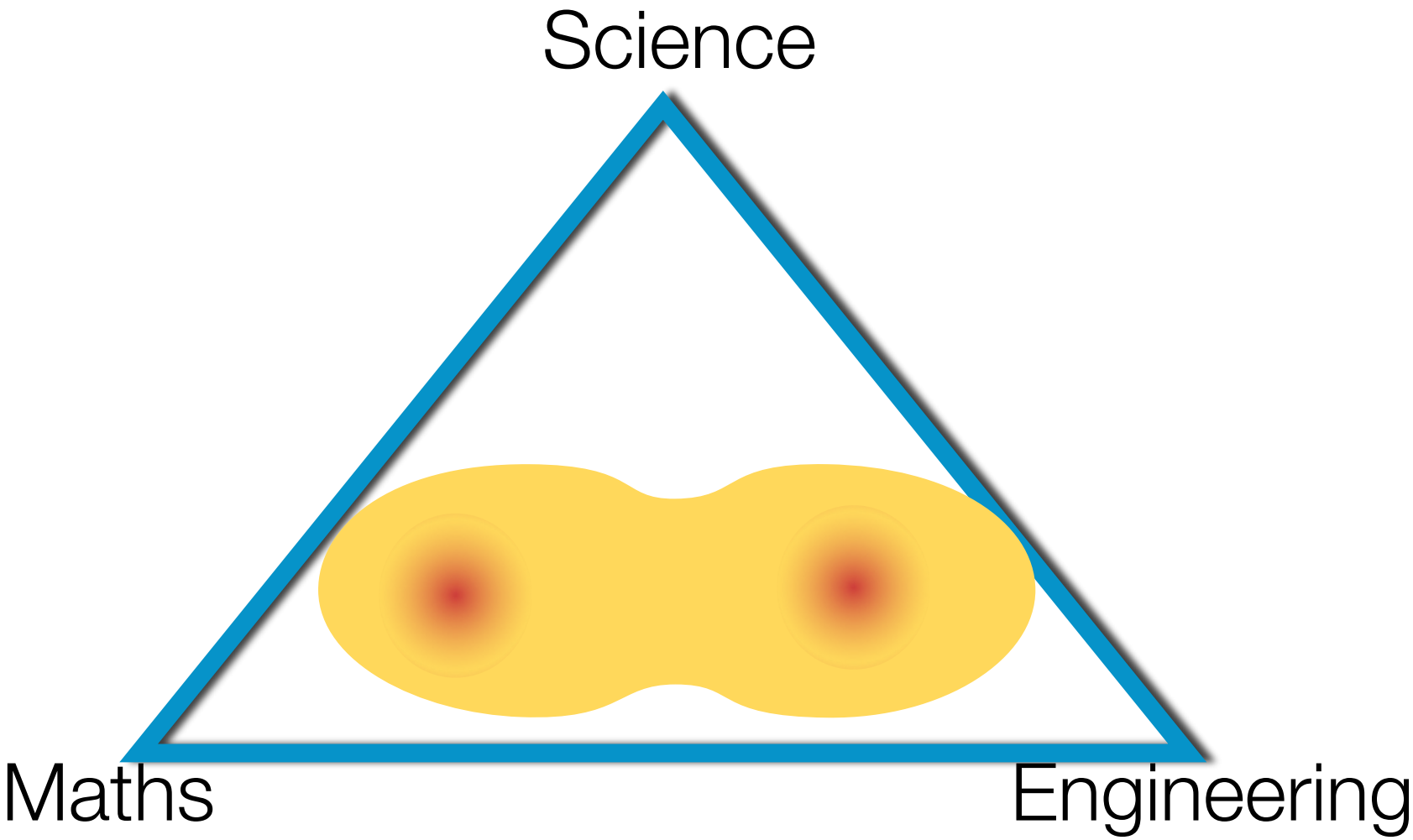
Research in Software Engineering

Peter J. Denning: Is computer science science?. *Communications of the ACM, 2005.* ***48****(4): p. 27 – 31.*



Research in Software Engineering

Peter J. Denning: Is computer science science?. *Communications of the ACM, 2005.* ***48****(4): p. 27 – 31.*



4.

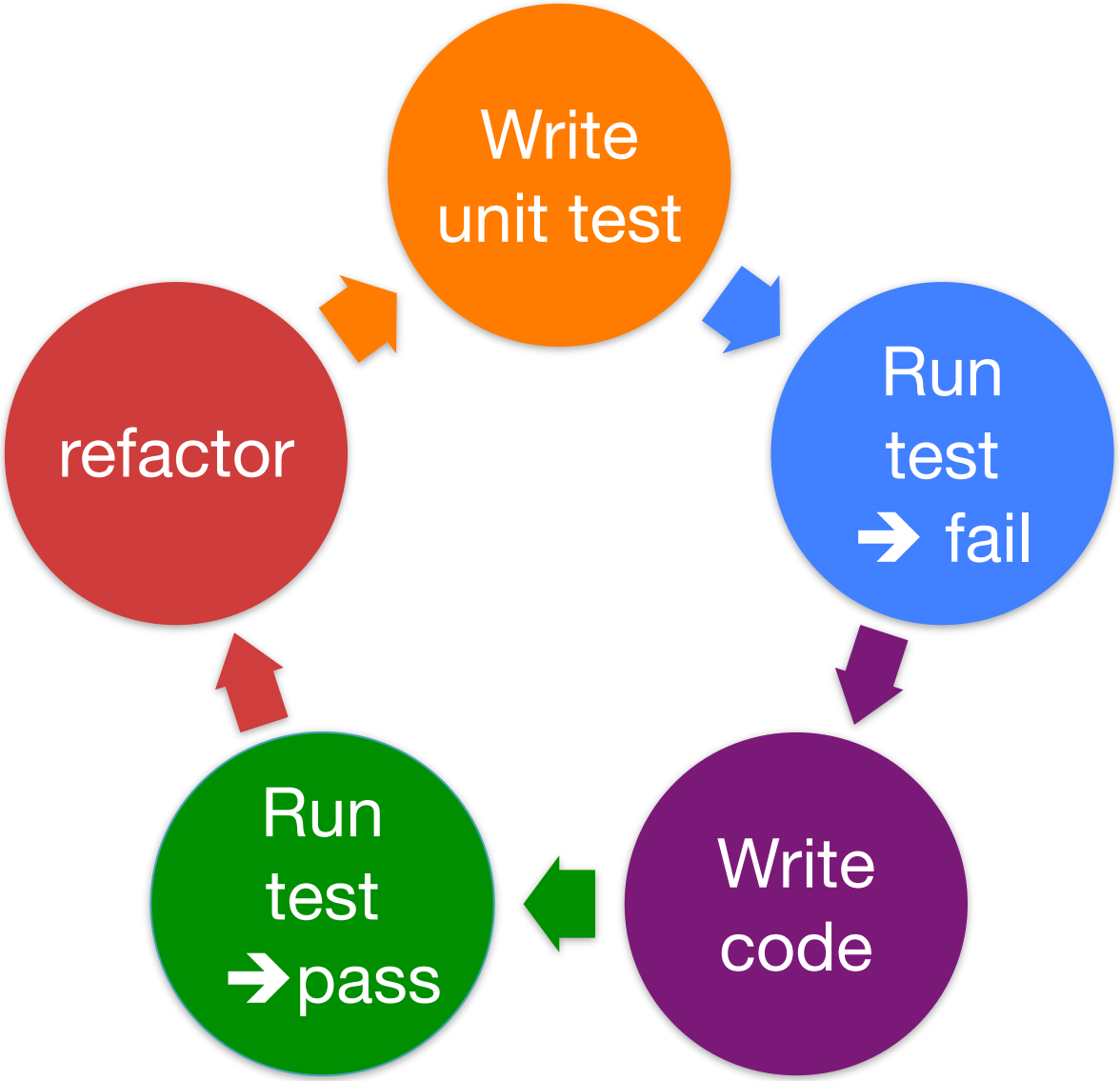
The confrontation of these core principles with the broadening scope of software leads to the emergence of new software construction paradigms

Software construction paradigms

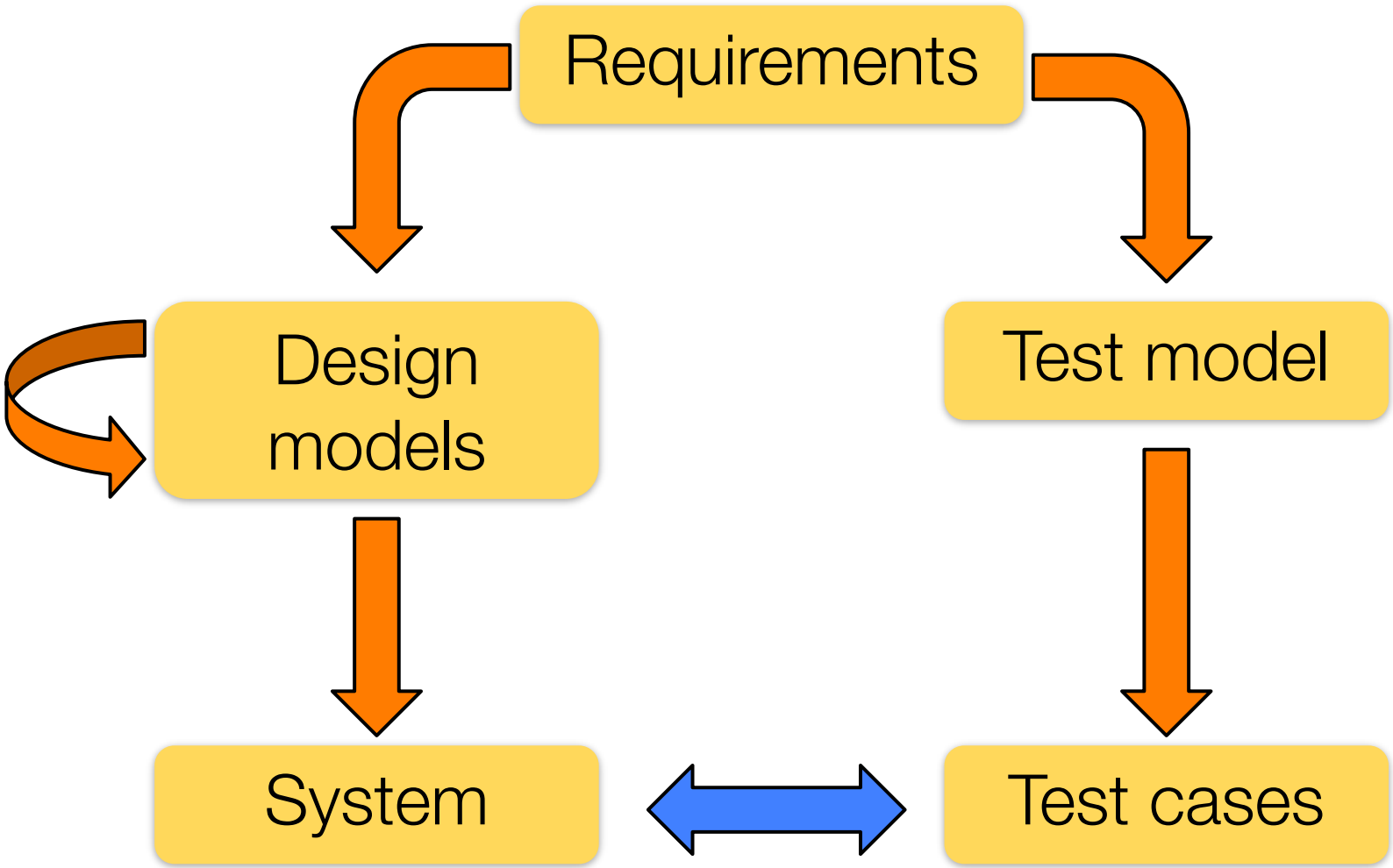
- Object-oriented programming
- Aspect-oriented programming
- Model-driven development
- Aspect-oriented modelling
- Multi-paradigm modelling

Software testing research for emerging paradigms extends beyond error detection to analyze and include the context imposed by these paradigms

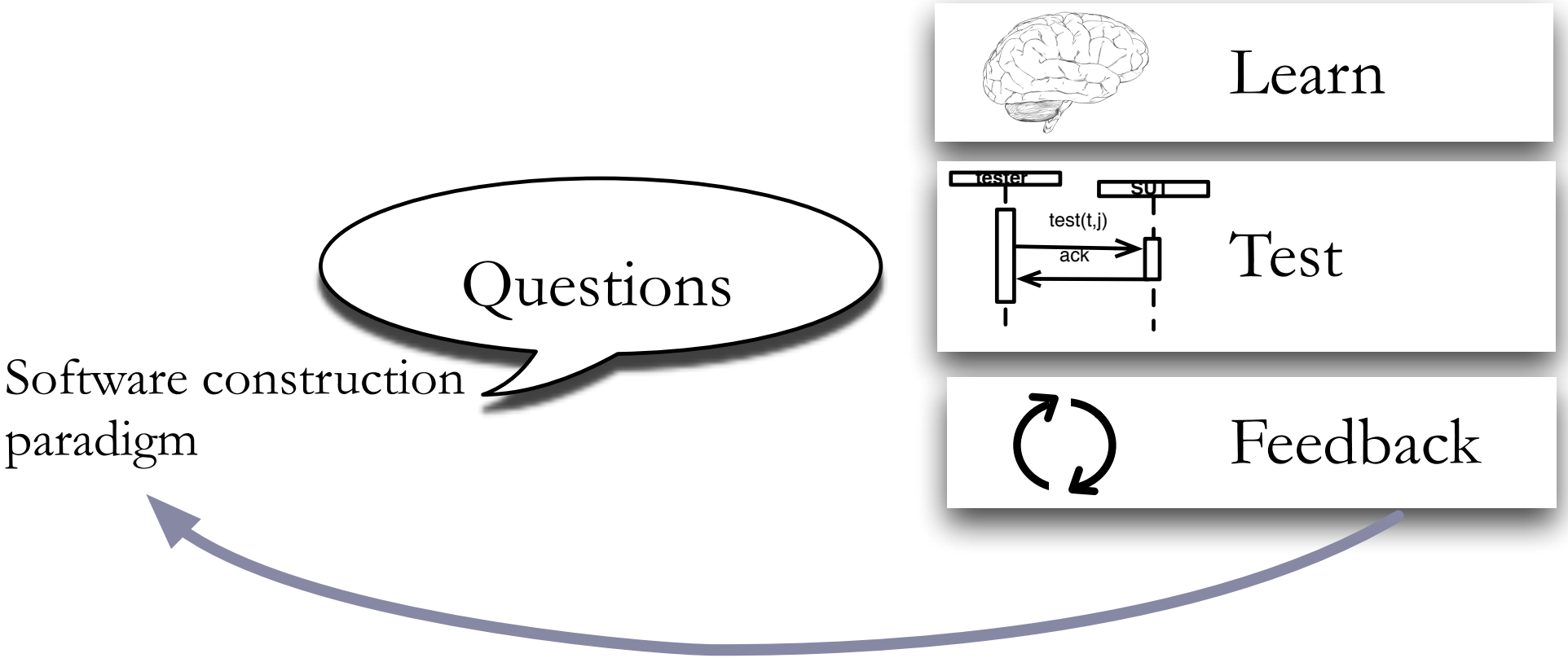
Agility and Test Driven Development



MDD and MBT

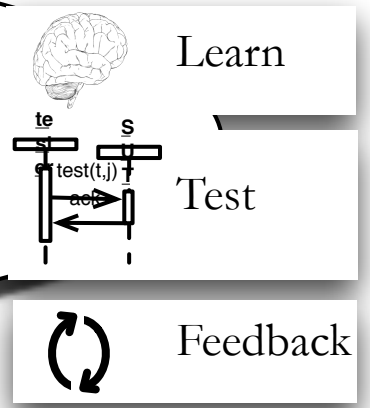


QLTF pattern



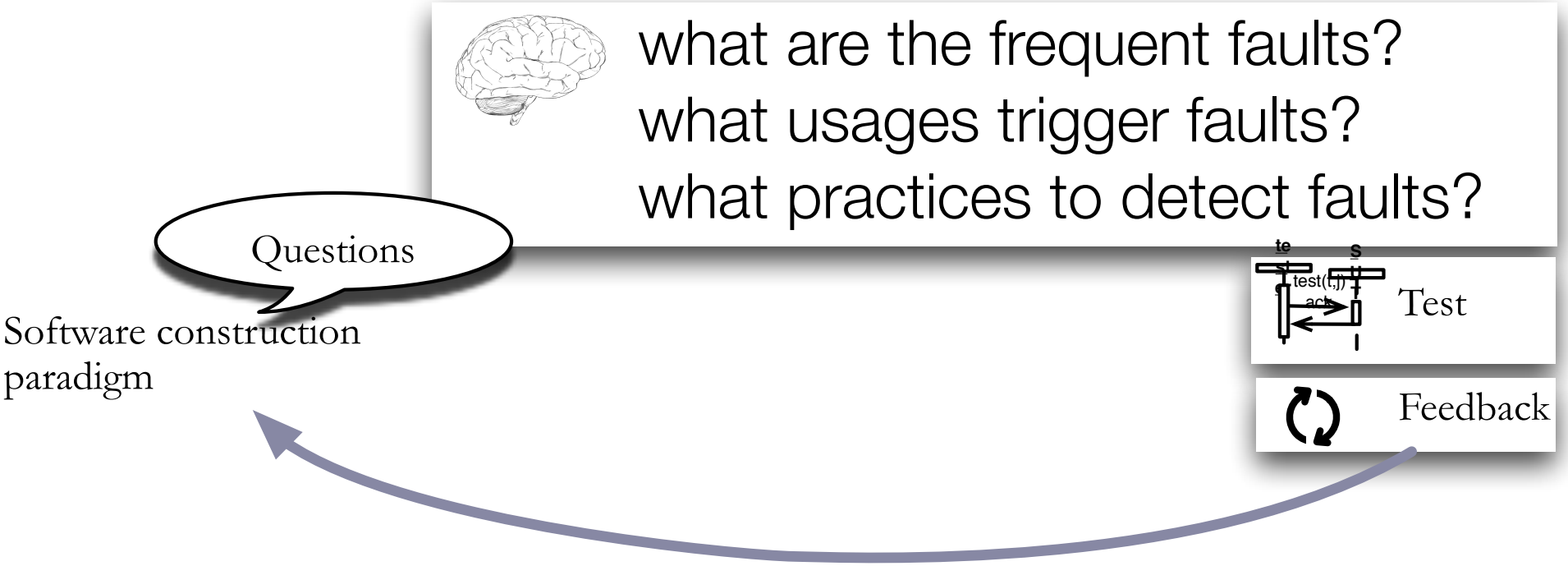
Mutation in QLTF

what can go wrong?
what type of errors are we looking for?
what can mitigate/increase error risks?

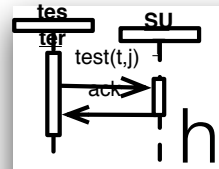


Software construction paradigm

Mutation in QLTF



Mutation in QLTF



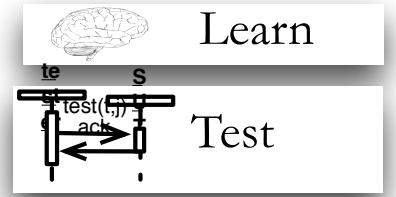
how can we assist fault detection?
do our techniques detect faults?
are new adequacy criteria adequate?

Questions

Software construction paradigm

Feedback

Mutation in QLTF

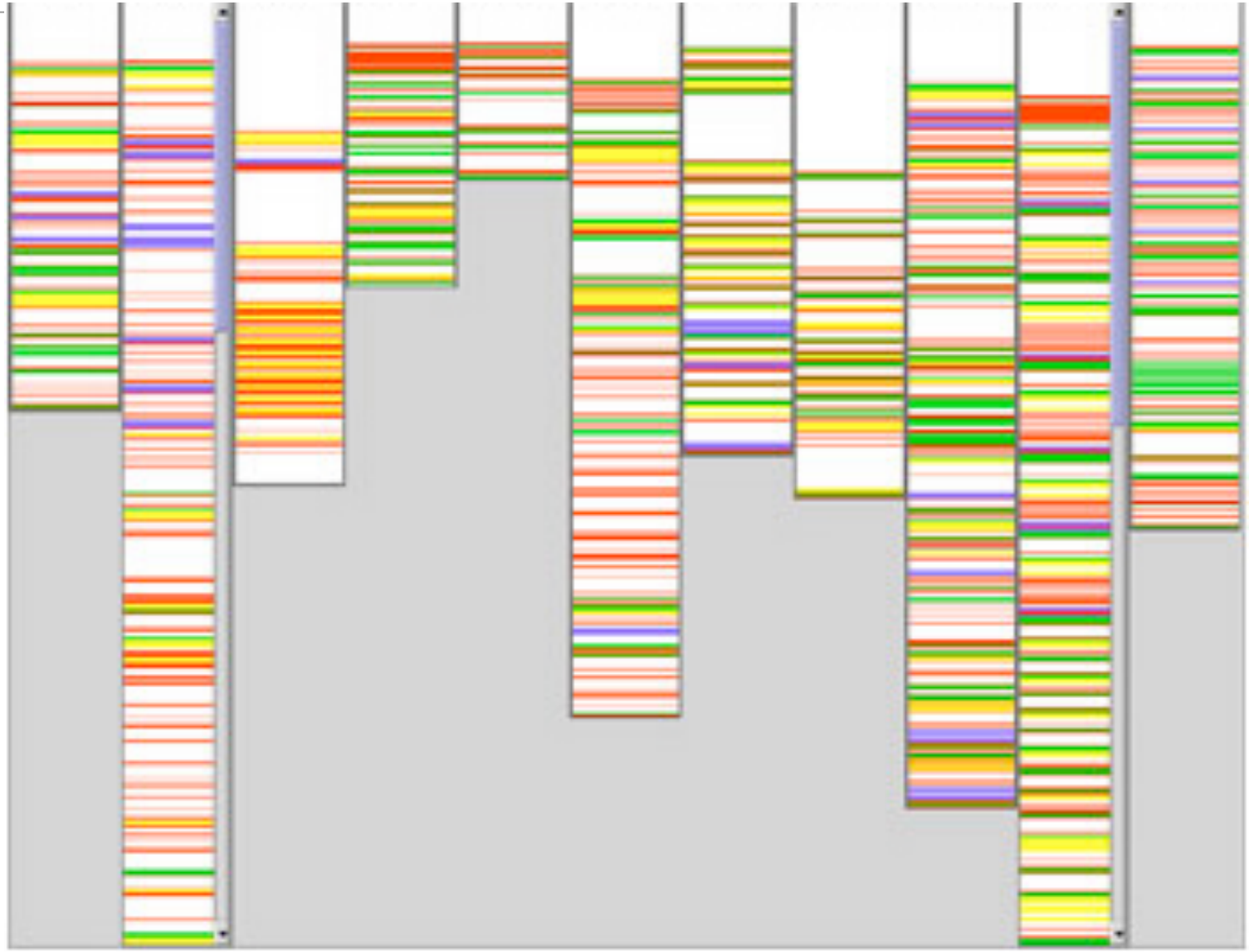


Questions

how can we prevent faults?
how can testability be improved?
how can test be integrated in dev. processes?

Software construction paradigm

Aspect oriented programming



An example: AOP

```
class Point implements FigureElement {
  private int x = 0, y = 0;

   int getX() { return x; }
   int getY() { return y; }
   void setX(int x) {
        this.x = x;
   }
   void setY(int y) {
        this.y = y;
   }
   void moveBy(int dx, int dy) {
        ...
   }
}
 class Line implements FigureElement{
    private Point p1, p2;

     Point getP1() { return p1; }
     Point getP2() { return p2; }
     void setP1(Point p1) {
          this.p1 = p1;
     }
     void setP2(Point p2) {
          this.p2 = p2;
     }
     void moveBy(int dx, int dy) {
          ...
     }
 }
```

```
aspect DisplayUpdating {
    pointcut move():
         call(void FigureElement.moveBy(int, int))||
         call(void Line.setP1(Point))             ||
         call(void Line.setP2(Point))             ||
         call(void Point.setX(int))               ||
         call(void Point.setY(int));
    after(): move() {
         Display.update();
    }
}
```

An example: AOP

```
class Point implements FigureElement {
  private int x = 0, y = 0;

   int getX() { return x; }
   int getY() { return y; }
   void setX(int x) {
        this.x = x;
   }
   void setY(int y) {
        this.y = y;
   }
   void moveBy(int dx, int dy) {
        ...
   }
}
 class Line implements FigureElement{
    private Point p1, p2;

     Point getP1() { return p1; }
     Point getP2() { return p2; }
     void setP1(Point p1) {
          this.p1 = p1;
     }
     void setP2(Point p2) {
          this.p2 = p2;
     }
     void moveBy(int dx, int dy) {
          ...
     }
 }
```

```
aspect DisplayUpdating {
    pointcut move():
         call(void FigureElement.moveBy(int, int))||
         call(void Line.setP1(Point))             ||
         call(void Line.setP2(Point))             ||
         call(void Point.setX(int))               ||
         call(void Point.setY(int));
    after(): move() {
         Display.update();
    }
}
```

Advice

An example: AOP

```
class Point implements FigureElement {
  private int x = 0, y = 0;

   int getX() { return x; }
   int getY() { return y; }
   void setX(int x) {
        this.x = x;
   }
   void setY(int y) {
        this.y = y;
   }
   void moveBy(int dx, int dy) {
        ...
   }
}

 class Line implements FigureElement{
    private Point p1, p2;

     Point getP1() { return p1; }
     Point getP2() { return p2; }
     void setP1(Point p1) {
          this.p1 = p1;
     }
     void setP2(Point p2) {
          this.p2 = p2;
     }
     void moveBy(int dx, int dy) {
          ...
     }
 }
```

Pointcut descriptor (PCD)

```
aspect DisplayUpdating {
    pointcut move():
        call(void FigureElement.moveBy(int, int))||
        call(void Line.setP1(Point))             ||
        call(void Line.setP2(Point))             ||
        call(void Point.setX(int))               ||
        call(void Point.setY(int));
    after(): move() {
        Display.update();
    }
}
```

Advice

An example: AOP

```
class Point implements FigureElement {
  private int x = 0, y = 0;

   int getX() { return x; }
   int getY() { return y; }
   void setX(int x) {
        this.x = x;
   }
   void setY(int y) {
        this.y = y;
   }
   void moveBy(int dx, int dy) {
        ...
   }
}

 class Line implements FigureElement{
    private Point p1, p2;

     Point getP1() { return p1; }
     Point getP2() { return p2; }
     void setP1(Point p1) {
          this.p1 = p1;
     }
     void setP2(Point p2) {
          this.p2 = p2;
     }
     void moveBy(int dx, int dy) {
          ...
     }
 }
```

JoinPoint

Pointcut descriptor (PCD)

```
aspect DisplayUpdating {
    pointcut move():
        call(void FigureElement.moveBy(int, int))||
        call(void Line.setP1(Point))             ||
        call(void Line.setP2(Point))             ||
        call(void Point.setX(int))               ||
        call(void Point.setY(int));
    after(): move() {
        Display.update();
    }
}
```

Advice

An example: AOP

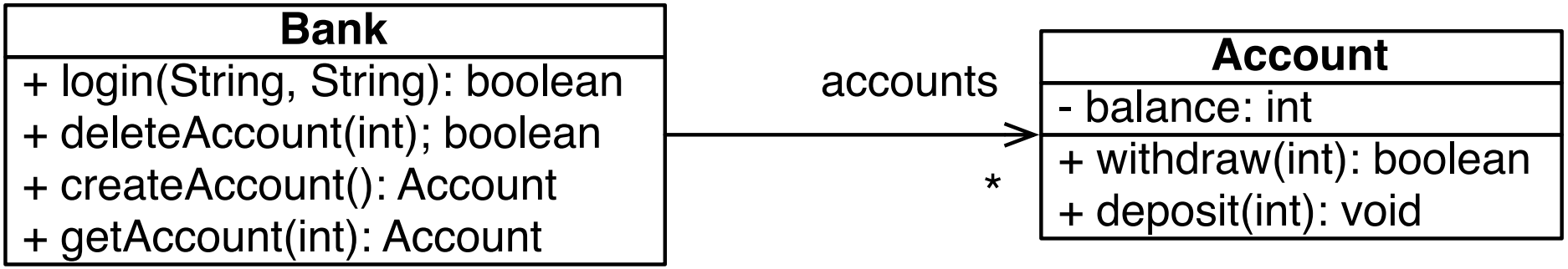
```
class Point implements FigureElement {
  private int x = 0, y = 0;

   int getX() { return x; }
   int getY() { return y; }
   void setX(int x) {
        this.x = x;
   }
   void setY(int y) {
        this.y = y;
   }
   void moveBy(int dx, int dy) {
        ...
   }
}

 class Line implements FigureElement{
    private Point p1, p2;

     Point getP1() { return p1; }
     Point getP2() { return p2; }
     void setP1(Point p1) {
          this.p1 = p1;
     }
     void setP2(Point p2) {
          this.p2 = p2;
     }
     void moveBy(int dx, int dy) {
          ...
     }
 }
```

JoinPoint

Pointcut descriptor (PCD)

```
aspect DisplayUpdating {
    pointcut move():
        call(void FigureElement.moveBy(int, int))||
        call(void Line.setP1(Point))             ||
        call(void Line.setP2(Point))             ||
        call(void Point.setX(int))               ||
        call(void Point.setY(int));
    after(): move() {
        Display.update();
    }
}
```

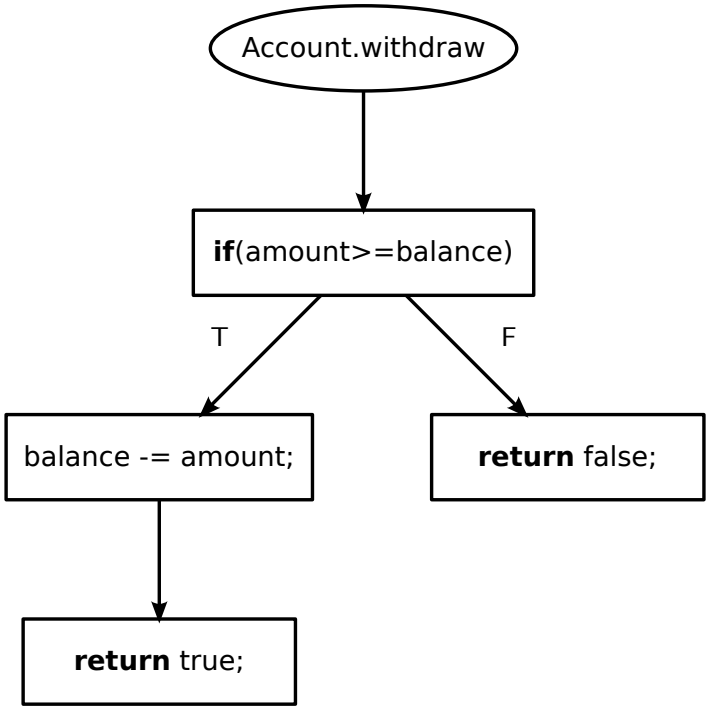
Advice



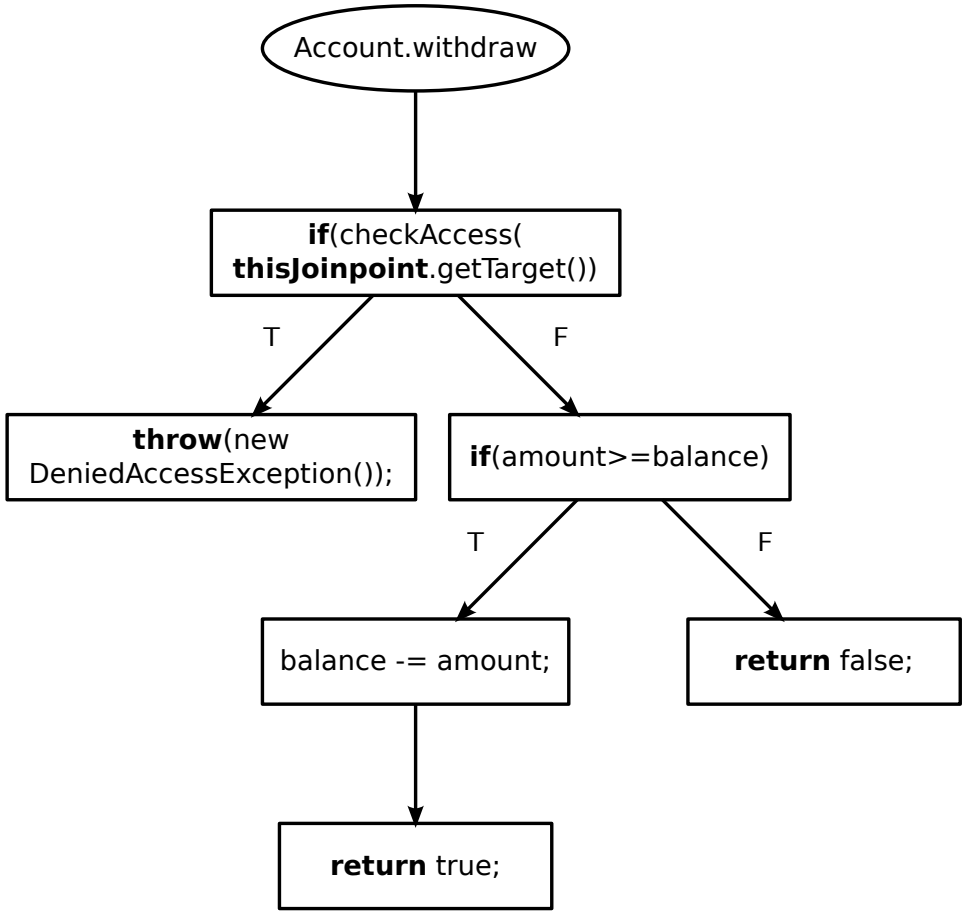

```
public aspect AccessControl {
    pointcut controlledAccess():
      execution(* Account.*(int));

    @AdviceName("AccessControl")
    before(): controlledAccess() {
        if(!checkAccess(thisJoinPoint.getTarget()))
            throw new DeniedAccessException();
    }
}
```

- A fault in the PCD can have a ripple effect and result in many different faults

Withdraw

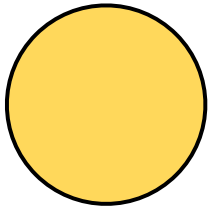

Account.withdraw
if(amount>=balance)
T
F
balance -= amount;
return false;
return true;

(a)


Account.withdraw
if(checkAccess(
thisJoinpoint.getTarget())
T
F
throw(new
DeniedAccessException());
if(amount>=balance)
T
F
balance -= amount;
return false;
return true;

(b)

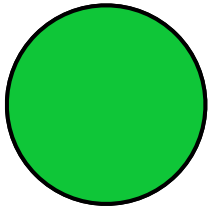
Faults in PCD



Intended

Matched

Faults in PCD

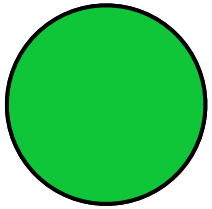


correct PCD

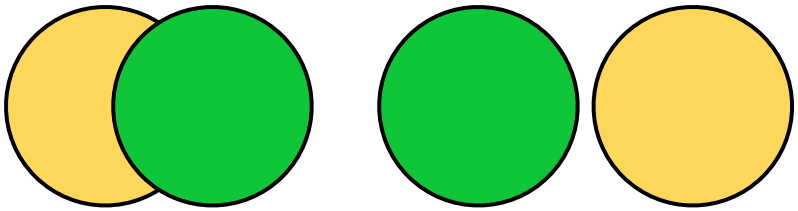
Intended

Matched

Faults in PCD



correct PCD

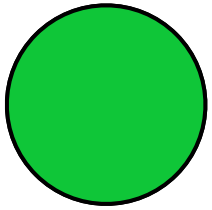


both neglected and unintended

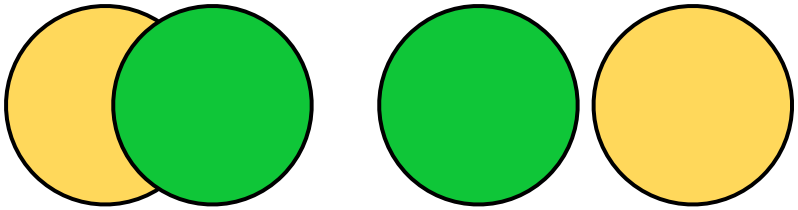
Intended

Matched

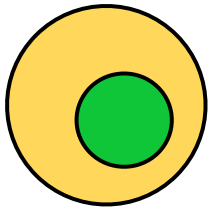
Faults in PCD



correct PCD



both neglected and unintended

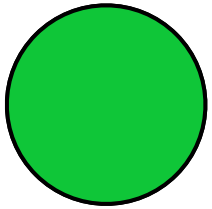


neglected joinpoints

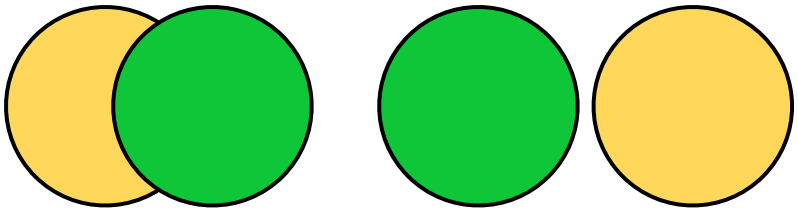
Intended

Matched

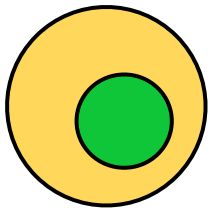
Faults in PCD



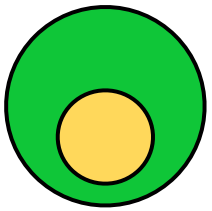
correct PCD



both neglected and unintended



neglected joinpoints



unintended joinpoints

- both neglected and unintended

```
pointcut controlledAccess(): execution(* Bank.*(int));
```

matches `deleteAccount()` and `getAccount()`

```
pointcut controlledAccess(): execution(boolean Bank.*(int));
```

matches `deleteAccount()` and `login()`

- neglected joinpoints

```
pointcut controlledAccess(): execution(* Account.*(int));
```

matches `withdraw()` and `deposit()`

```
pointcut controlledAccess(): execution(boolean Account.*(int));
```

matches `withdraw()`

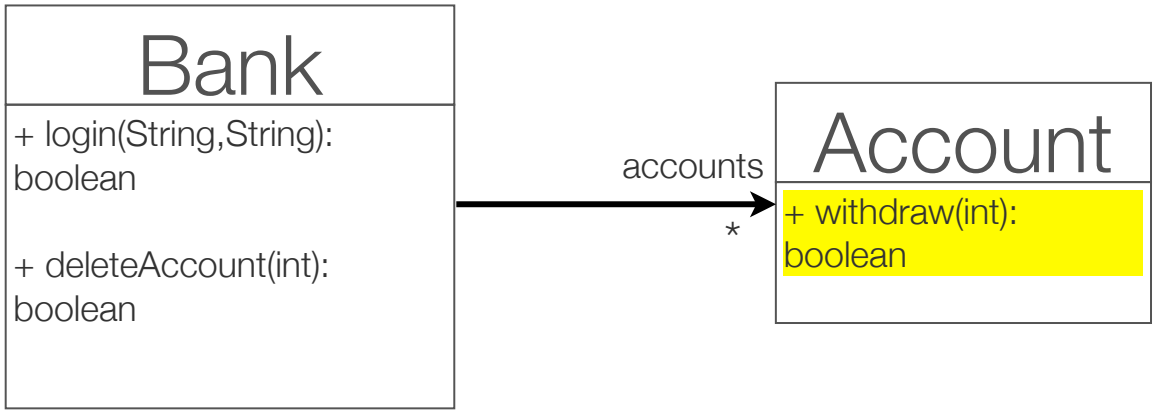
Mutant PCD

- A PCD where a fault has been inserted
  - Selects a different set of joinpoints
- Equivalent mutant
  - Mutant that matches the same set of joinpoint
  - Equivalent mutants can be detected statically

Mutation Operators

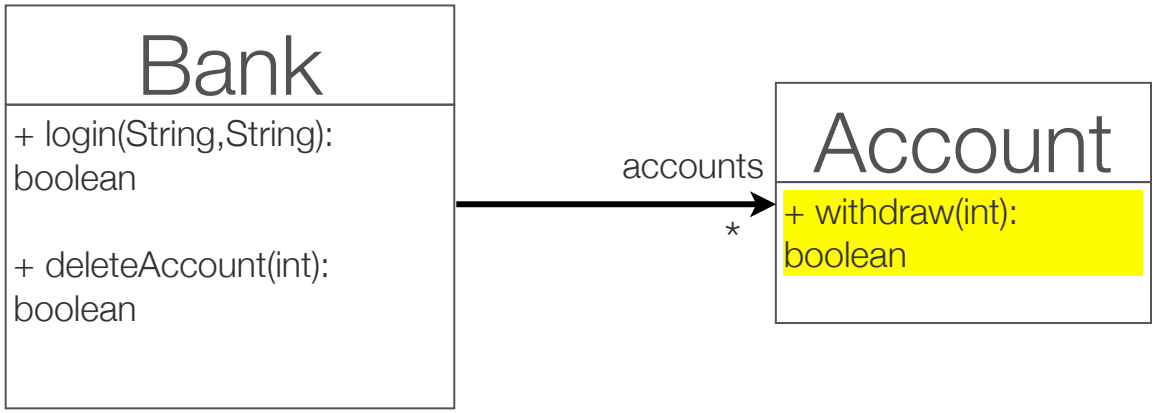
Operator	Description
PCCC	Replaces a `cflow` by a `cflowbelow`, or the contrary
PCCE	Replaces a `call` by an `execution`, or the contrary
PCGS	Replaces a `get` by a `set`, or the contrary
PCLO	Changes the logical operators in a composition of PCDs
PCTT	Replaces a `this` by a `target`, or the contrary
POEC	Adds, removes or changes throwing clauses
POPL	Changes the parameter list
PSWR	Removes wildcards
PWAR	Removes annotation from type, field or method patterns
PWIW	Adds wildcards

Testing the PCD: an example – 1



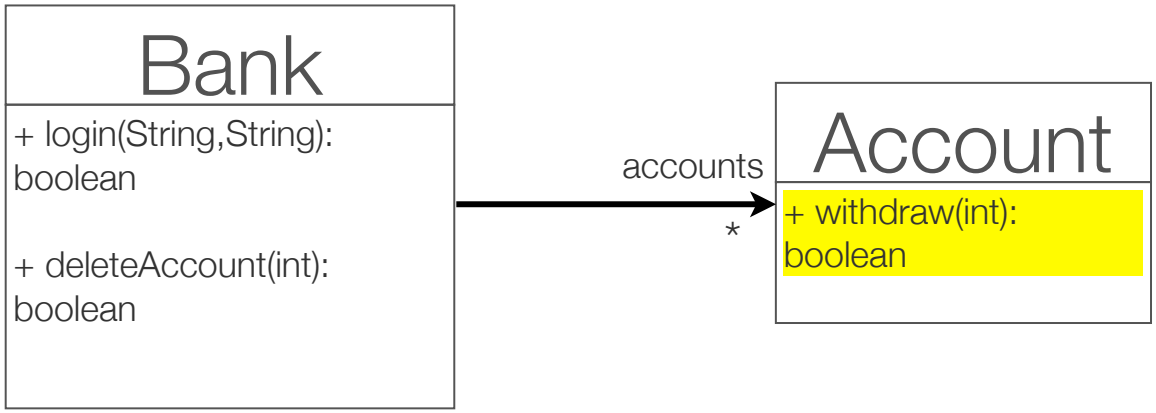

```
public aspect AccessControl {
    pointcut controlledAccess(): execution(* Account.*(int))

    @AdviceName("AccessControl")
    before(): controlledAccess() {
        if(!checkAccess(thisJoinPoint.getTarget()))
            throw new DeniedAccessException();
    }
}
```

Testing the PCD: an example – 1



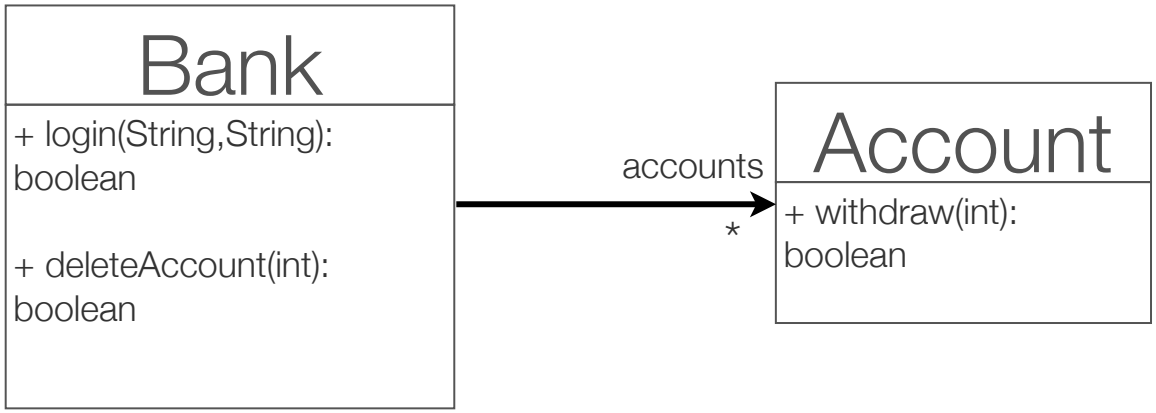

```
public void testAccessControlDelete() {
    bank.login(nonAuthorizedUser,password);
    try {
        account.withdraw(30);
        fail("Access should not be authorized");
    } catch(DeniedAccessException() {}
}
```

Testing the PCD: an example – 1



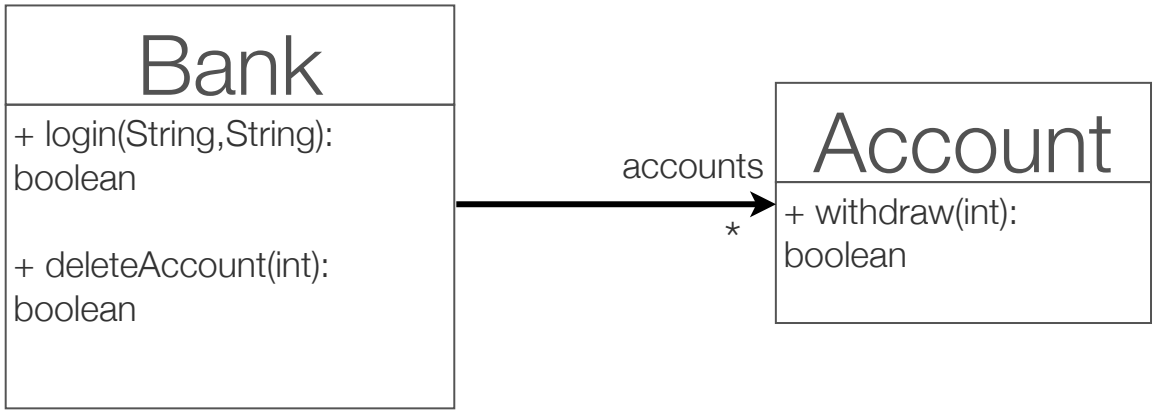

```
public void testAccessControlDelete() {
    bank.login(nonAuthorizedUser,password);
    try {
        account.withdraw(30);
        fail("Access should not be authorized");
    } catch(DeniedAccessException() {}
}
```

Testing the PCD: example – 2



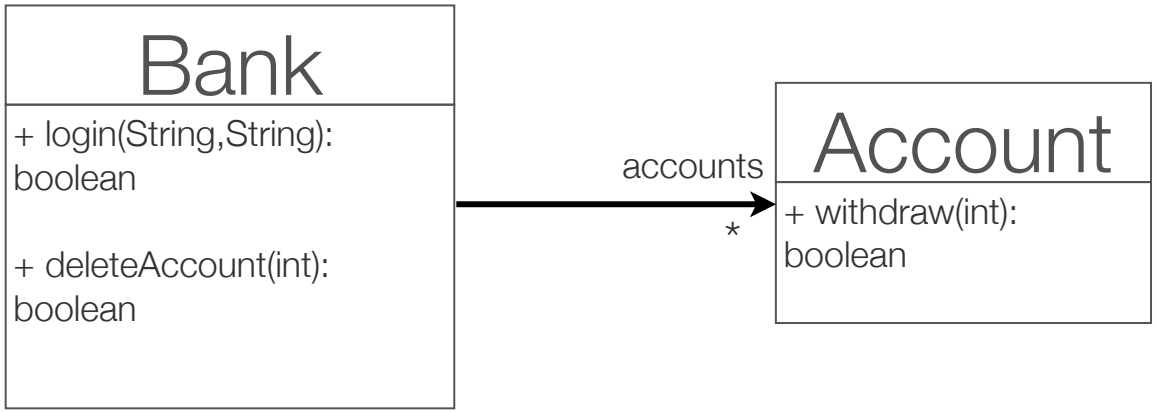

```
public aspect AccessControl {
    pointcut controlledAccess(): execution(void Account.*(int))

    @AdviceName("AccessControl")
    before(): controlledAccess() {
        if(!checkAccess(thisJoinPoint.getTarget()))
            throw new DeniedAccessException();
    }
}
```

Testing the PCD: example – 2



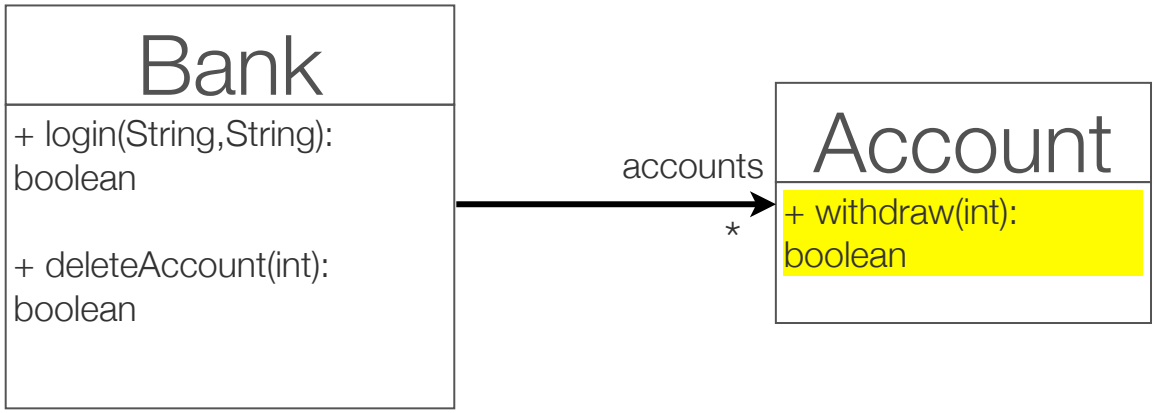

```
public void testAccessControlDelete() {
    bank.login(nonAuthorizedUser,password);
    try {
        account.withdraw(30);
        fail("Access should not be authorized");
    } catch(DeniedAccessException() {}
}
```

Testing the PCD: example – 2



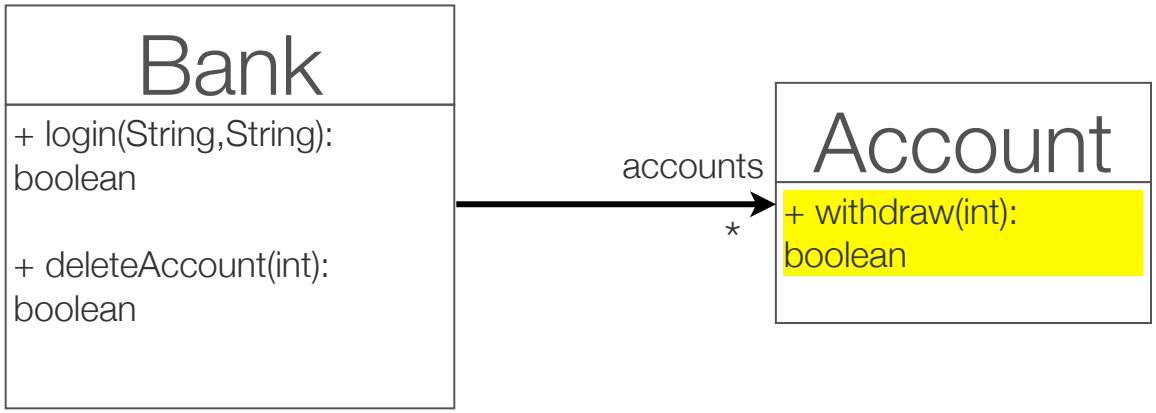

```
public void testAccessControlDelete() {
    bank.login(nonAuthorizedUser,password);
    try {
        account.withdraw(30);
        fail("Access should not be authorized");
    } catch(DeniedAccessException() {}
}
```

Testing the PCD: example – 3



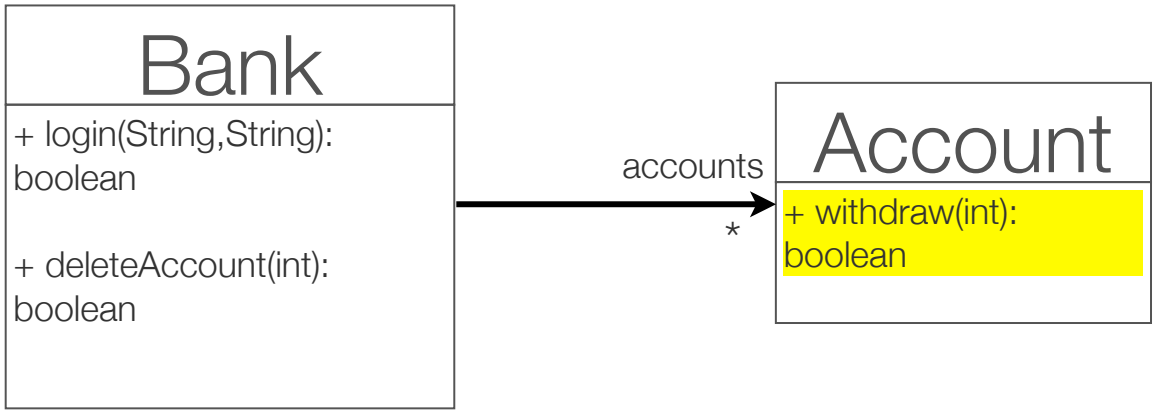

```
public aspect AccessControl {
    pointcut controlledAccess(): execution(* Account.*(int))

    @AdviceName("AccessControl")
    before(): controlledAccess() {
        if(checkAccess(thisJoinPoint.getTarget()))
            throw new DeniedAccessException();
    }
}
```

Testing the PCD: example – 3



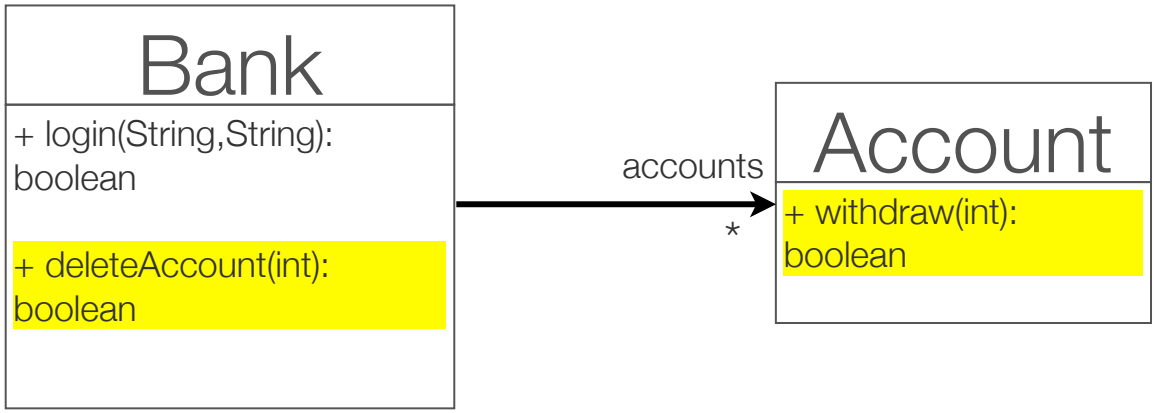

```
public void testAccessControlDelete() {
    bank.login(nonAuthorizedUser,password);
    try {
        account.withdraw(30);
        fail("Access should not be authorized");
    } catch(DeniedAccessException() {}
}
```

Testing the PCD: example – 3



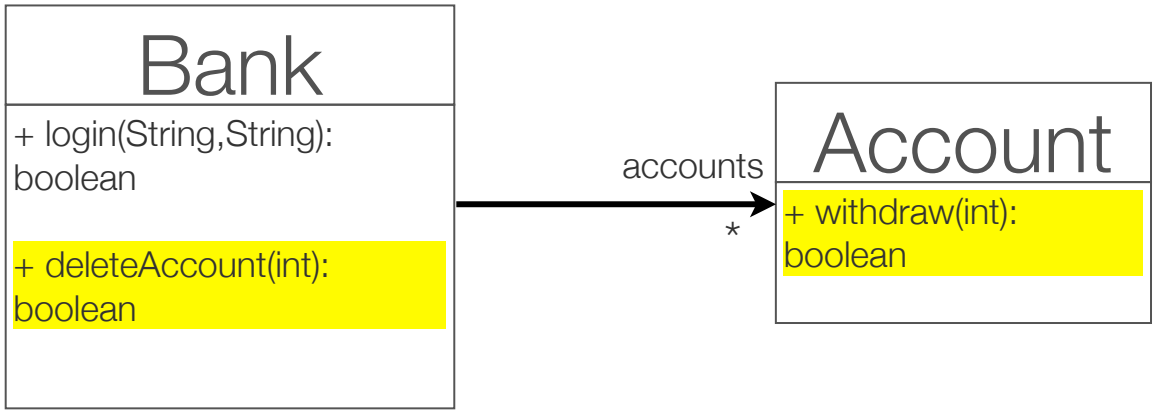

```
public void testAccessControlDelete() {
    bank.login(nonAuthorizedUser,password);
    try {
        account.withdraw(30);
        fail("Access should not be authorized");
    } catch(DeniedAccessException() {}
}
```

Testing the PCD: example – 4



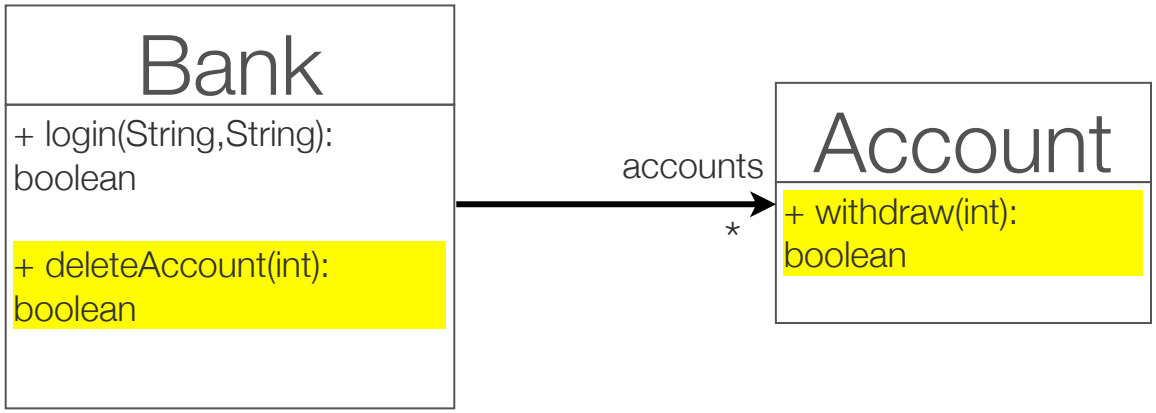

```
public aspect AccessControl {
    pointcut controlledAccess(): execution(* *(int))

    @AdviceName("AccessControl")
    before(): controlledAccess() {
        if(!checkAccess(thisJoinPoint.getTarget()))
            throw new DeniedAccessException();
    }
}
```

Testing the PCD: example – 4



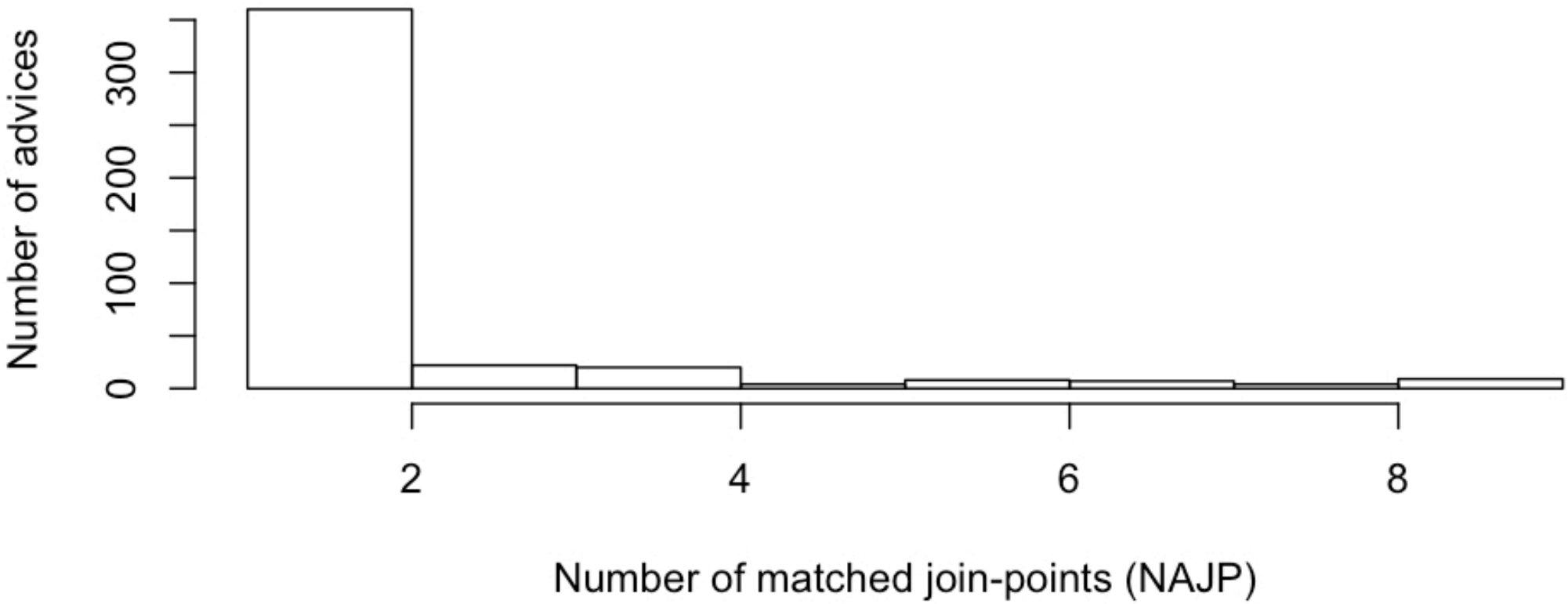

```
public void testAccessControlDelete() {
    bank.login(nonAuthorizedUser,password);
    try {
        account.withdraw(30);
        fail("Access should not be authorized");
    } catch(DeniedAccessException() {}
}
```

Testing the PCD: example – 4




```
public void testAccessControlDelete() {
    bank.login(nonAuthorizedUser,password);
    try {
        account.withdraw(30);
        fail("Access should not be authorized");
    } catch(DeniedAccessException() {}
}
```

Crosscutting aspects

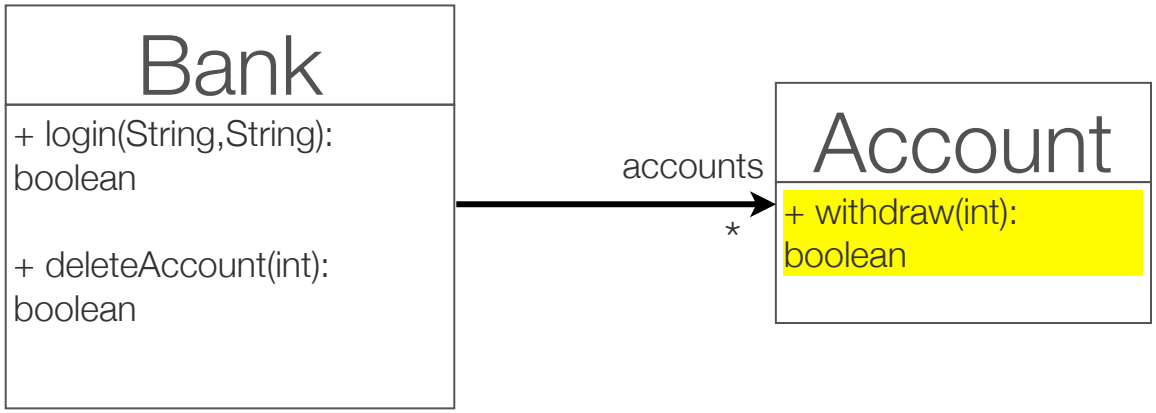


38 open source projects:
479 aspects, 522 advices in 21245 join-points

AdviceTracer: writing PCD specific Oracles

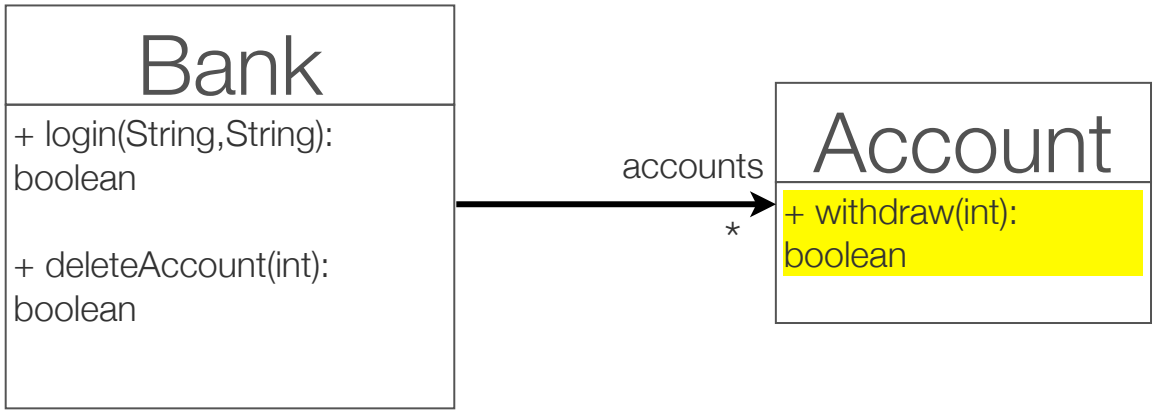
- AdviceTracer: AspectJ library to write oracles that specifically target the PCD
- Oracles written with AdviceTracer for one PCD are:
  - Independent from the advices
  - Independent from other PCDs

AdviceTracer: example – 1



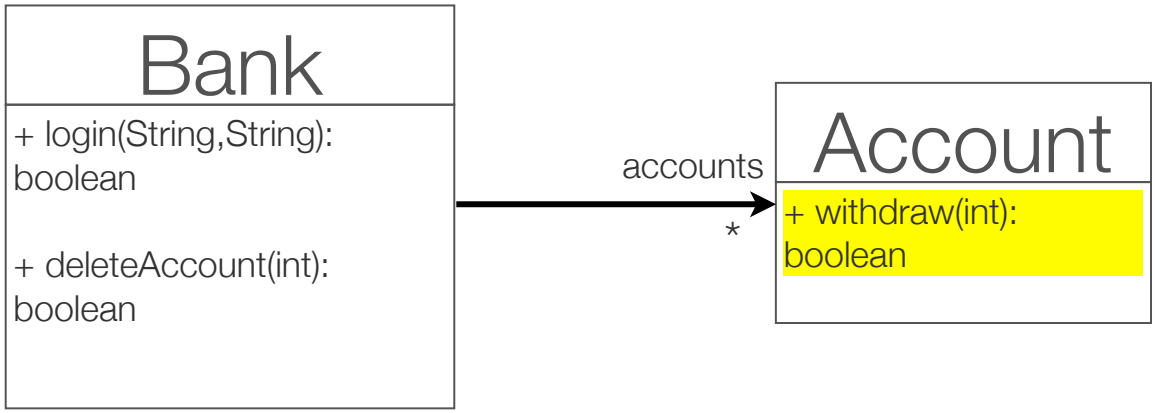

```
public aspect AccessControl {
    pointcut controlledAccess(): execution(* Account.*(int))

    @AdviceName("AccessControl")
    before(): controlledAccess() {
        if(!checkAccess(thisJoinPoint.getTarget()))
            throw new DeniedAccessException();
    }
}
```

AdviceTracer: example – 1



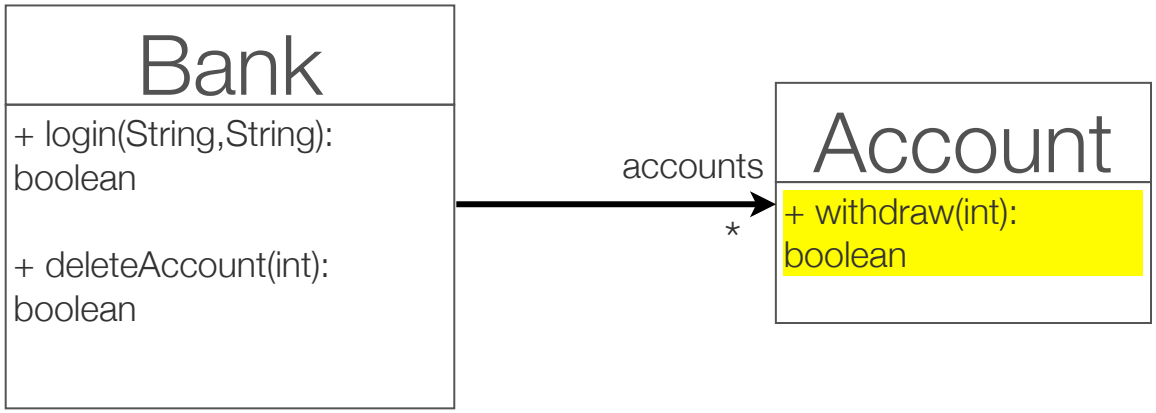

```
public void testAccessControlDelete() {
    bank.login(nonAuthorizedUser,password);
    addTracedAdvice("AccessControl");
    setAdviceTracerOn();
    try {
        account.withdraw(30);
    } catch(DeniedAccessException() {}
    setAdviceTracerOff();
    assertAdviceExecutionEquals(1);
    assertExecutedAdvice("AccessControl","Account.withdraw:47");
}
```

AdviceTracer: example – 1



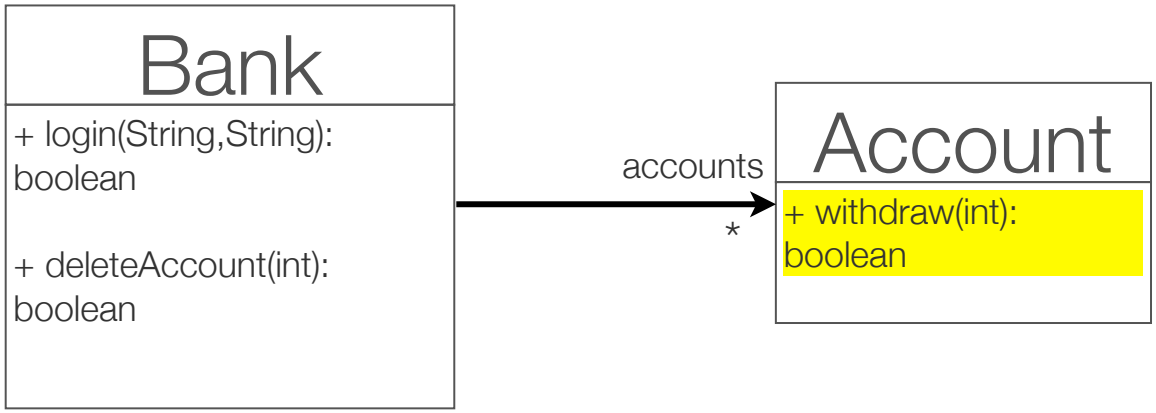

```
public void testAccessControlDelete() {
    bank.login(nonAuthorizedUser,password);
    addTracedAdvice("AccessControl");
    setAdviceTracerOn();
    try {
        account.withdraw(30);
    } catch(DeniedAccessException() {}
    setAdviceTracerOff();
    assertAdviceExecutionEquals(1);
    assertExecutedAdvice("AccessControl","Account.withdraw:47");
}
```

AdviceTracer: example – 1



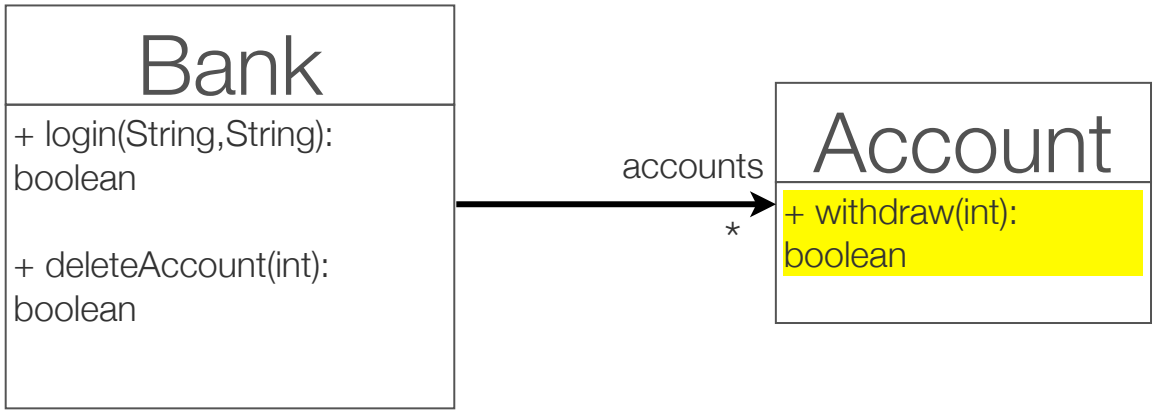

```
public void testAccessControlDelete() {
    bank.login(nonAuthorizedUser,password);
    addTracedAdvice("AccessControl");
    setAdviceTracerOn();
    try {
        account.withdraw(30);
    } catch(DeniedAccessException() {}
    setAdviceTracerOff();
    assertAdviceExecutionEquals(1);
    assertExecutedAdvice("AccessControl","Account.withdraw:47");
}
```

AdviceTracer: example – 1



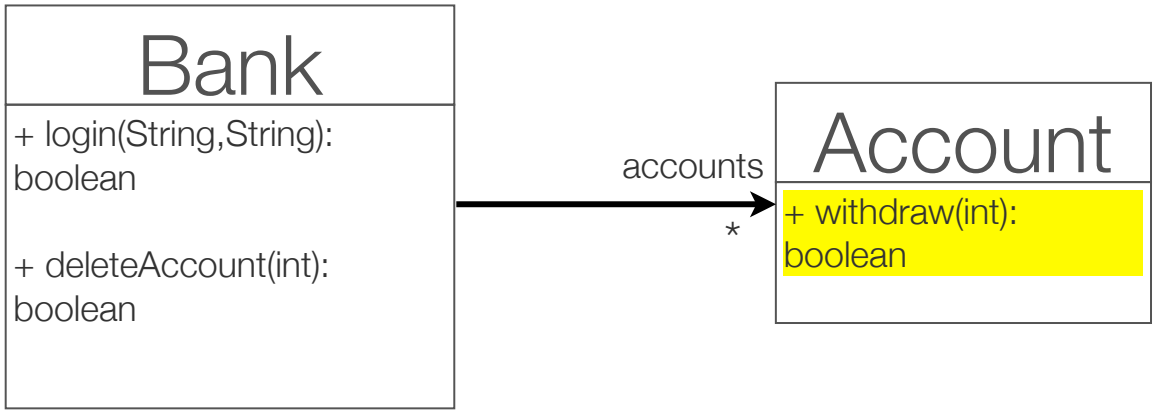

```
public void testAccessControlDelete() {
    bank.login(nonAuthorizedUser,password);
    addTracedAdvice("AccessControl");
    setAdviceTracerOn();
    try {
        account.withdraw(30);
    } catch(DeniedAccessException() {}
    setAdviceTracerOff();
    assertAdviceExecutionEquals(1);
    assertExecutedAdvice("AccessControl","Account.withdraw:47");
}
```

AdviceTracer: example – 1



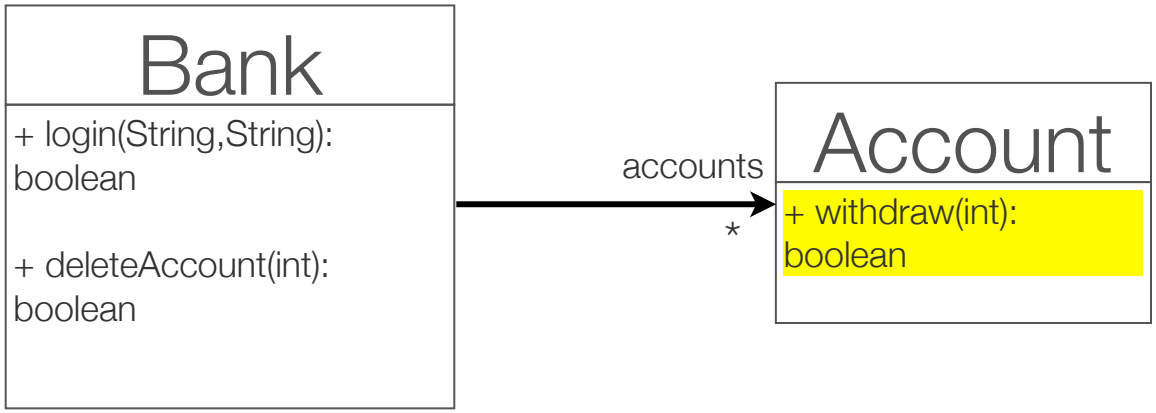

```
public void testAccessControlDelete() {
    bank.login(nonAuthorizedUser,password);
    addTracedAdvice("AccessControl");
    setAdviceTracerOn();
    try {
        account.withdraw(30);
    } catch(DeniedAccessException() {}
    setAdviceTracerOff();
    assertAdviceExecutionEquals(1);
    assertExecutedAdvice("AccessControl","Account.withdraw:47");
}
```

AdviceTracer: example – 1



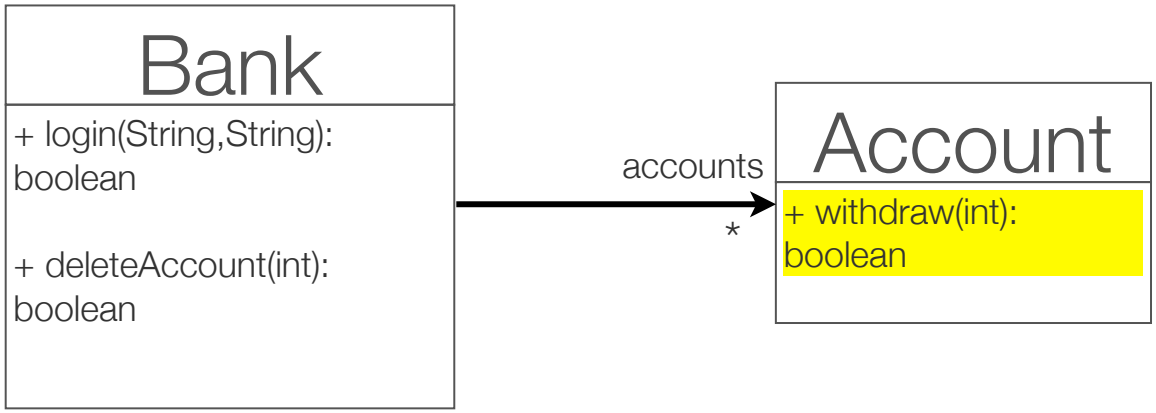

```
public void testAccessControlDelete() {
    bank.login(nonAuthorizedUser,password);
    addTracedAdvice("AccessControl");
    setAdviceTracerOn();
    try {
        account.withdraw(30);
    } catch(DeniedAccessException() {}
    setAdviceTracerOff();
    assertAdviceExecutionEquals(1);
    assertExecutedAdvice("AccessControl","Account.withdraw:47");
}
```

AdviceTracer: example – 1



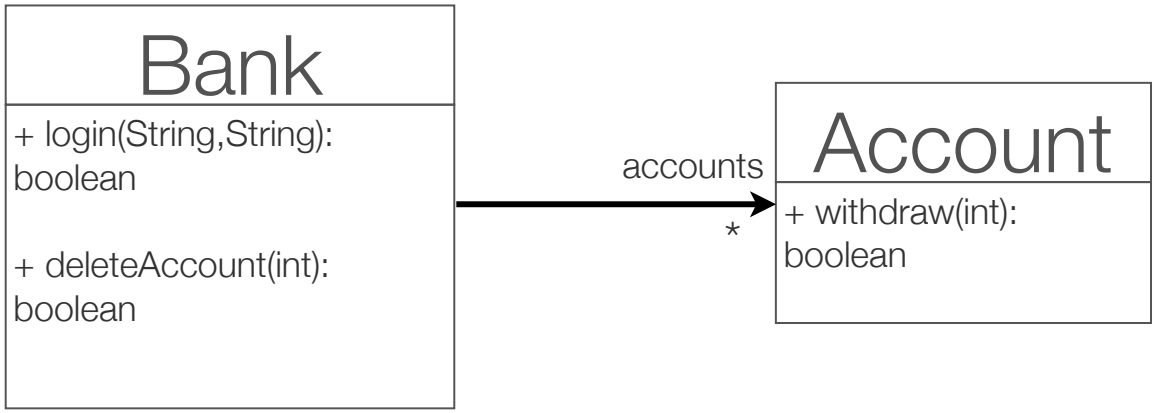

```
public void testAccessControlDelete() {
    bank.login(nonAuthorizedUser,password);
    addTracedAdvice("AccessControl");
    setAdviceTracerOn();
    try {
        account.withdraw(30);
    } catch(DeniedAccessException() {}
    setAdviceTracerOff();
    assertAdviceExecutionEquals(1);
    assertExecutedAdvice("AccessControl","Account.withdraw:47");
}
```

AdviceTracer: example – 1



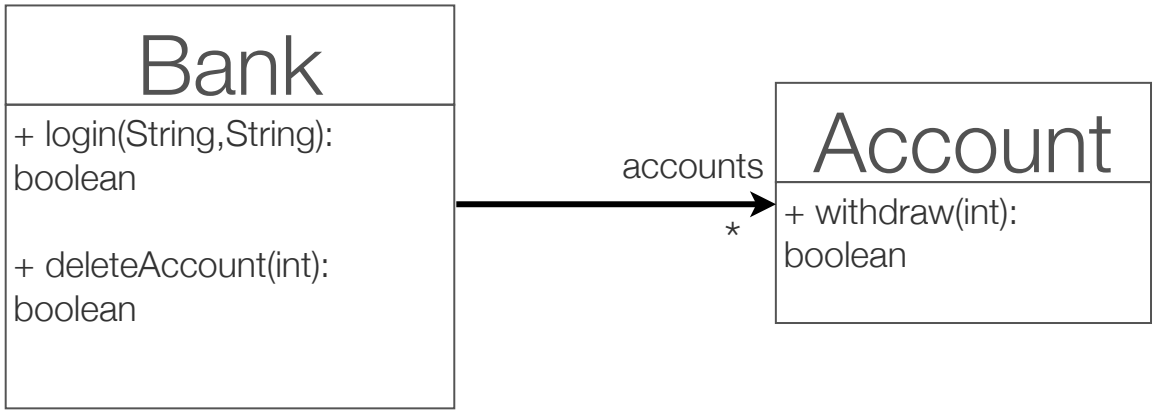

```
public void testAccessControlDelete() {
    bank.login(nonAuthorizedUser,password);
    addTracedAdvice("AccessControl");
    setAdviceTracerOn();
    try {
        account.withdraw(30);
    } catch(DeniedAccessException() {}
    setAdviceTracerOff();
    assertAdviceExecutionEquals(1);
    assertExecutedAdvice("AccessControl","Account.withdraw:47");
}
```

AdviceTracer: example – 2



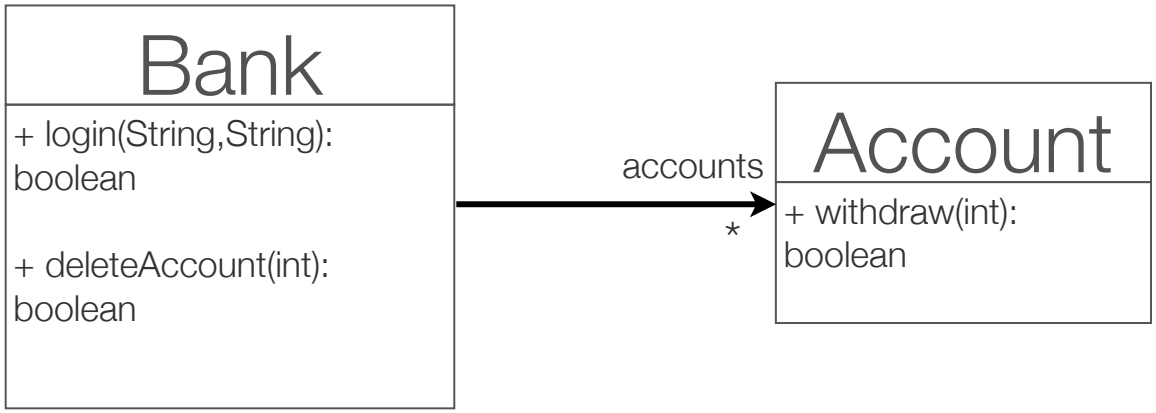

```
public aspect AccessControl {
    pointcut controlledAccess(): execution(void Account.*(int))

    @AdviceName("AccessControl")
    before(): controlledAccess() {
        if(!checkAccess(thisJoinPoint.getTarget()))
            throw new DeniedAccessException();
    }
}
```

AdviceTracer: example – 2



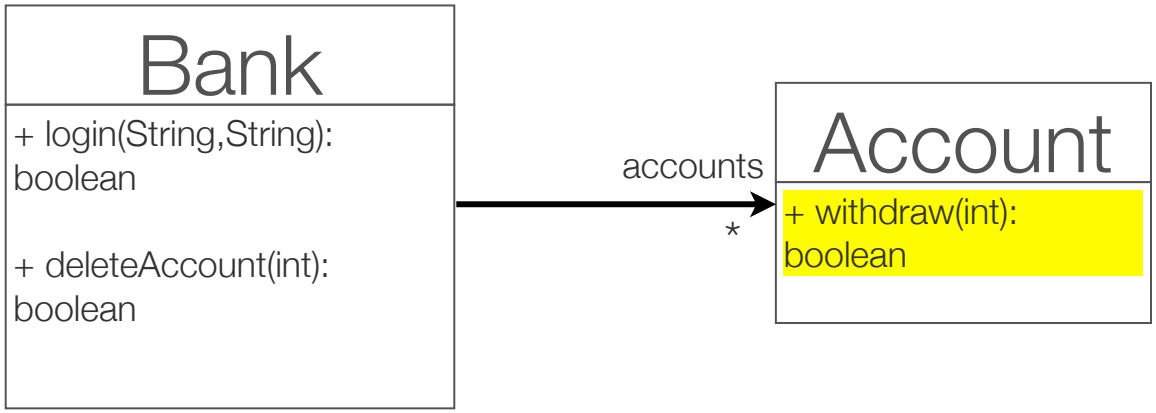

```
public void testAccessControlDelete() {
    bank.login(nonAuthorizedUser,password);
    addTracedAdvice("AccessControl");
    setAdviceTracerOn();
    try {
        account.withdraw(30);
    } catch(DeniedAccessException() {}
    setAdviceTracerOff();
    assertAdviceExecutionEquals(1);
    assertExecutedAdvice("AccessControl","Account.withdraw:47");
}
```

AdviceTracer: example – 2



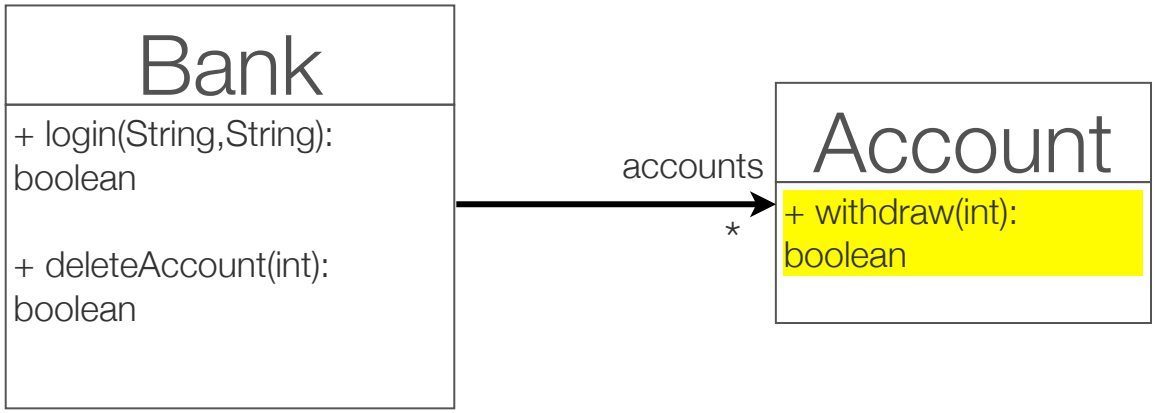

```
public void testAccessControlDelete() {
    bank.login(nonAuthorizedUser,password);
    addTracedAdvice("AccessControl");
    setAdviceTracerOn();
    try {
        account.withdraw(30);
    } catch(DeniedAccessException() {}
    setAdviceTracerOff();
    assertAdviceExecutionEquals(1);
    assertExecutedAdvice("AccessControl","Account.withdraw:47");
}
```

AdviceTracer: example – 3



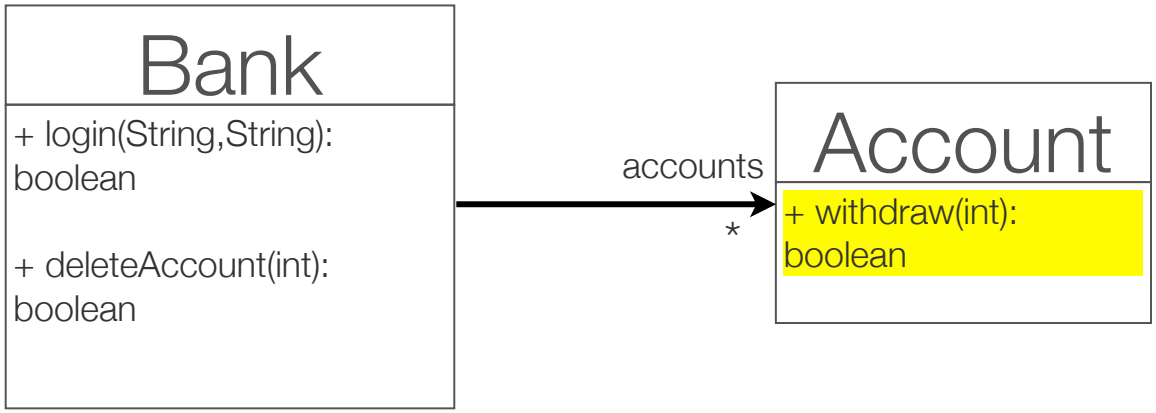

```
public aspect AccessControl {
    pointcut controlledAccess(): execution(* Account.*(int))

    @AdviceName("AccessControl")
    before(): controlledAccess() {
        if(checkAccess(thisJoinPoint.getTarget()))
            throw new DeniedAccessException();
    }
}
```

AdviceTracer: example – 3



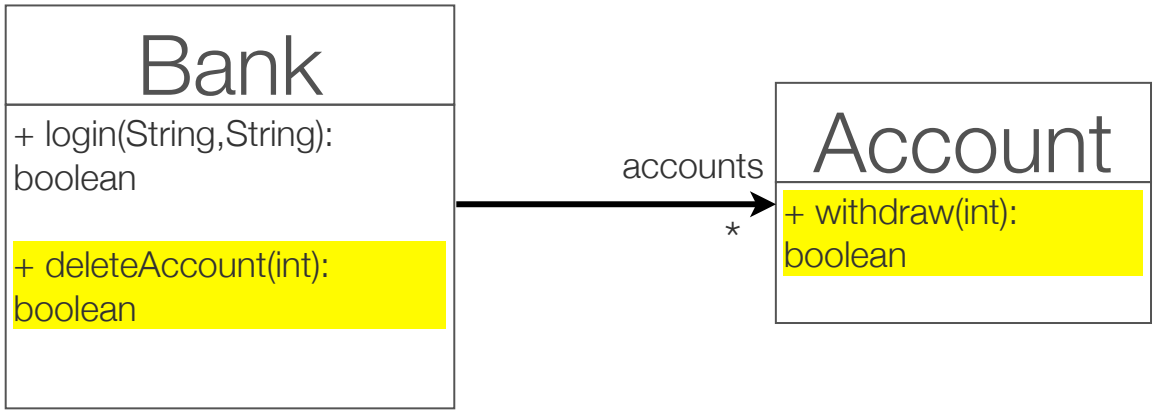

```
public void testAccessControlDelete() {
    bank.login(nonAuthorizedUser,password);
    addTracedAdvice("AccessControl");
    setAdviceTracerOn();
    try {
        account.withdraw(30);
    } catch(DeniedAccessException() {}
    setAdviceTracerOff();
    assertAdviceExecutionEquals(1);
    assertExecutedAdvice("AccessControl","Account.withdraw:47");
}
```

AdviceTracer: example – 3



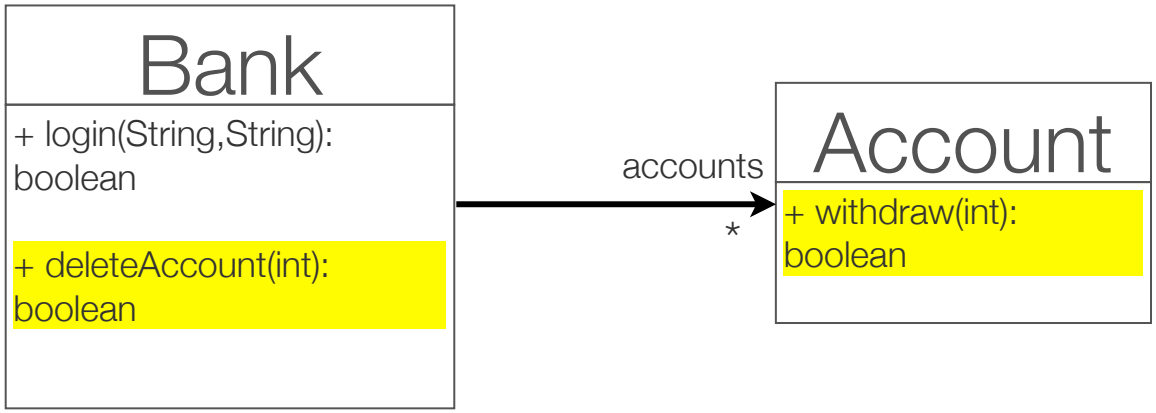

```
public void testAccessControlDelete() {
    bank.login(nonAuthorizedUser,password);
    addTracedAdvice("AccessControl");
    setAdviceTracerOn();
    try {
        account.withdraw(30);
    } catch(DeniedAccessException() {}
    setAdviceTracerOff();
    assertAdviceExecutionEquals(1);
    assertExecutedAdvice("AccessControl","Account.withdraw:47");
}
```

AdviceTracer: example – 4



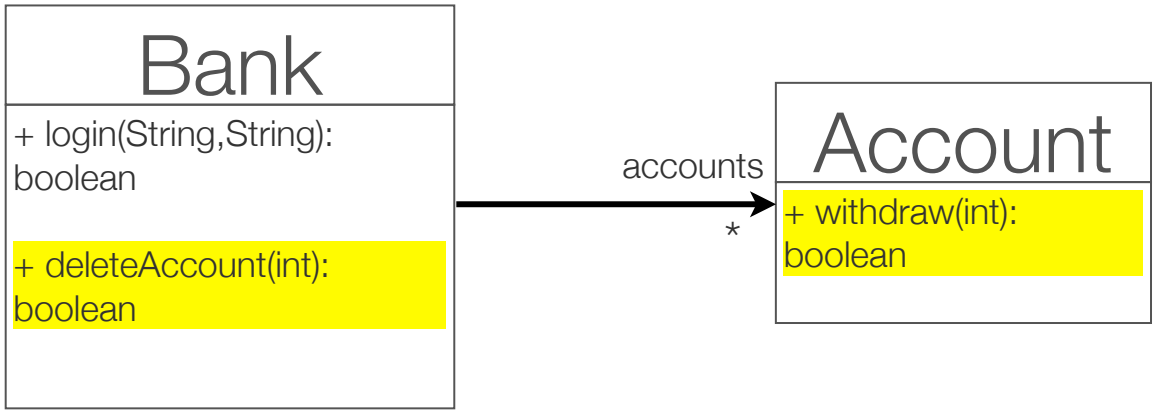

```
public aspect AccessControl {
    pointcut controlledAccess(): execution(* *(int))

    @AdviceName("AccessControl")
    before(): controlledAccess() {
        if(!checkAccess(thisJoinPoint.getTarget()))
            throw new DeniedAccessException();
    }
}
```

AdviceTracer: example – 4



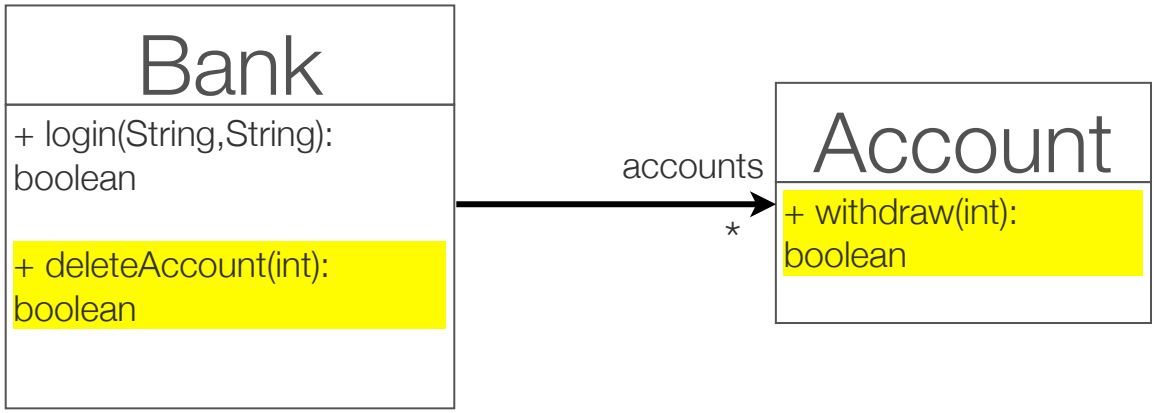

```
public void testAccessControlDelete() {
    bank.login(nonAuthorizedUser,password);
    addTracedAdvice("AccessControl");
    setAdviceTracerOn();
    try {
        account.withdraw(30);
    } catch(DeniedAccessException() {}
    setAdviceTracerOff();
    assertAdviceExecutionEquals(1);
    assertExecutedAdvice("AccessControl","Account.withdraw:47");
}
```

AdviceTracer: example – 4



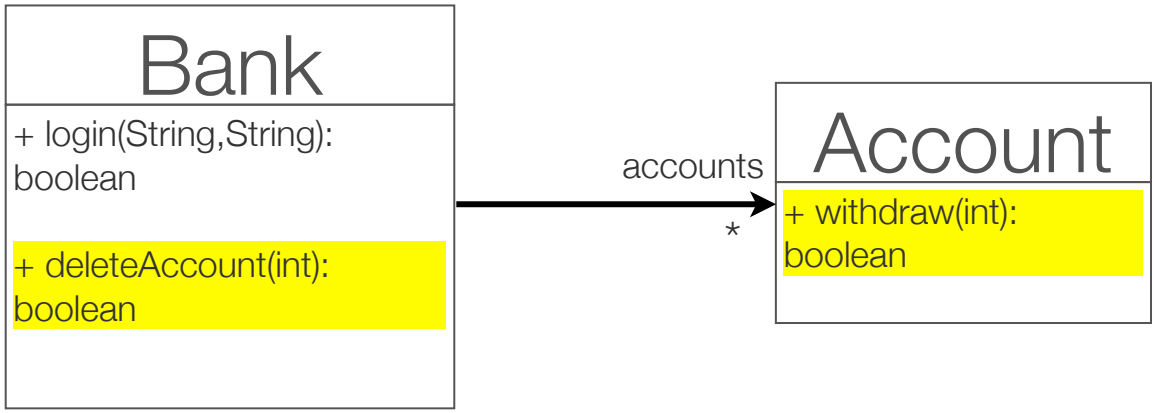

```
public void testAccessControlDelete() {
    bank.login(nonAuthorizedUser,password);
    addTracedAdvice("AccessControl");
    setAdviceTracerOn();
    try {
        account.withdraw(30);
    } catch(DeniedAccessException() {}
    setAdviceTracerOff();
    assertAdviceExecutionEquals(1);
    assertExecutedAdvice("AccessControl","Account.withdraw:47");
}
```

AdviceTracer: example – 5




```
public void testAccessControlDelete() {
    bank.login(nonAuthorizedUser,password);
    addTracedAdvice("AccessControl");
    setAdviceTracerOn();
    // System Test
    setAdviceTracerOff();
    assertAdviceExecutionEquals(4);
}
```

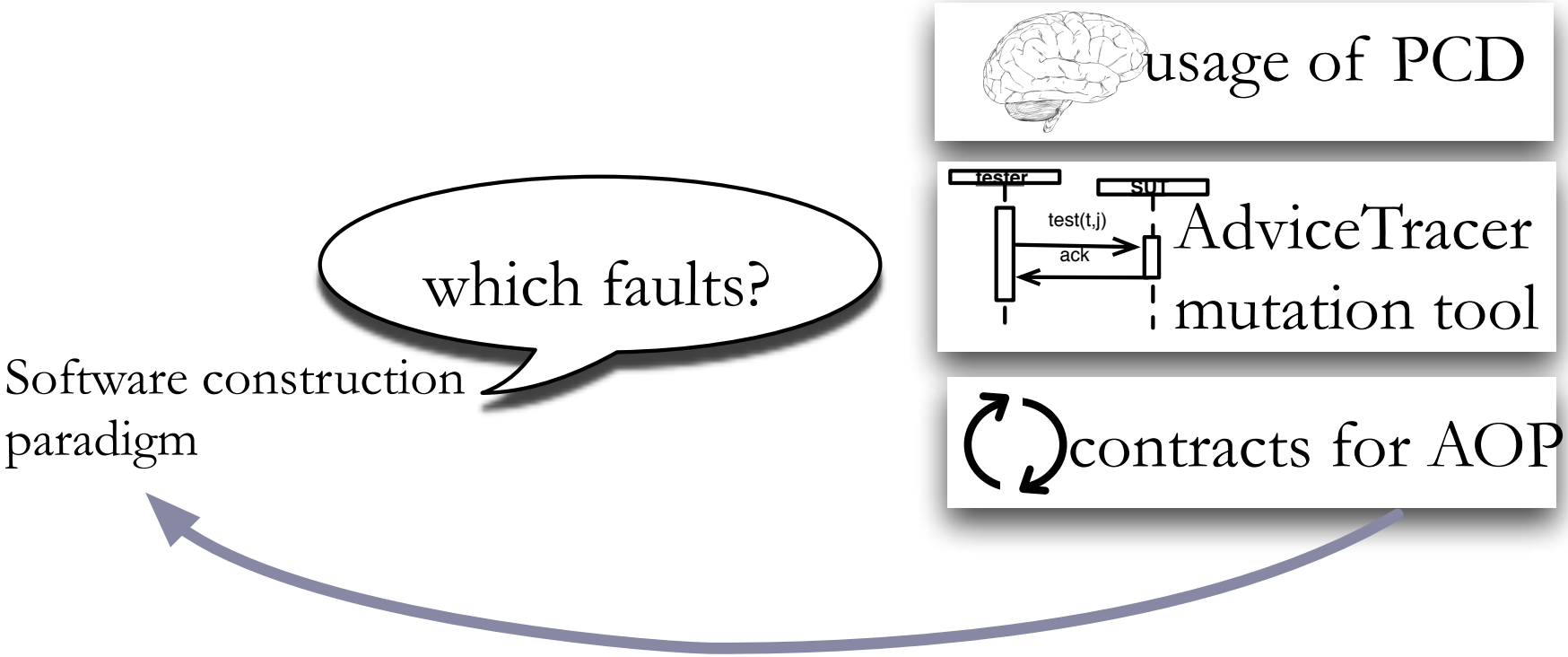
AdviceTracer: example – 5




```
public void testAccessControlDelete() {
    bank.login(nonAuthorizedUser,password);
    addTracedAdvice("AccessControl");
    setAdviceTracerOn();
    // System Test
    setAdviceTracerOff();
    assertAdviceExecutionEquals(4);
}
```

- R. Delamare, B. Baudry, S. Ghosh, Y. Le Traon. «An approach for testing pointcut descriptors in AspectJ». Journal of Software Testing, Verification and Reliability, 2011.

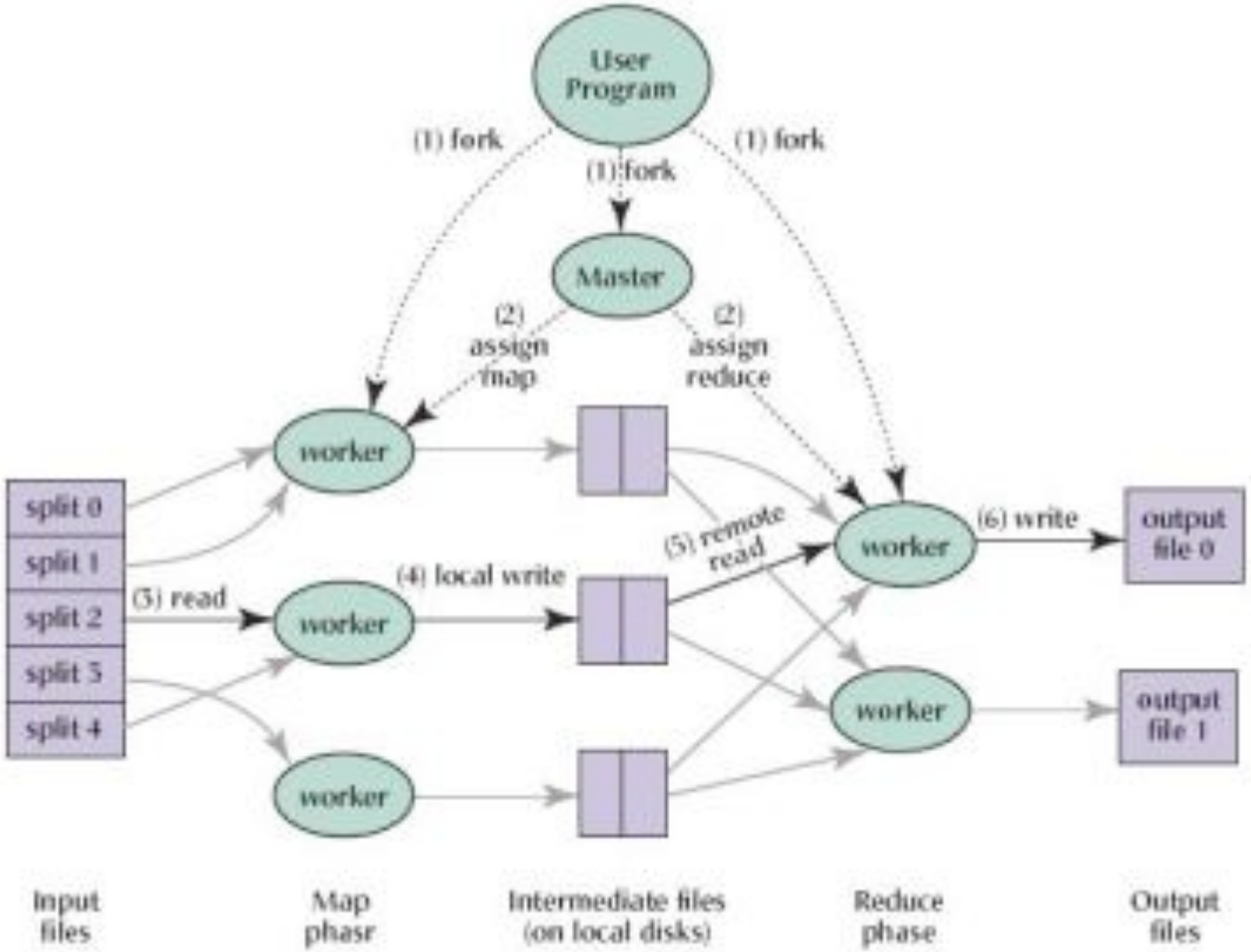
QLTF for AOP


usage of PCD
tester
SUT
test(t,j)
ack
AdviceTracer
mutation tool
contracts for AOP
which faults?
Software construction
paradigm

Cloud computing

- Build order of magnitude bigger systems with order of magnitude less efforts
  - http://boom.cs.berkeley.edu/
- Distribution, communication and synchronization are hidden
  - establish trust in these mechanisms
- Design applications as atoms and bits
  - reason on pieces of programs
  - understand the interactions between these pieces and the cloud framework

MapReduce


User Program
(1) fork
(1) fork
(1) fork
Master
(2) assign map
(2) assign reduce
worker
worker
worker
worker
worker
split 0
split 1
split 2
split 3
split 4
(3) read
(4) local write
(5) remote read
(6) write
output file 0
output file 1
Input files
Map phase
Intermediate files (on local disks)
Reduce phase
Output files

Word count

```
/**
 * Counts the words in each line.
 * For each line of input, break the line into words and emit them as
 * (<b>word</b>, <b>1</b>).
 */
public static class MapClass extends MapReduceBase
  implements Mapper<LongWritable, Text, Text, IntWritable> {

  private final static IntWritable one = new IntWritable(1);
  private Text word = new Text();

  public void map(LongWritable key, Text value,
                  OutputCollector<Text, IntWritable> output,
                  Reporter reporter) throws IOException {
    String line = value.toString();
    StringTokenizer itr = new StringTokenizer(line);
    while (itr.hasMoreTokens()) {
      word.set(itr.nextToken());
      output.collect(word, one);
    }
  }
}
```

Word count

```
/**
 * A reducer class that just emits the sum of the input values.
 */
public static class Reduce extends MapReduceBase
  implements Reducer<Text, IntWritable, Text, IntWritable> {

  public void reduce(Text key, Iterator<IntWritable> values,
                     OutputCollector<Text, IntWritable> output,
                     Reporter reporter) throws IOException {
    int sum = 0;
    while (values.hasNext()) {
      sum += values.next().get();
    }
    output.collect(key, new IntWritable(sum));
  }
}
```

MapReduce application

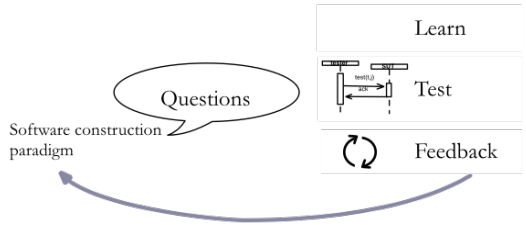
- Two independents parts
  - Mapper processes a bit of information; 1-to-1 input/output sources
  - Reducer processes n mapped bits of information in a global result; n-to-1 I/O sources
- + configuration, dispatch modules

- Map and reduce modules rely on
  - high level languages
    - low level arithmetic, control, etc. are hidden from the programmer
  - that have been heavily tested
    - i.e., we can trust the iterator Java library
  - framework that abstracts I/O and distribution operations
    - all errors are handled by the framework (e.g., Hadoop)

MapReduce faults

- Some faults in map and reduce
  - logic, arithmetic
- Most of them elsewhere
  - ‘cutting’ a function in map and reduce is a hard design task
  - understanding the MapReduce framework is hard

Conclusion



- New ways of building software
- require new ways of testing it
- Mutation analysis can play a central role
  - understanding faults in the methods and techniques
  - focusing testing on relevant defects
  - tailoring testing research questions