

IA {Test} AI

A new Battlefield!

Nadjib Lazaar

Ing - Phd - Assistant Professor - University of Montpellier - COCONUT Team

<http://www.lirmm.fr/~lazaar/>

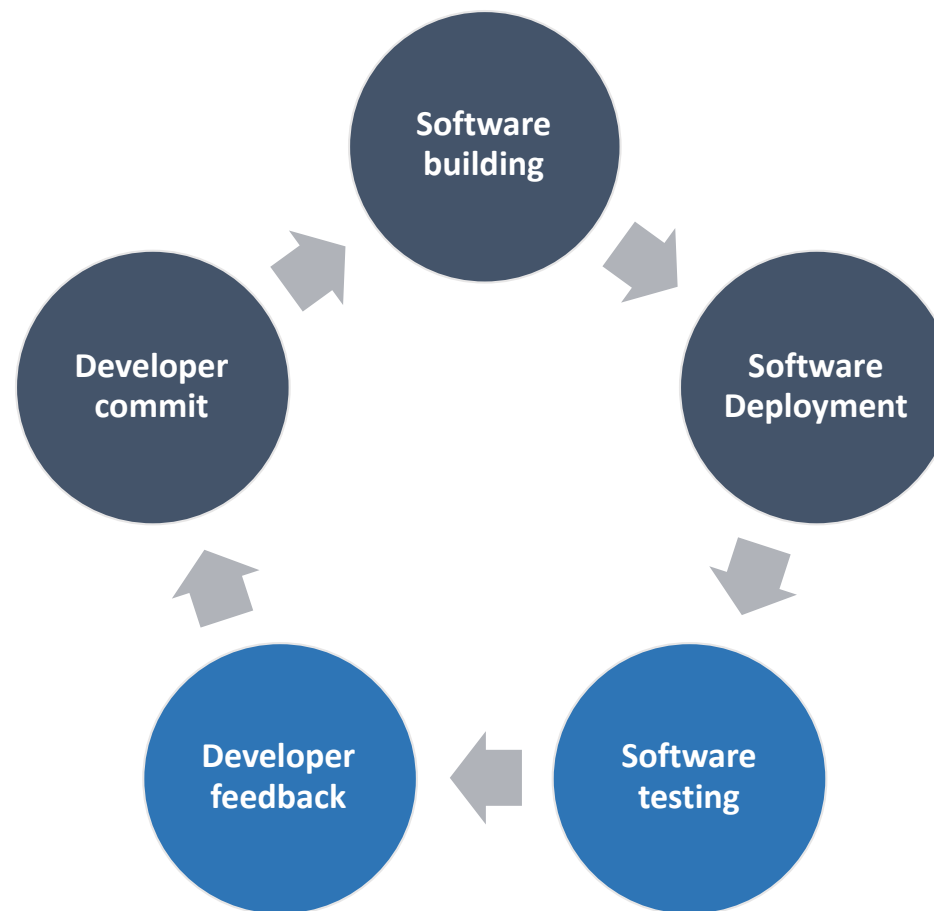
AI-based Testing

Continuous Integration Cycle

Software testing timeline

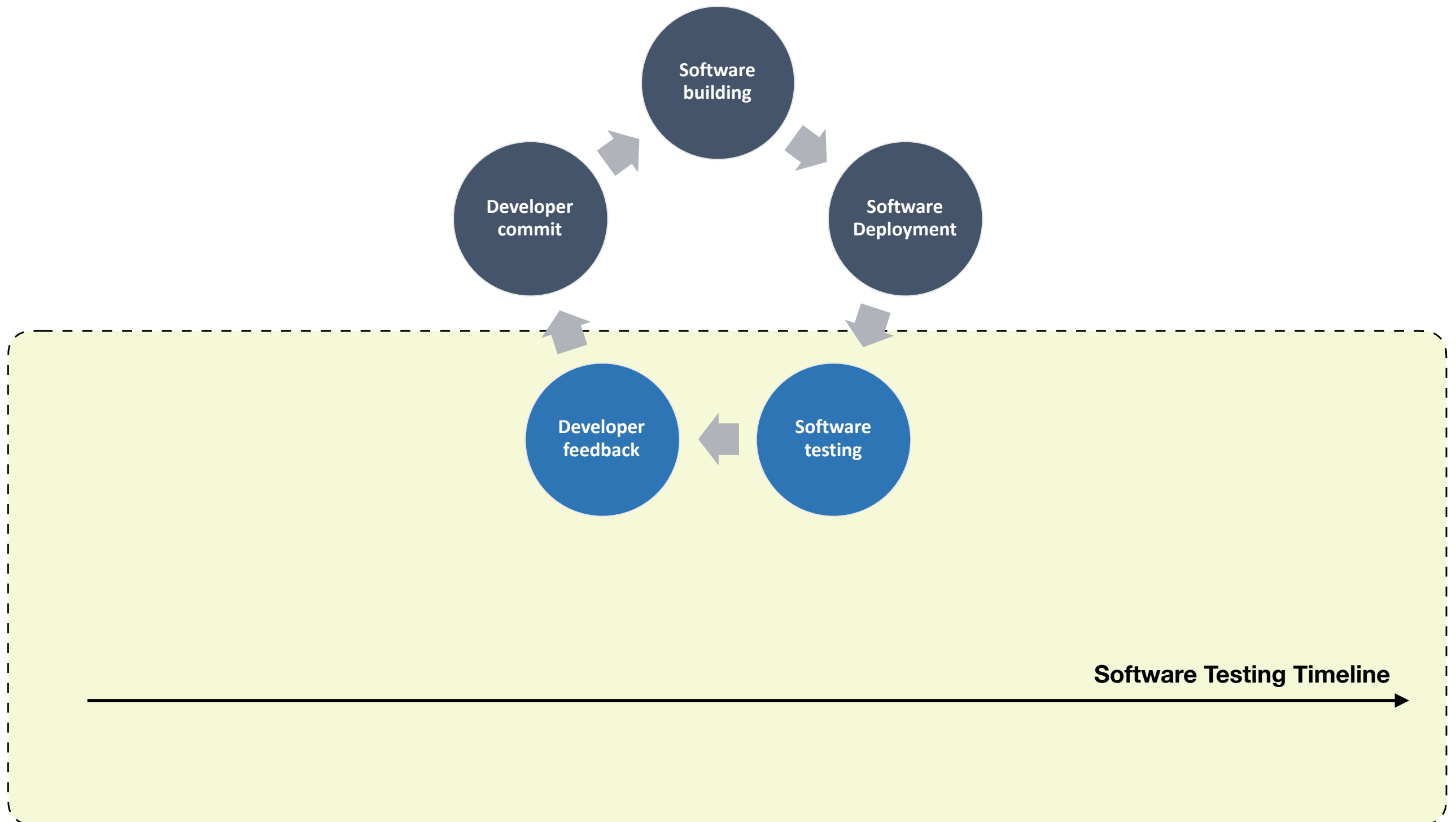
Continuous Integration Cycle

Software testing timeline



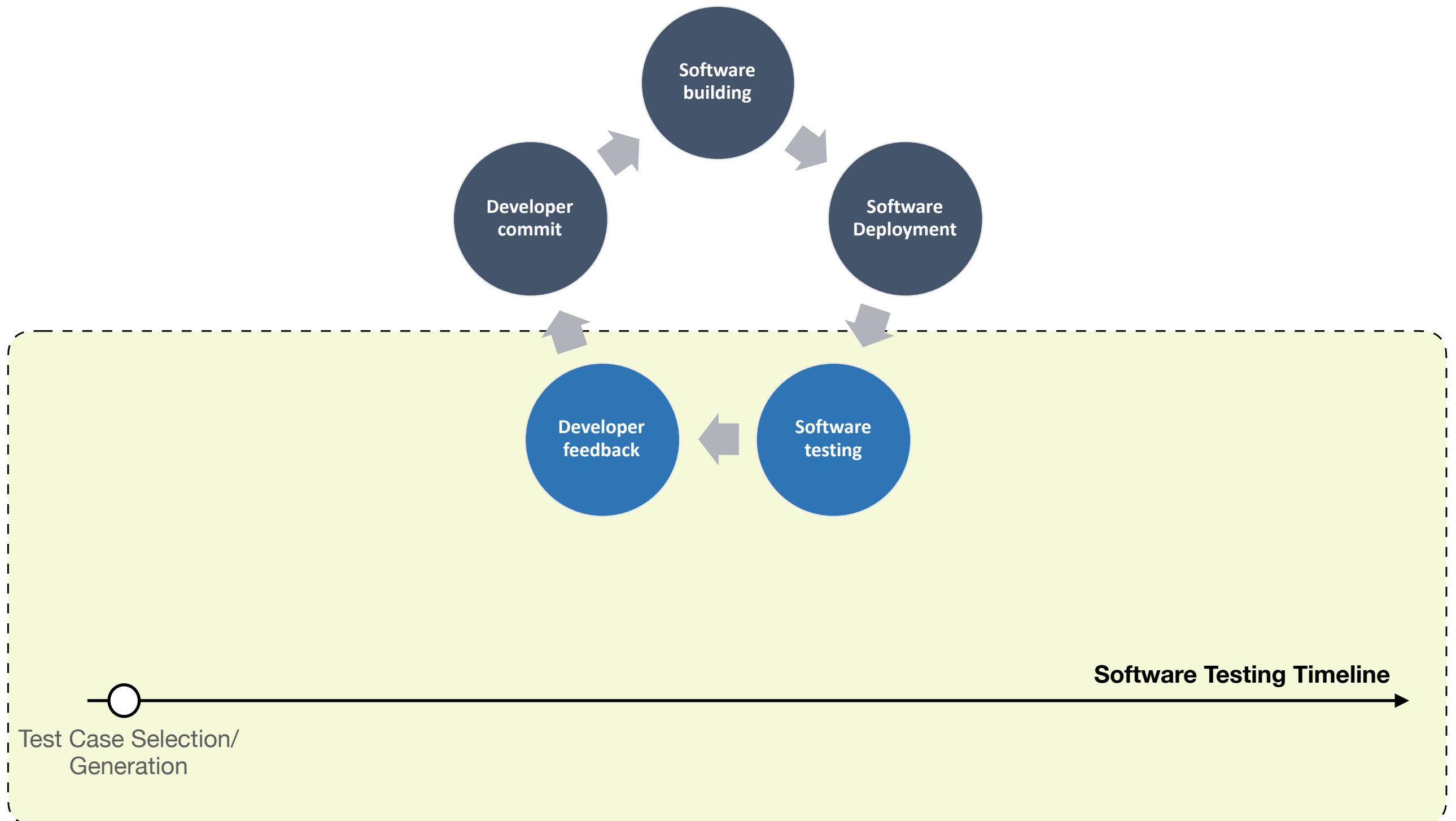
Continuous Integration Cycle

Software testing timeline



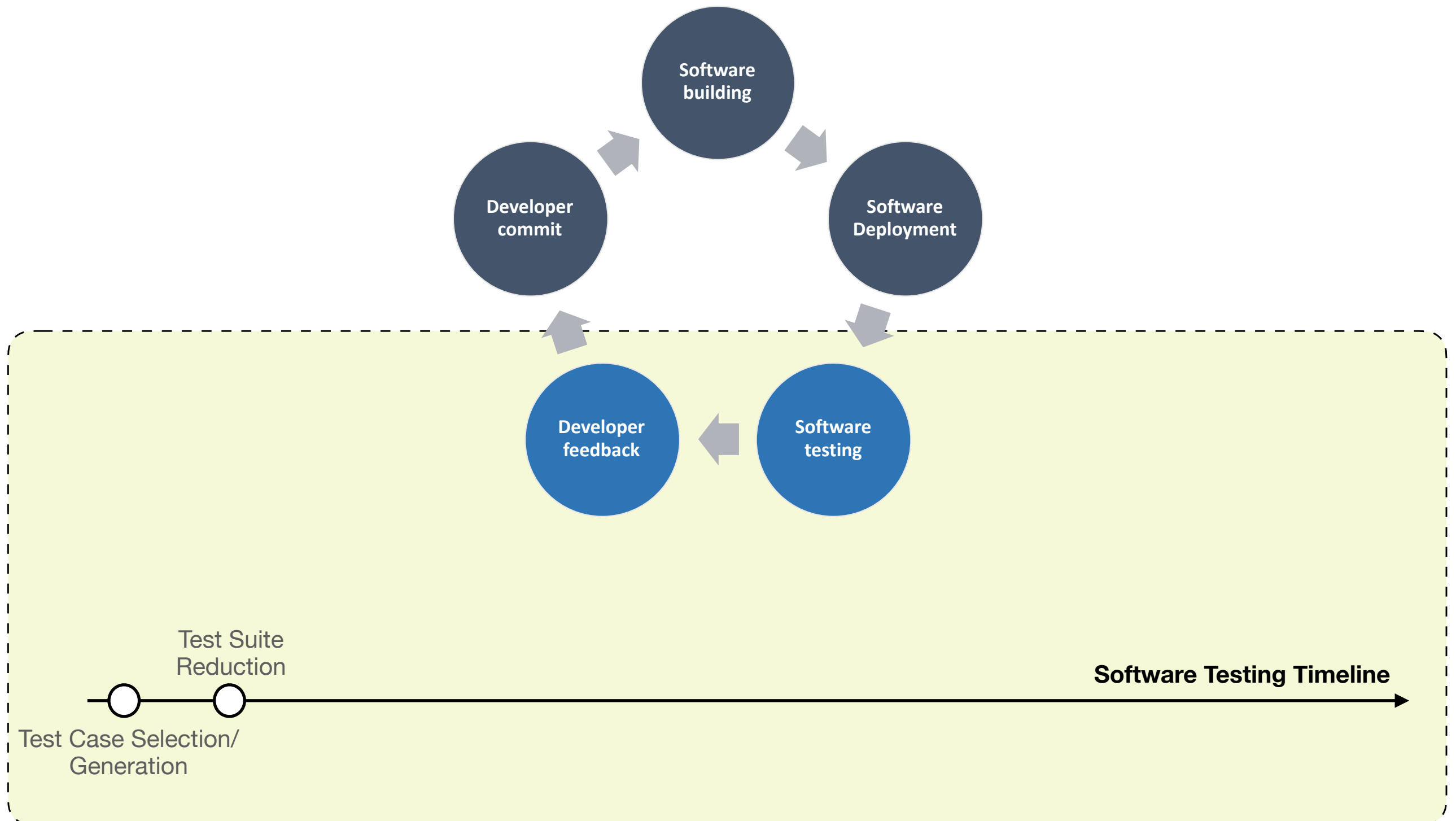
Continuous Integration Cycle

Software testing timeline



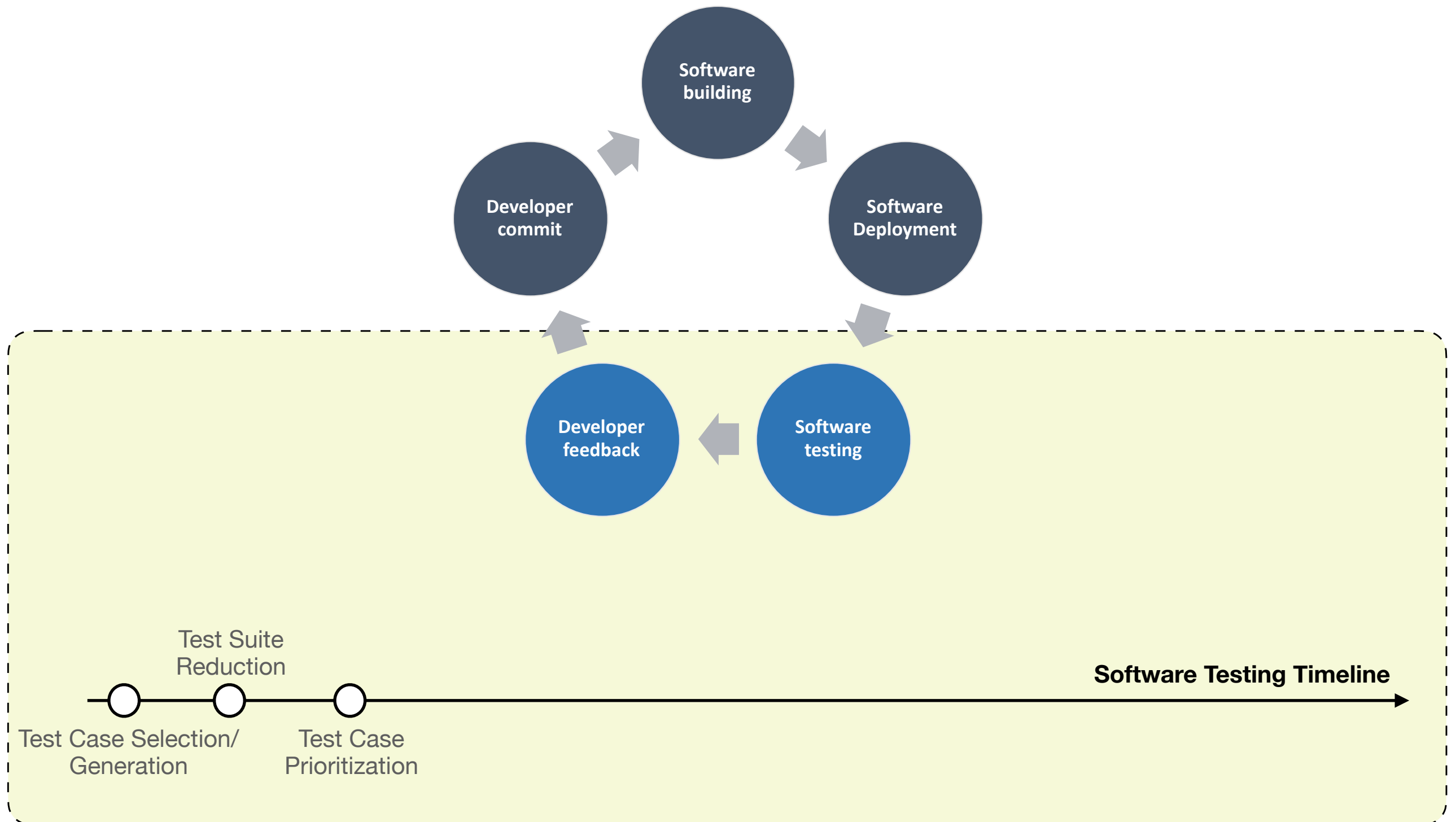
Continuous Integration Cycle

Software testing timeline



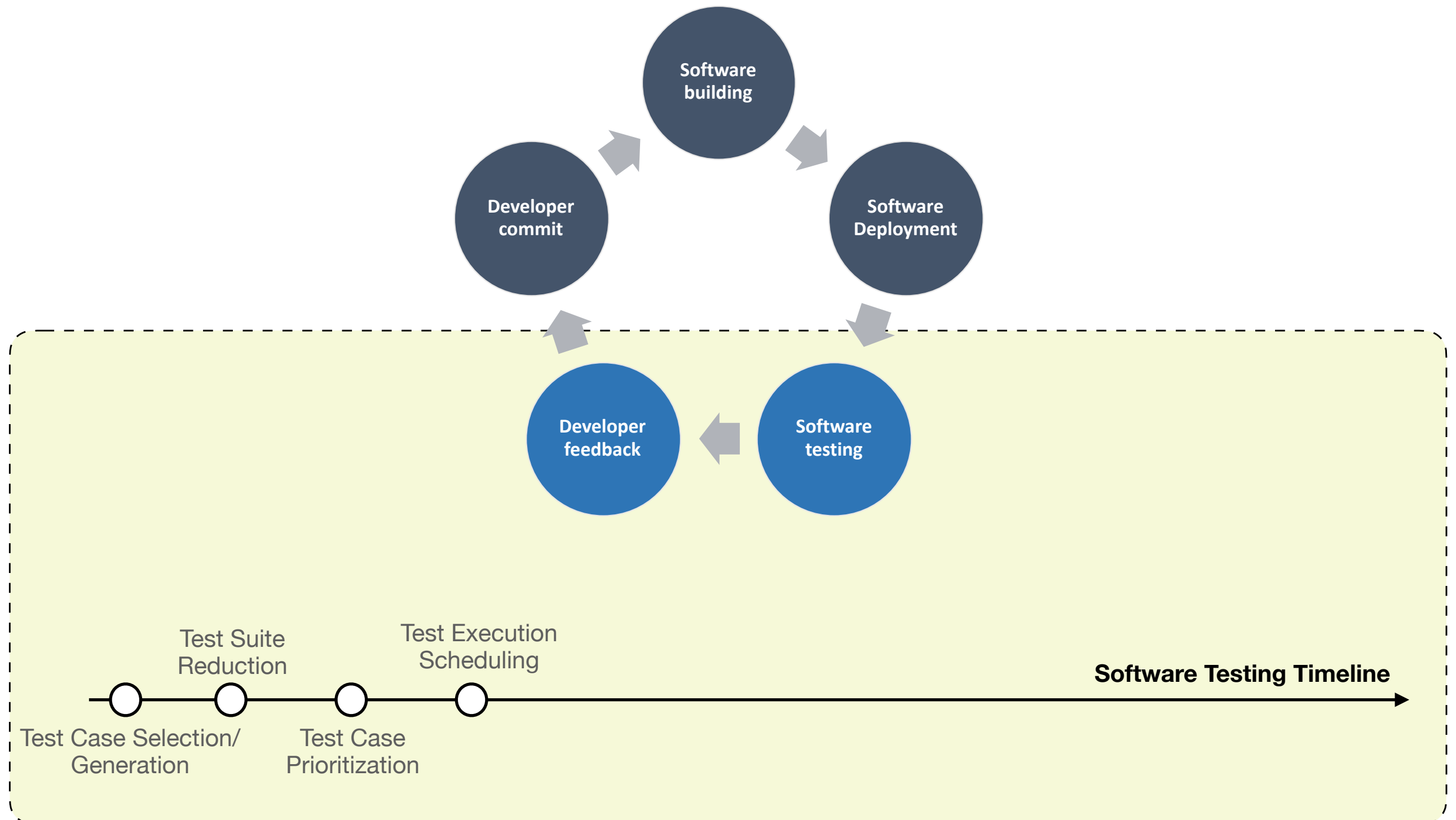
Continuous Integration Cycle

Software testing timeline



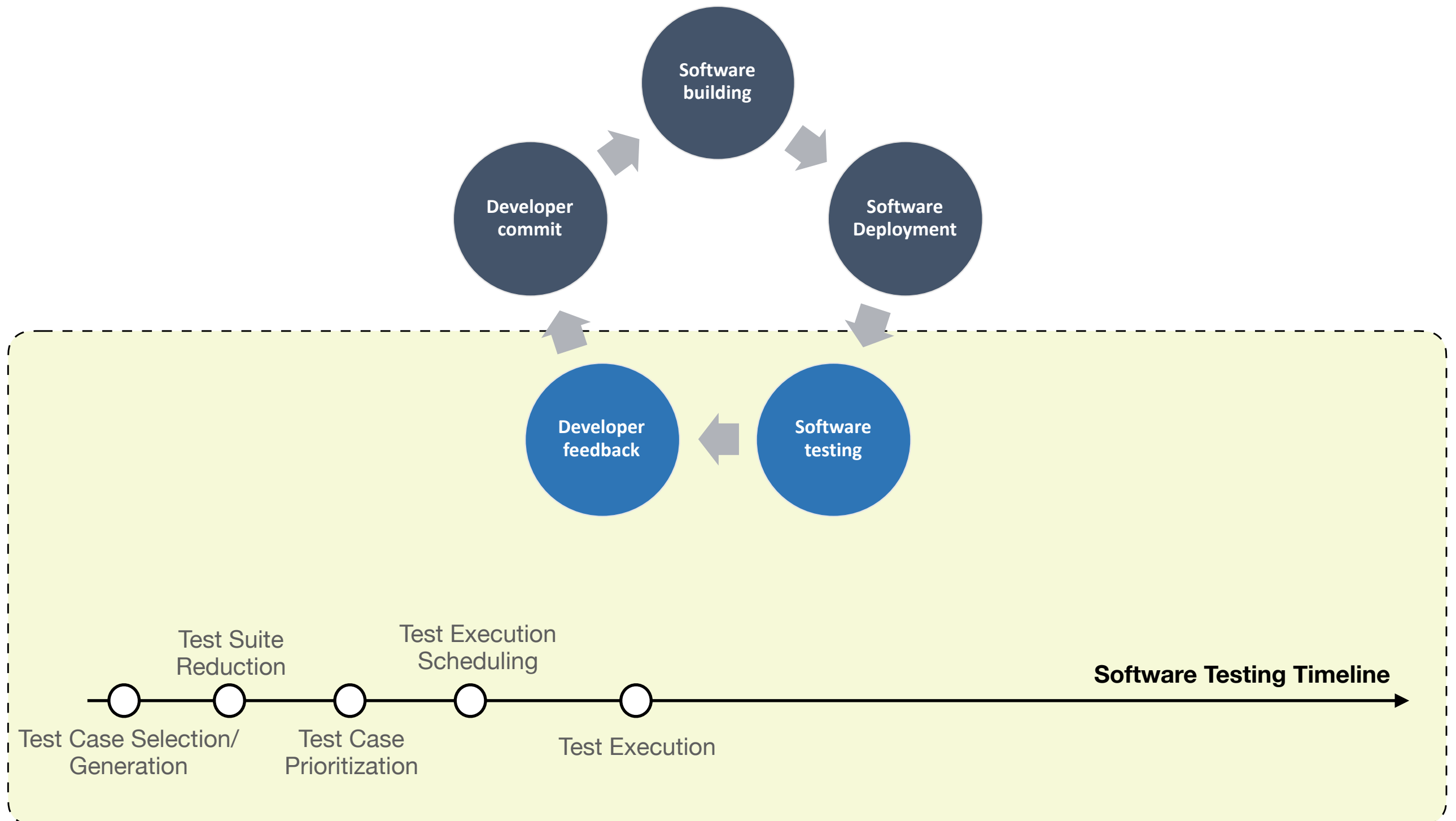
Continuous Integration Cycle

Software testing timeline



Continuous Integration Cycle

Software testing timeline



AI-based Testing

(AI₄VV)

Testing Tasks

- Automatic test case generation
- Test suite reduction
- Test case prioritization
- Test execution scheduling
- ...

AI-based Testing

Automatic Test Case Generation

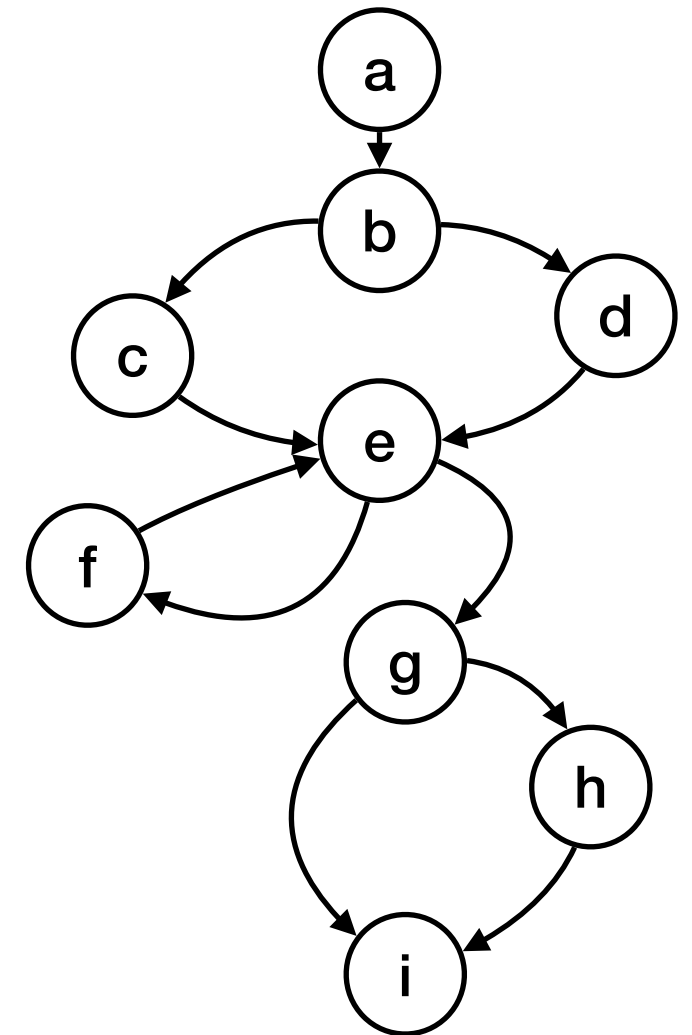
(AI4VW)

P

```
double power(int x, int y) {  
    int i, p;  
    i = 0;  
    double z = 1;  
    if (x < 0)  
        p = -x;  
    else  
        p = x;  
    while (i < p) {  
        z = z * x;  
        i = i + 1;  
    }  
    if (y < 0)  
        x = 1 / x;  
    return x;  
}
```

P_{SSA}

```
double power(int x_1, int y_1) {  
    int i_1, p_1;  
    i_1 = 0;  
    double z_1 = 1;  
    if (x_1 < 0)  
        p_1 = -x_1;  
    else  
        p_2 = x_1;  
    p_3 = phi(p_1, p_2);  
    while (i_1 < p_3) {  
        z_2 = z_1 * x_1;  
        i_2 = i_1 + 1;  
    }  
    z_3 = phi(z_1, z_2);  
    i_3 = phi(i_1, i_2);  
    if (y_1 < 0)  
        x_2 = 1 / x_1;  
    x_3 = phi(x_1, x_2);  
    return x_3;  
}
```



Automatic Test Case Generation

Exercise

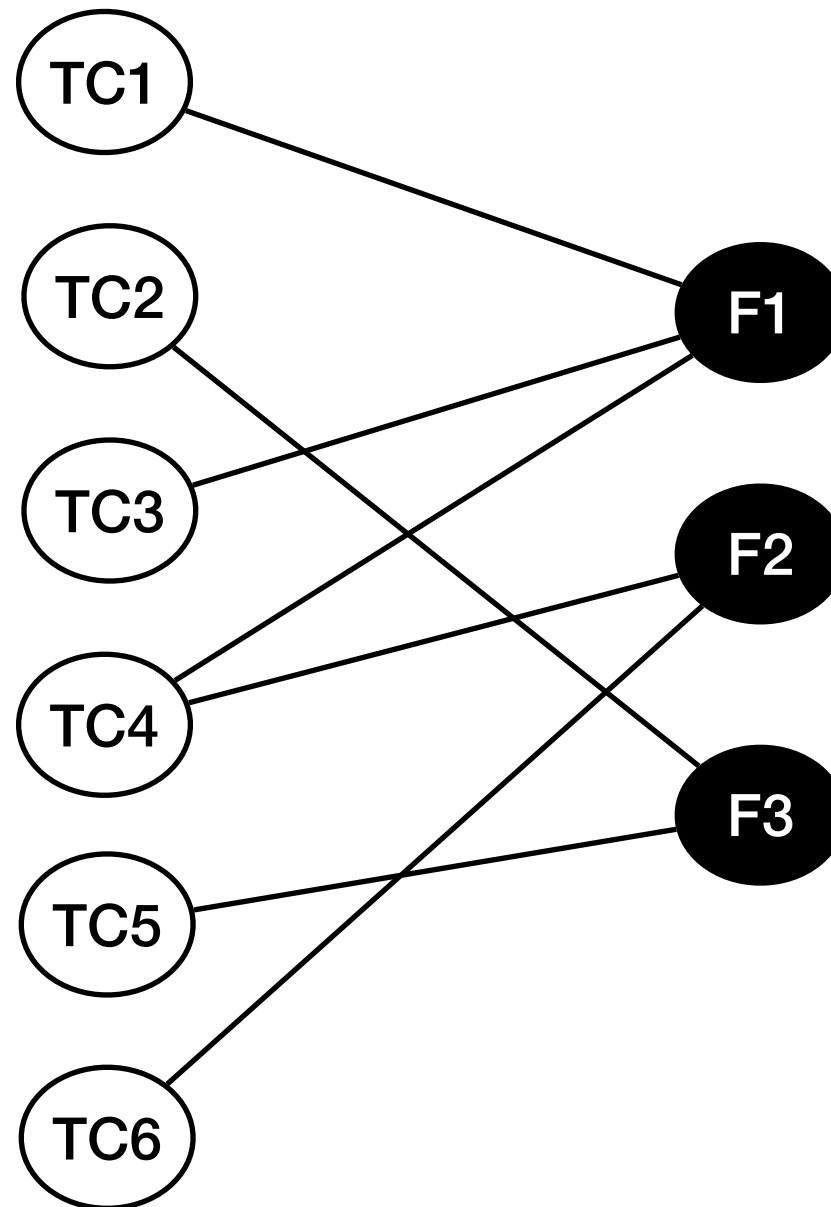
```
public int trityp(int a, int b, int c) {  
    int type;  
    if (a <= 0 || b <= 0 || c <= 0)  
        type = 1;  
    if (a >= (b + c) || c >= (b + a) || b >= (a + c))  
        type = 2;  
    if (a == b && b == c)  
        type = 3;  
    else if (((a * a) + (b * b)) == (c * c) || ((a * a) + (c * c)) == (b * b)  
             || ((c * c) + (b * b)) == (a * a))  
        type = 4;  
    else if (a != b && b != c && c != a)  
        type = 5;  
    else if ((a == b && b != c) || (a != b && c == a) || (c == b && c != a))  
        type = 6;  
    return type;  
}
```

Q: Give the corresponding FCG + the SSA program version + the 6 constraint networks to cover the 6 statements (from a to f)

AI-based Testing

Test Suite Reduction

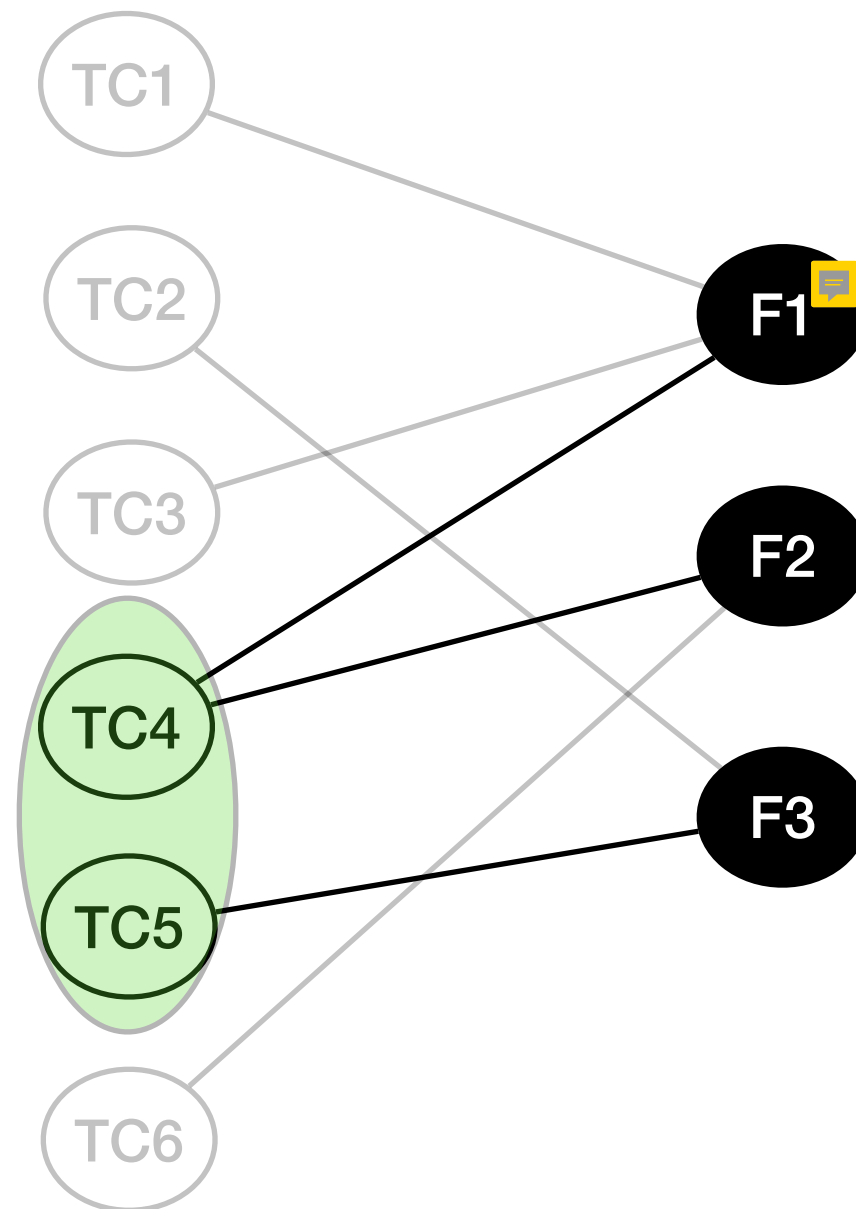
(AI₄VV)



AI-based Testing

Test Suite Reduction

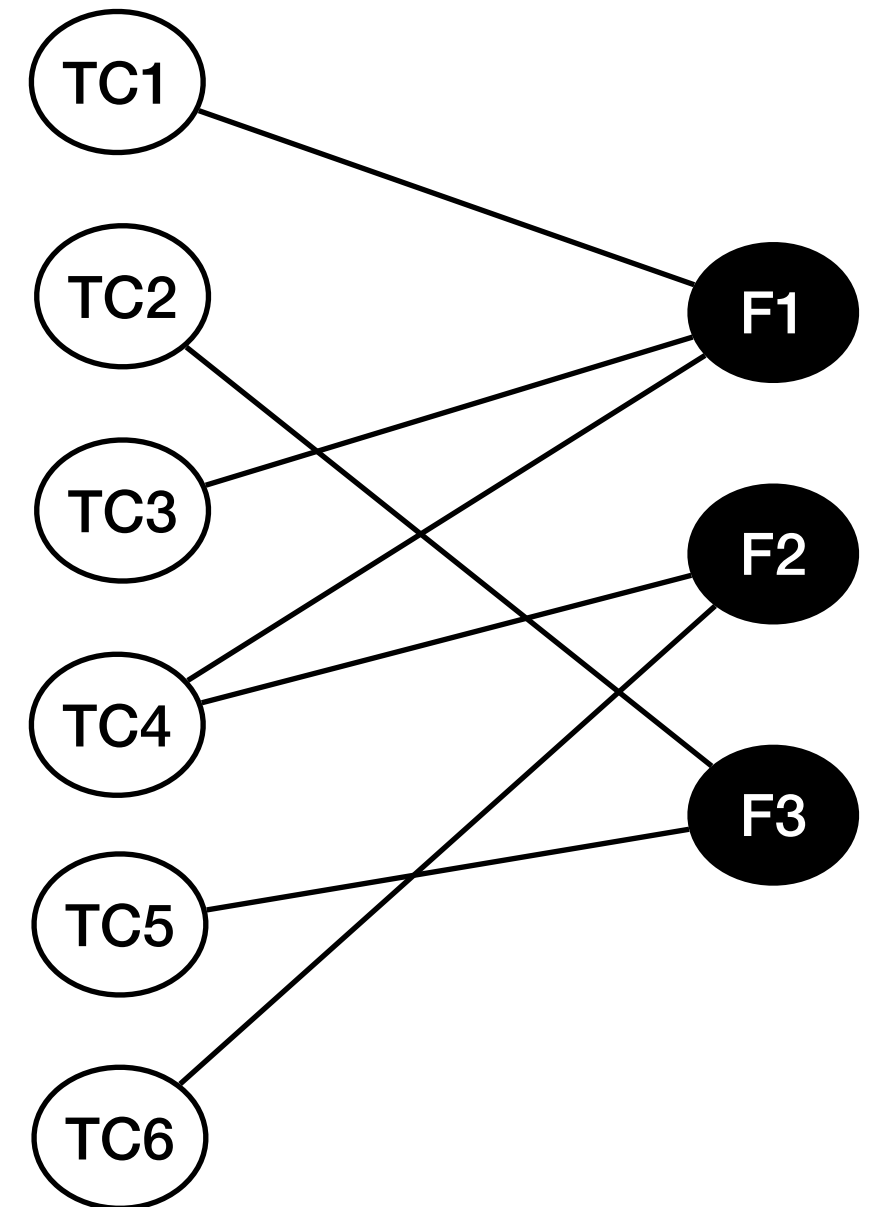
(AI₄VV)



AI-based Testing

Test Suite Reduction

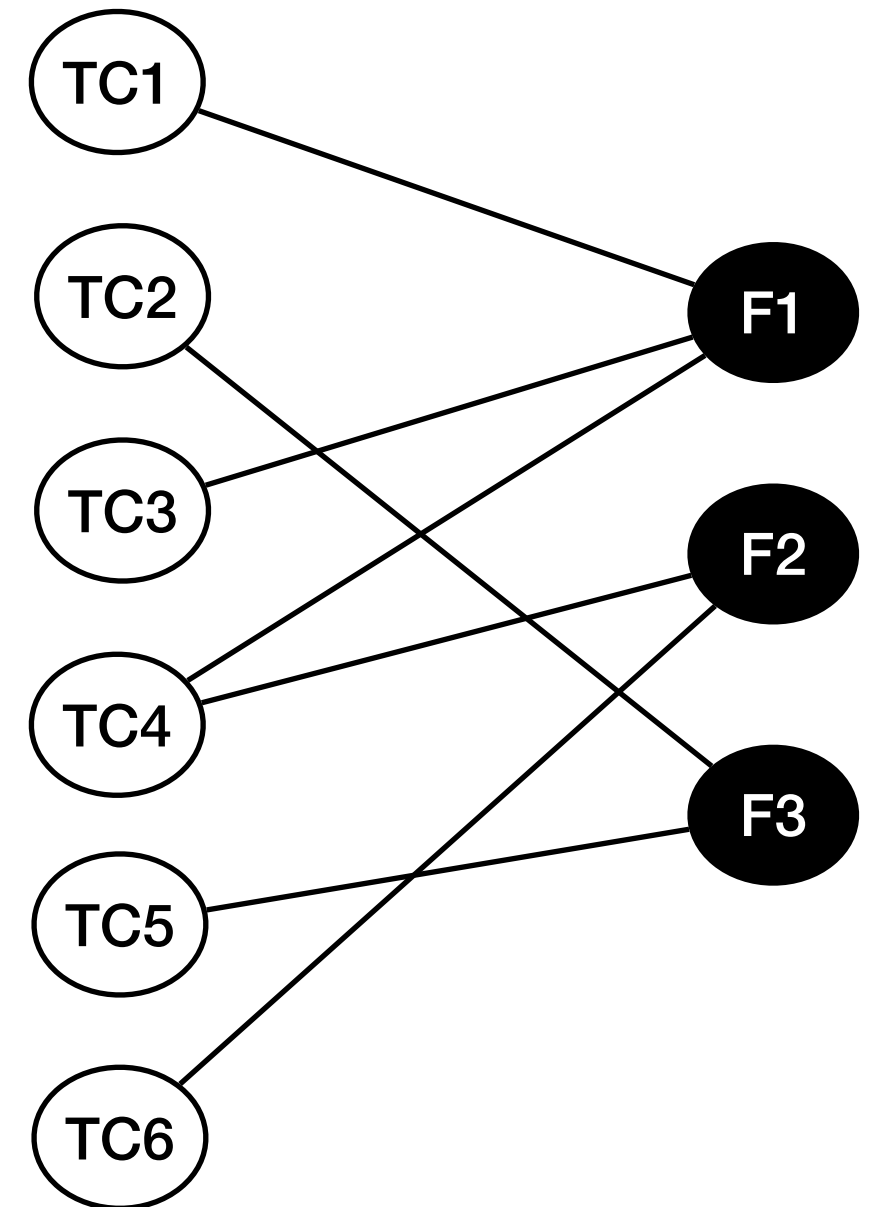
(AI₄VV)



AI-based Testing

Test Suite Reduction

- Vertex Cover Problem in a bipartite graph



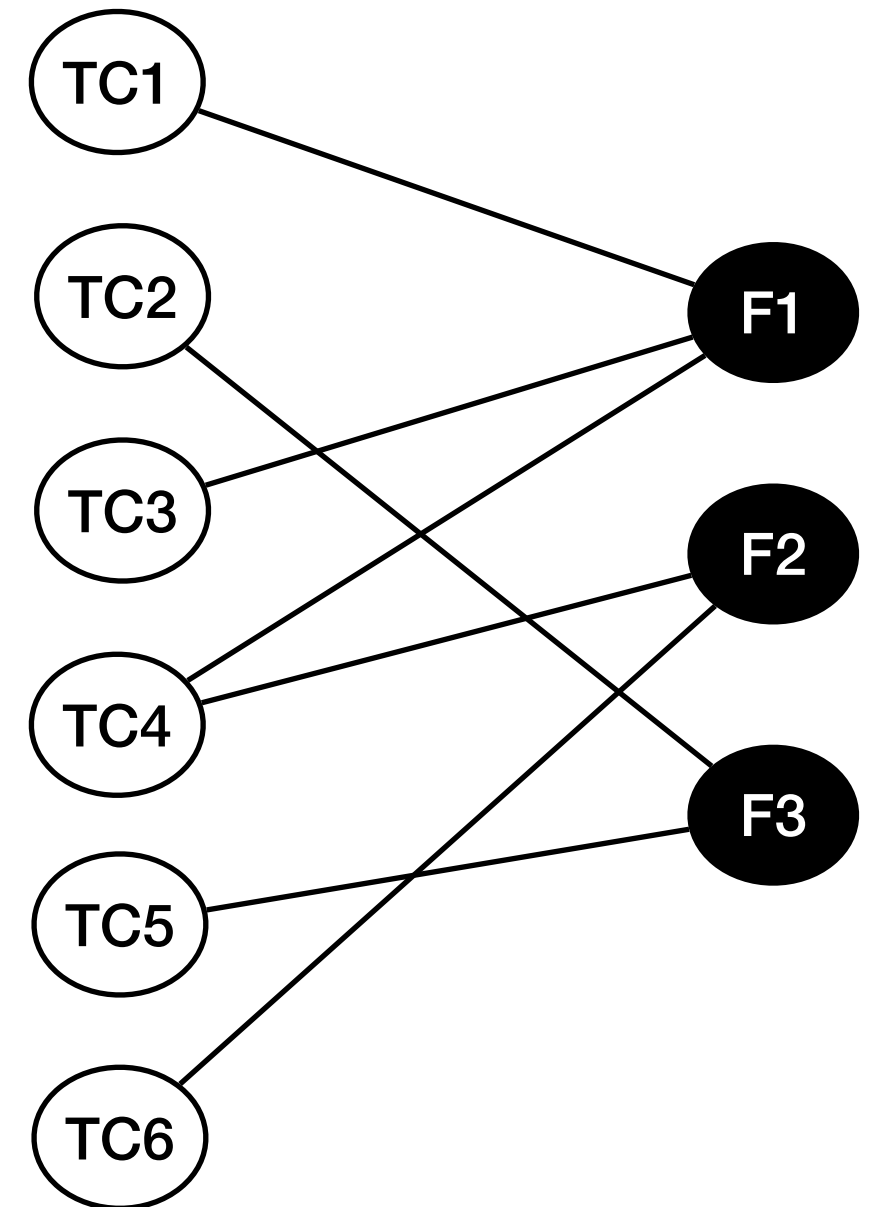
(AI₄VV)

AI-based Testing

Test Suite Reduction

- Vertex Cover Problem in a bipartite graph
- NP-Hard problem

(AI₄VV)

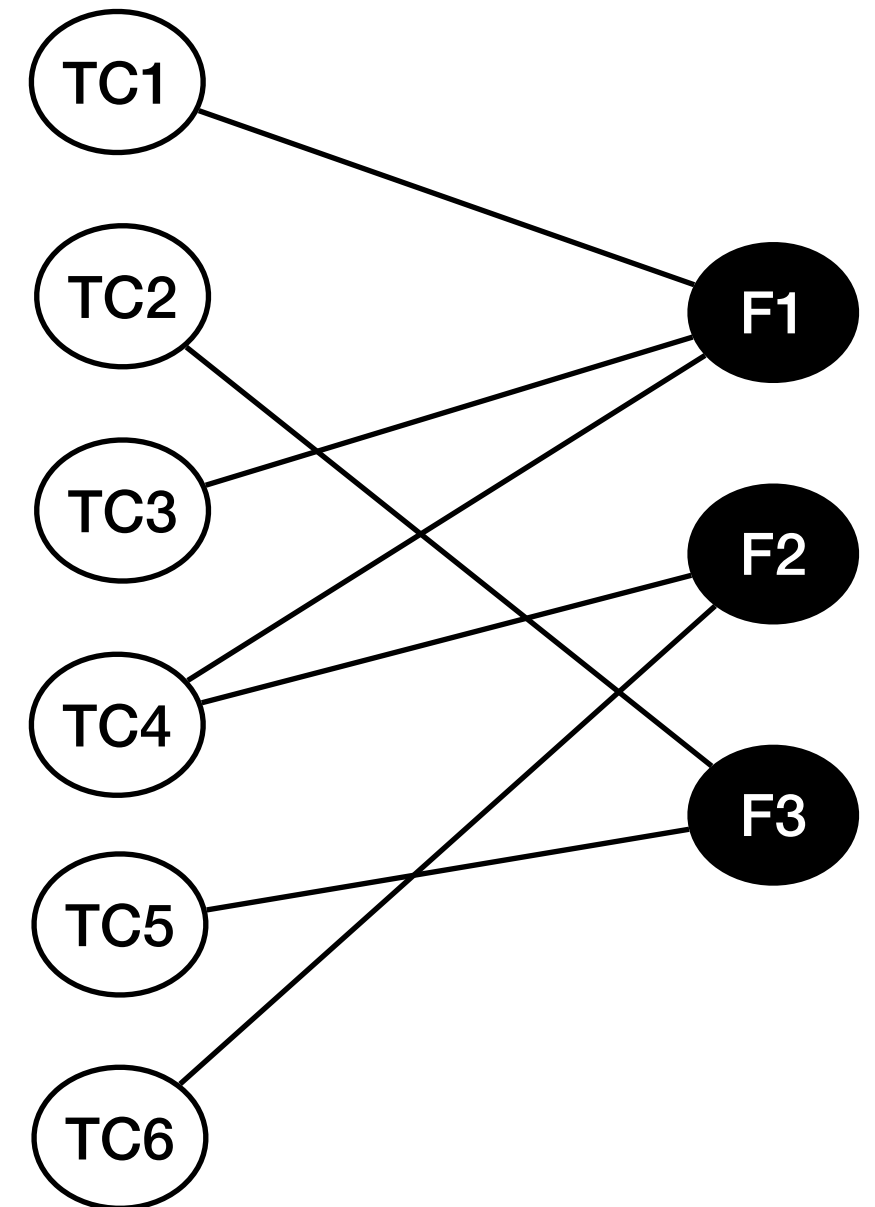


AI-based Testing

Test Suite Reduction


- Vertex Cover Problem in a bipartite graph
- NP-Hard problem
- Existing approaches:

(AI₄VV)

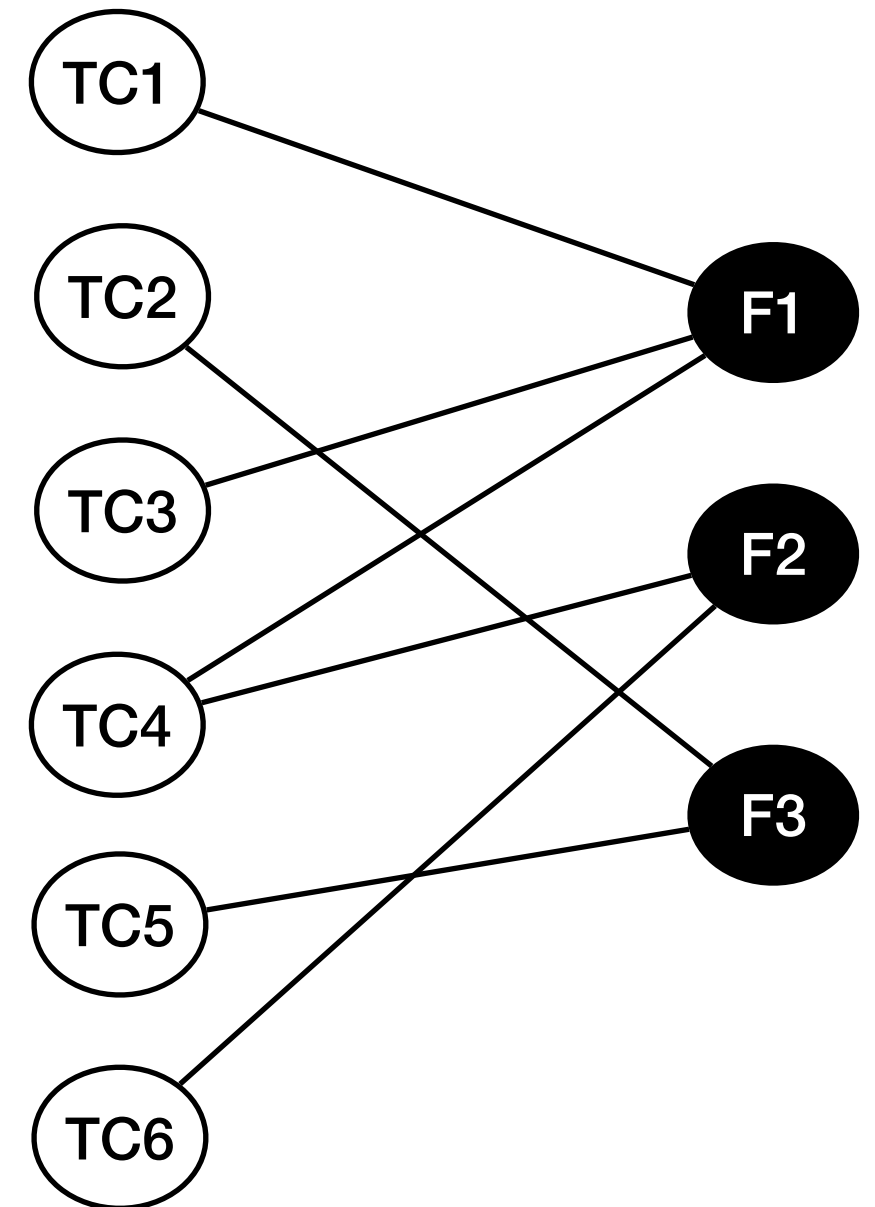


AI-based Testing

Test Suite Reduction

- Vertex Cover Problem in a bipartite graph
- NP-Hard problem
- Existing approaches:
 - ILP  approaches [Hsu Orso ICSE 2009, Campos Abreu QSIC 0213,...]

(AI₄VV)



AI-based Testing

Test Suite Reduction

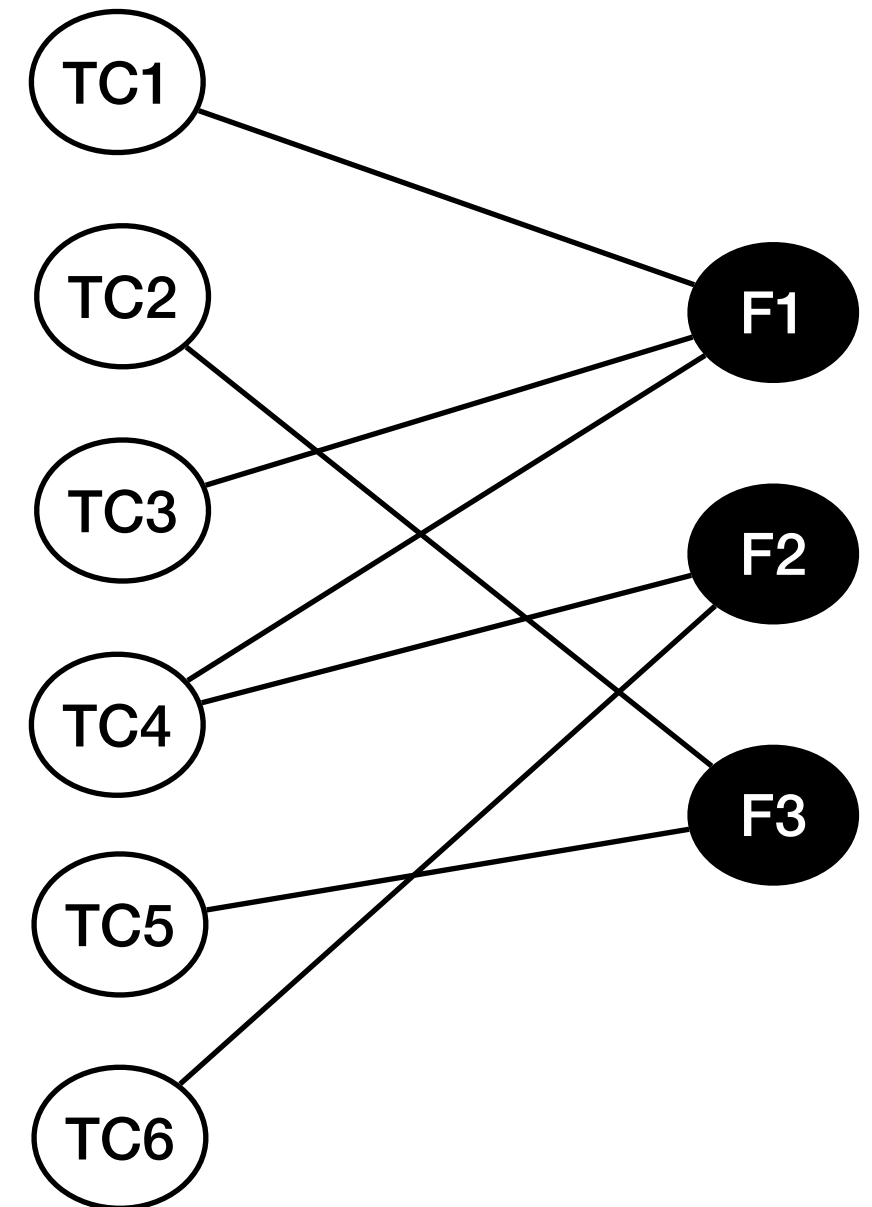
- Vertex Cover Problem in a bipartite graph
- NP-Hard problem
- Existing approaches:
 - ILP approaches [Hsu Orso ICSE 2009, Campos Abreu QSIC 0213,...]

$$\text{Minimize} \quad \sum_{i \in [1,6]} x_i$$

Subject to:

- $x_1 + x_3 + x_4 \geq 1$
- $x_4 + x_6 \geq 1$
- $x_2 + x_5 \geq 1$

(AI₄VV)

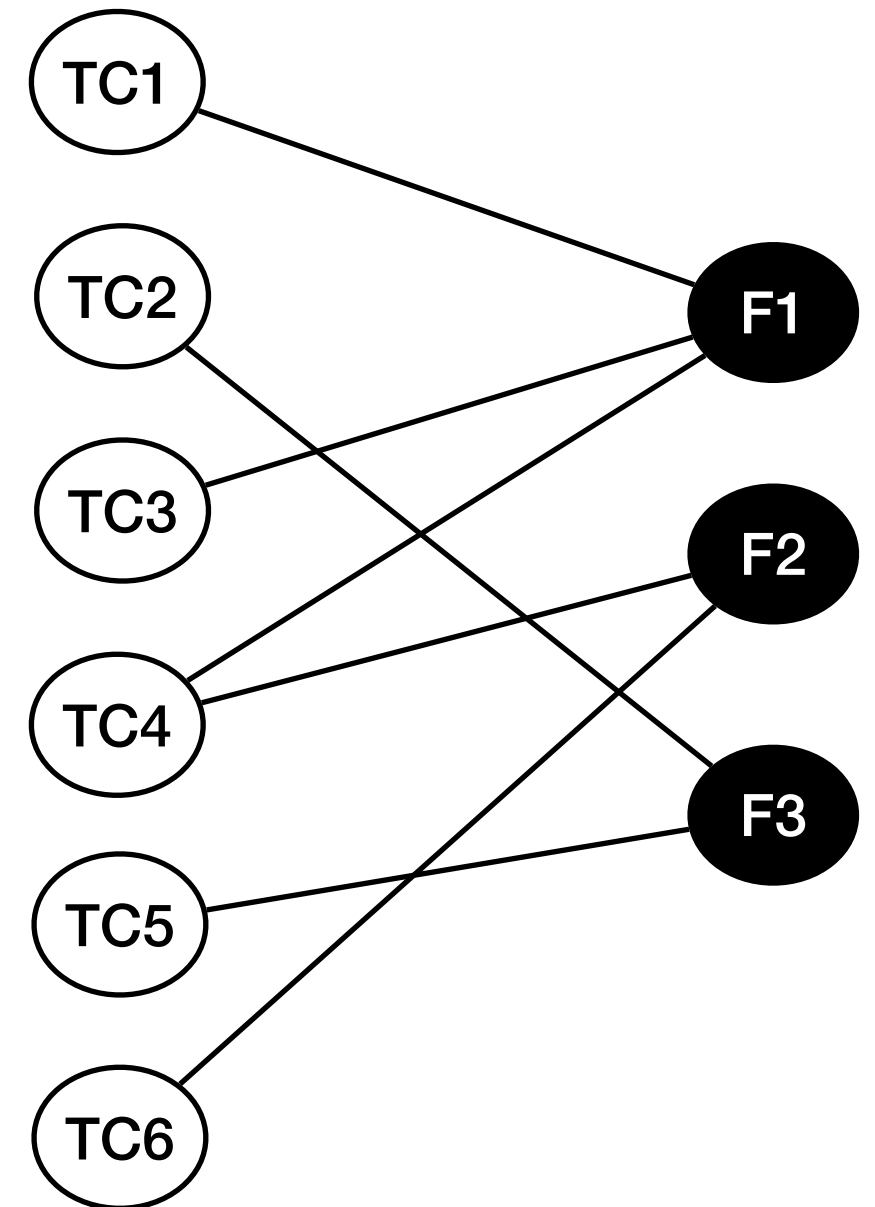


AI-based Testing

Test Suite Reduction

- Vertex Cover Problem in a bipartite graph
- NP-Hard problem
- Existing approaches:
 - ILP approaches [Hsu Orso ICSE 2009, Campos Abreu QSIC 0213,...]
 - Approximation algorithms [Harrold et al. TOSEM 1993,...]

(AI₄VV)



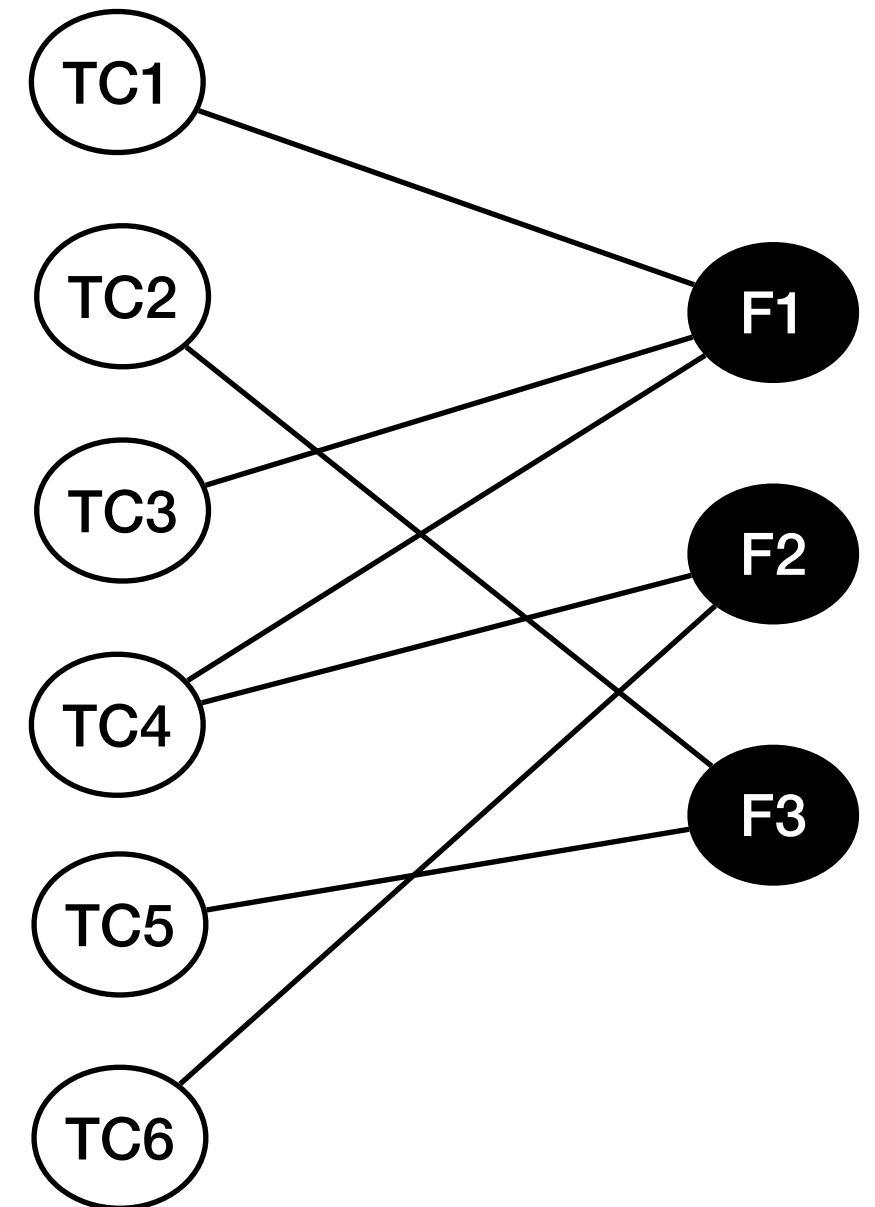
AI-based Testing

Test Suite Reduction

- Vertex Cover Problem in a bipartite graph
- NP-Hard problem
- Existing approaches:
 - ILP approaches [Hsu Orso ICSE 2009, Campos Abreu QSIC 0213,...]
 - Approximation algorithms [Harrold et al. TOSEM 1992, ...]

```
F={F1, F2, F3}; S=∅  
while(S≠∅)  
    Pick TCi covering max(F\S)  
    S= S ∪ cover(TCi)  
Return S
```

(AI₄VV)

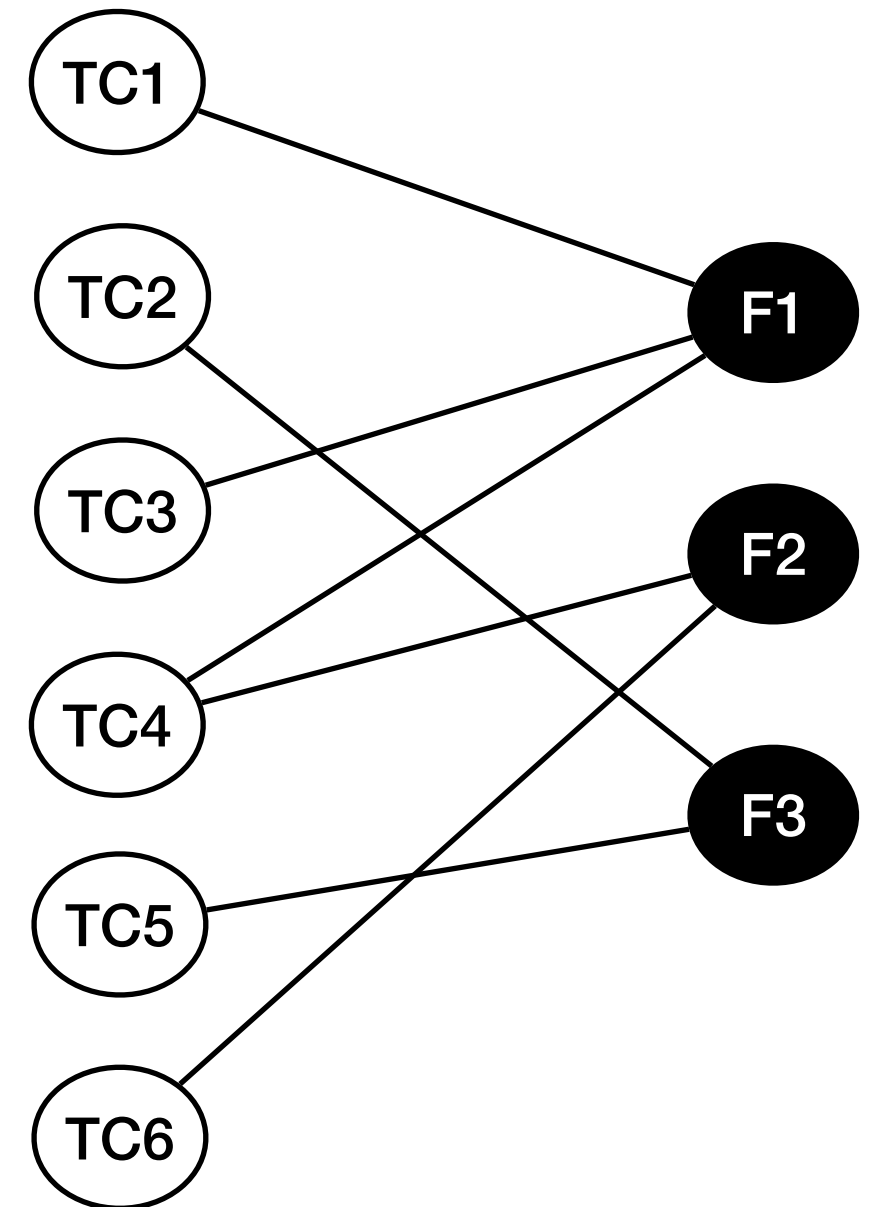


AI-based Testing

Test Suite Reduction

- Vertex Cover Problem in a bipartite graph
- NP-Hard problem
- Existing approaches:
 - ILP approaches [Hsu Orso ICSE 2009, Campos Abreu QSIC 0213,...]
 - Approximation algorithms [Harrold et al. TOSEM 1993,...]
 - Constraint Programming [Gotlieb et al. ISSTA 2014, AI Maganize 2016,...]

(AI₄VV)

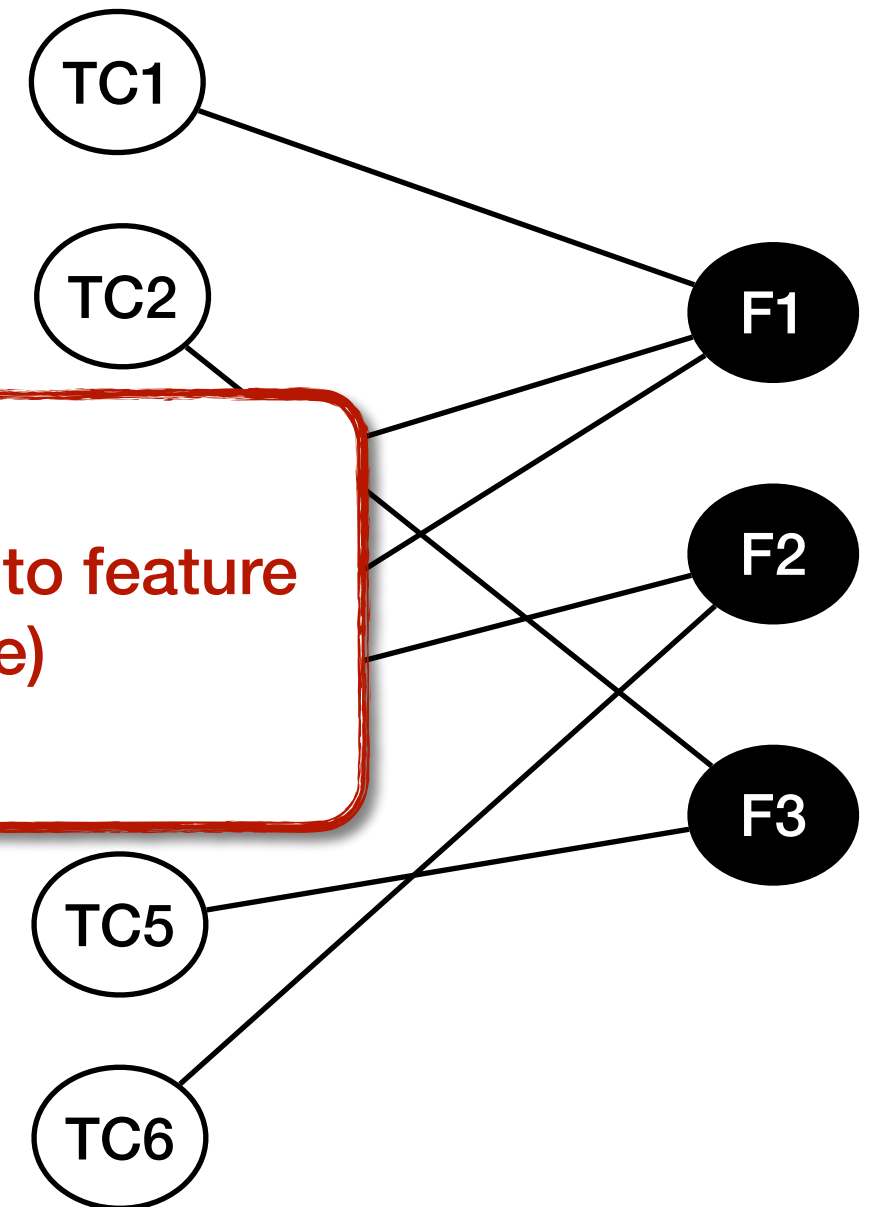


AI-based Testing

(AI₄VV)

Test Suite Reduction

- Vertex Cover Problem in a bipartite graph
- NP-Hard problem
- Existing approaches
 - ILP approach [Abreu QSI 2014]
 - Approximation algorithms [1993,...]
 - Constraint Programming [Gotlieb et al. ISSTA 2014, AI Magazine 2016,...]

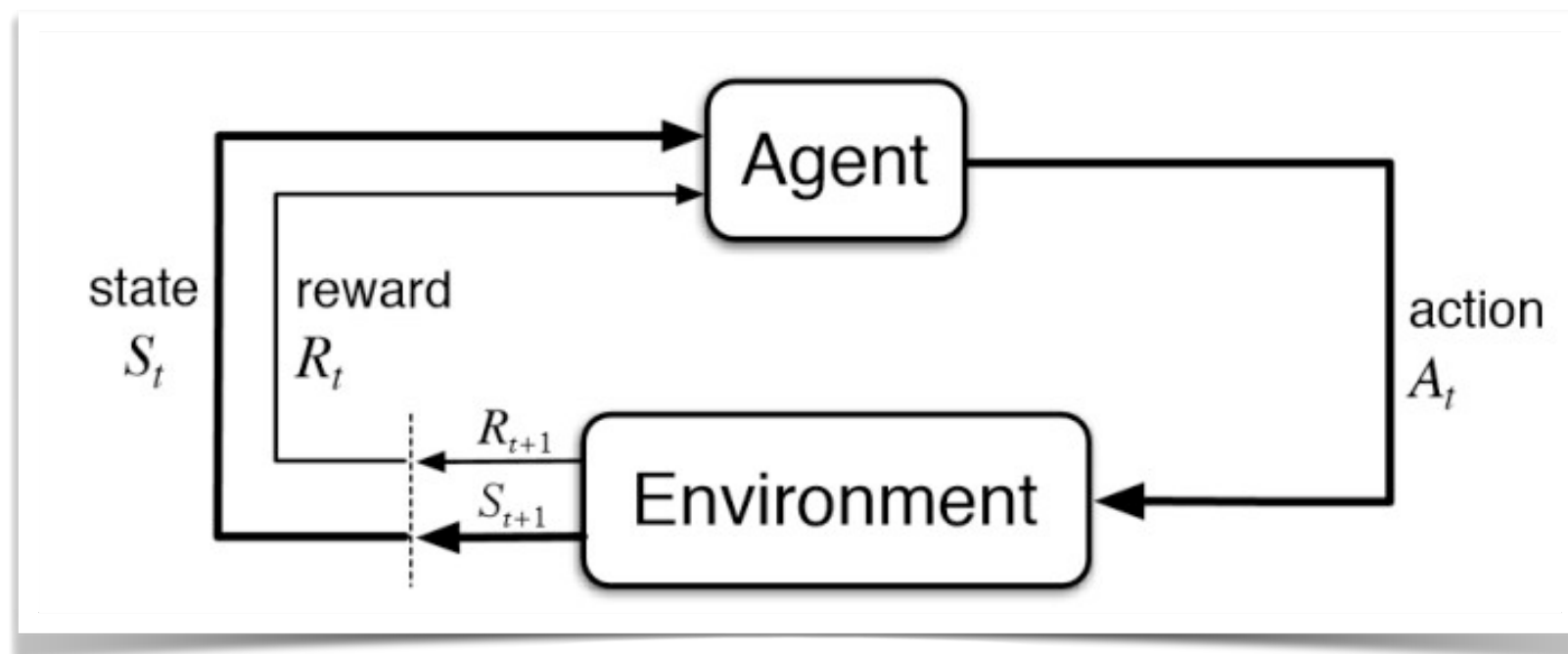


AI-based Testing

(AI4VW)

Test Case Prioritization

- Adaptive Testing:
 - focus on the more error-prone parts of the system
 - Execution environment (available devices, limited time and resources, previous feedback from continuous integration cycle)
-



AI-based Testing

(AI4V)

Test Execution Scheduling

- Test cases with distinct characteristics using limited resources
- Limited number of test cases that can be executed simultaneously
- Test cases sharing same resources cannot be executed simultaneously
- Each test case must be executed!
- Timespan to minimize (the overall schedule duration)



Testing AI-based systems

Testing AI-based systems

Crucial and Challenging Task

- Widespread application of AI
- AI-based systems are vulnerable
- Vulnerability of AI-based systems can lead to fatal failures
- The verification/validation of AI-based systems still involve too much human labour
- AI-based system behaviours evolve with changing contexts
- Testing of AI-based systems faces a number of challenges compared to testing of traditional systems

Testing AI-based systems

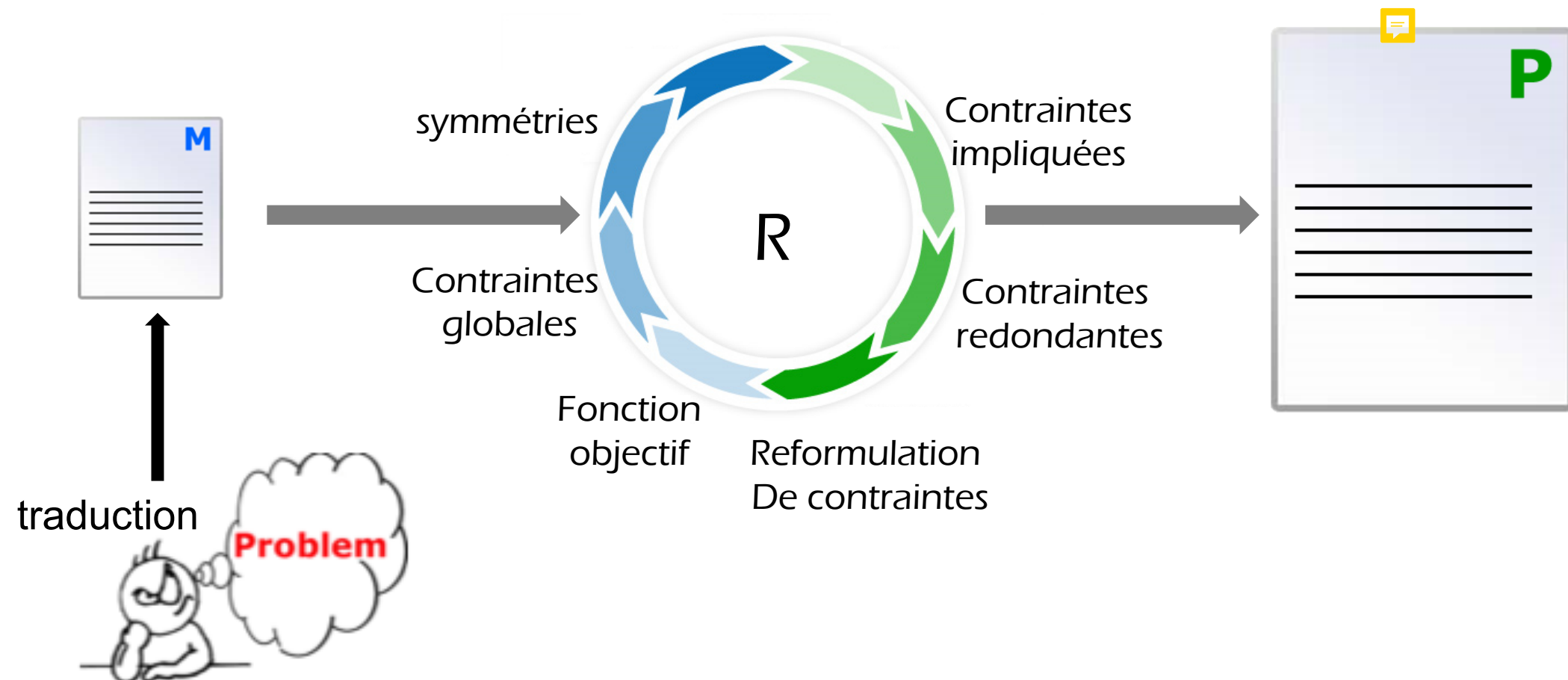
« Testing AI »

- AI point of view:
 - Testing AI is performed to estimate its accuracy, and improve its performance
 - During model creation, using validation and test datasets, to evaluate the model fit on the training dataset
- Software testing community
 - Testing has a wider scope, aiming to evaluate the system behaviour against a range of quality attributes (functional/non-functional requirements)

Testing AI-based systems

CPTTEST

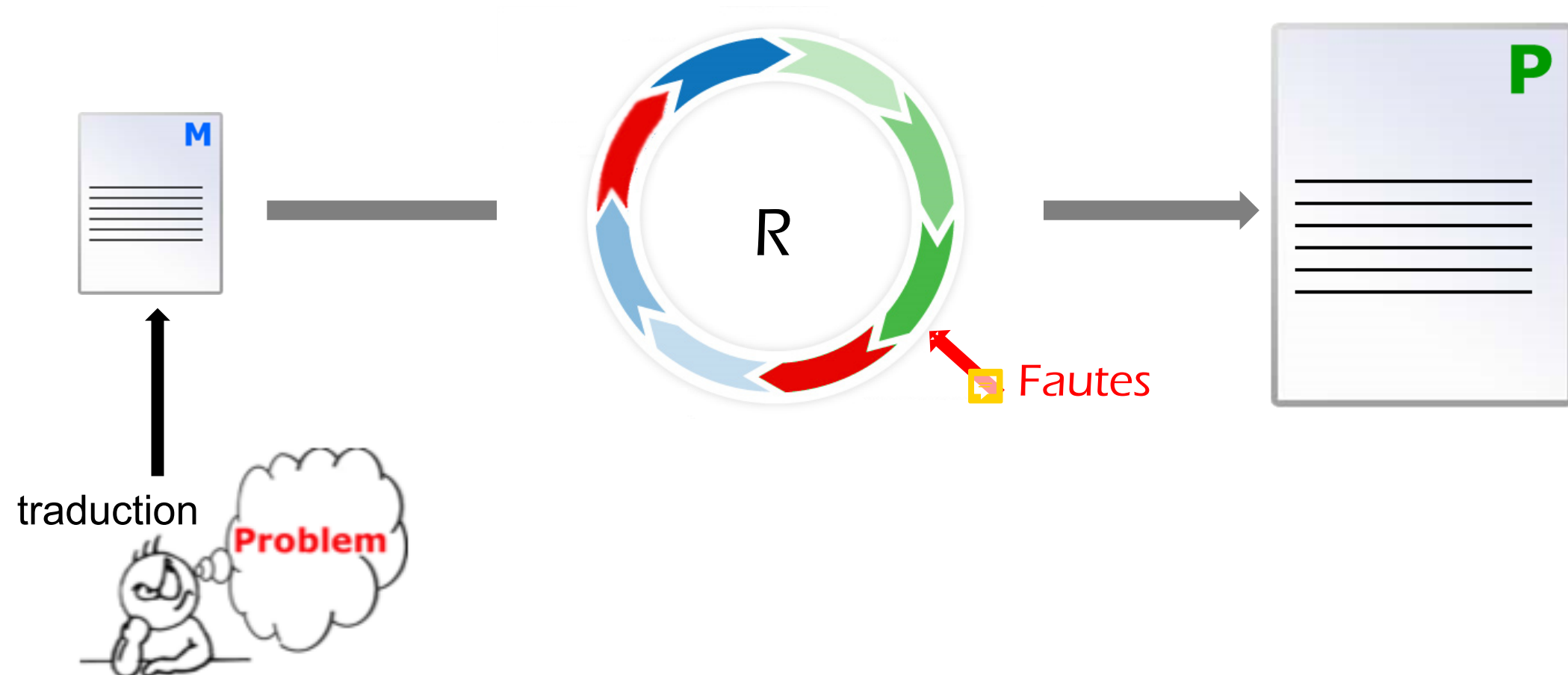
[Lazaar et al, 2012]



Testing AI-based systems

CPTTEST

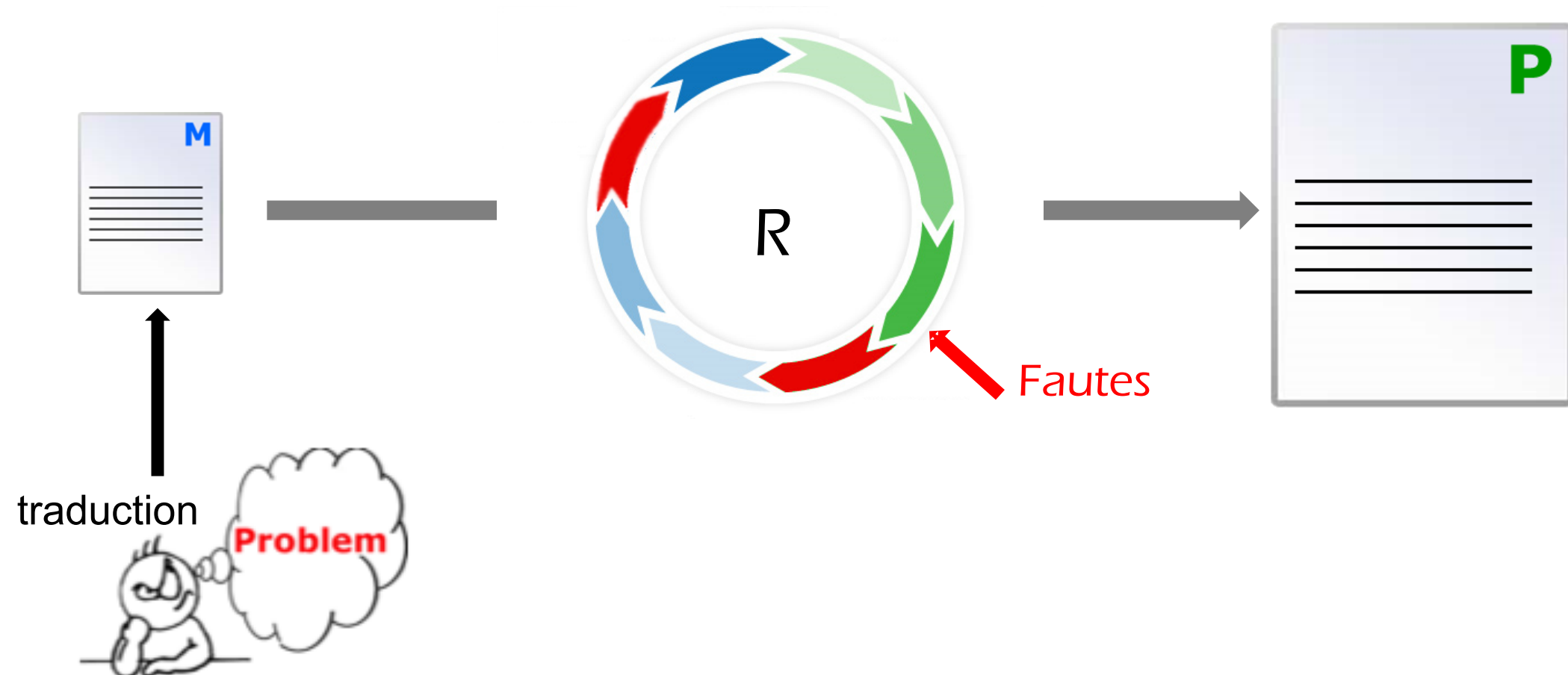
[Lazaar et al, 2012]



Testing AI-based systems

CPTTEST

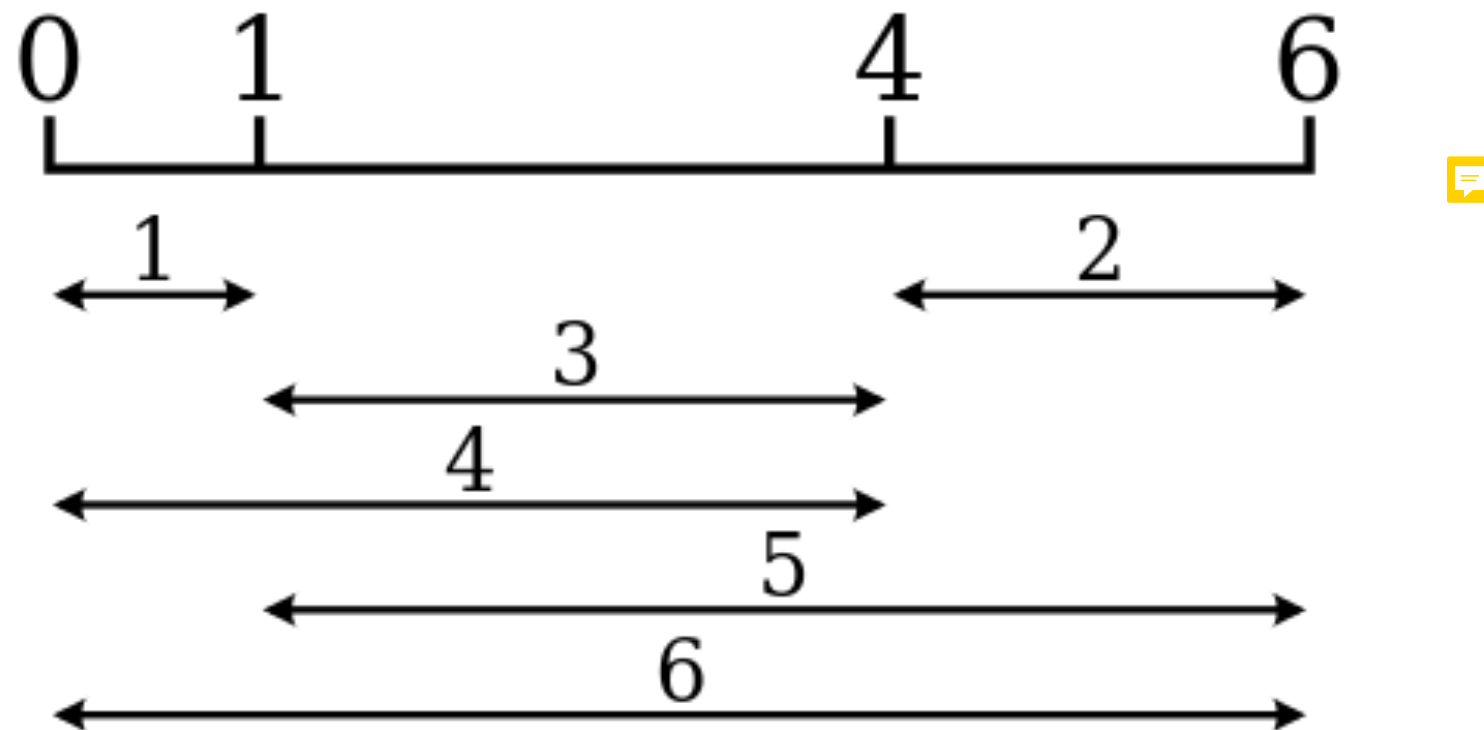
[Lazaar et al, 2012]



CPTEST

Example (Golomb ruler)

- a **Golomb ruler** is a set of marks at integer positions along a ruler such that no two pairs of marks are the same distance apart.



CPTTEST

Example (Golomb ruler)

(A)

```
using CP;

int m=...;

dvar int x[1..m] in 0..m *m;

minimize x[m];

subject to{

c1: forall(i in 1..m - 1)
    x[i] < x[i+1] ;

c2: forall(ordered i,j in 1..m)
    forall(ordered k, l in 1..m:
        (i!=k || j!=l))
        !((x[j]-x[i]) == (x[l]-x[k]));
}
```

(B)

```
using CP;

int m=...;
int nbDist= m*(m-1)div 2;

dvar int x[1..m] in 0..m*m;
dvar int d[1..nbDist];

minimize d[m-1];

subject to {

cc1: forall (i in 1..m-1)
    x[i] < x[i+1];

cc2: forall(ordered i,j in 1..m)
    d[(nbDist-((m-i+1)*(m-i)div 2)+(j-i))]
        == x[j]-x[i];
//cc2': forall(ordered i,j in 1..nbDist div m)
//    d[nbDist-(j-i)] == x[j]-x[i];
cc3: x[1]==0;

cc4: x[2] <= d[nbDist];

cc5: x[m] >= nbDist;

cc6: allDifferent(d) ;

cc7: forall(ordered i,j in 2..m, k in 1..m*m)
    x[i]==x[i-1]+k => x[j]!=x[j-1]+k;
}
```

faute

Testing AI-based systems

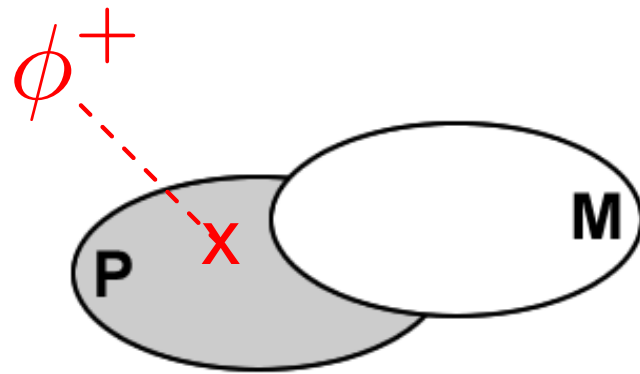
CPTTEST

[Lazaar et al, 2012]

Testing AI-based systems

CPTTEST

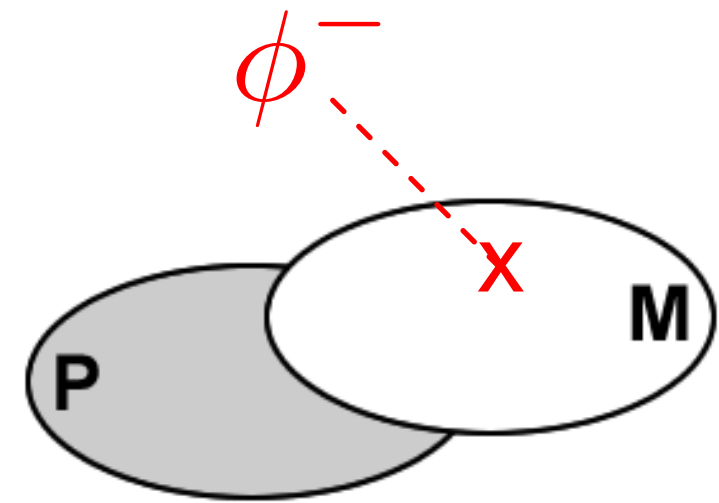
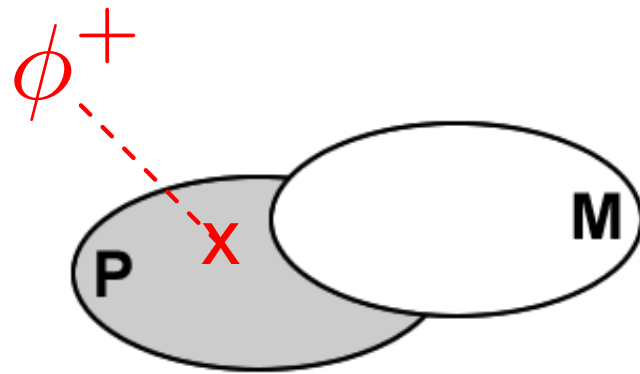
[Lazaar et al, 2012]



Testing AI-based systems

CPTTEST

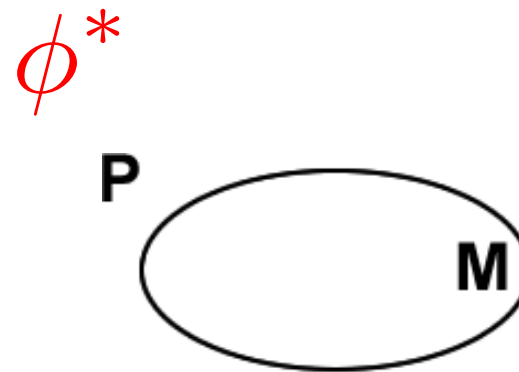
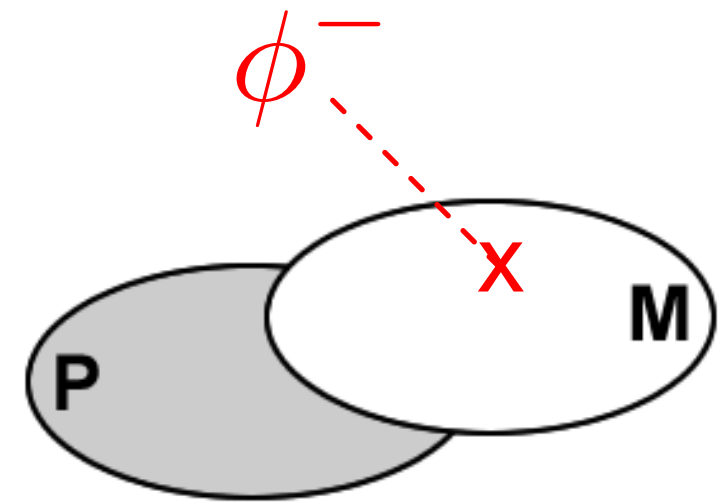
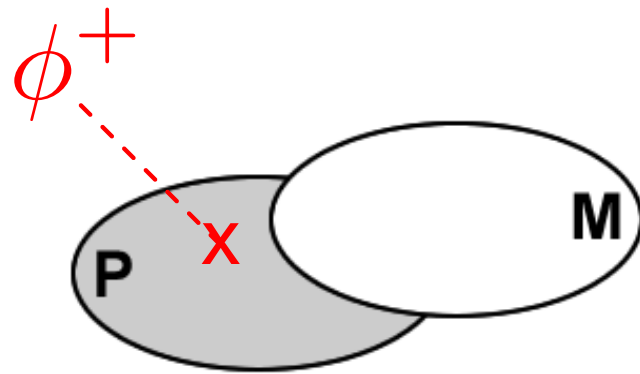
[Lazaar et al, 2012]



Testing AI-based systems

CPTTEST

[Lazaar et al, 2012]

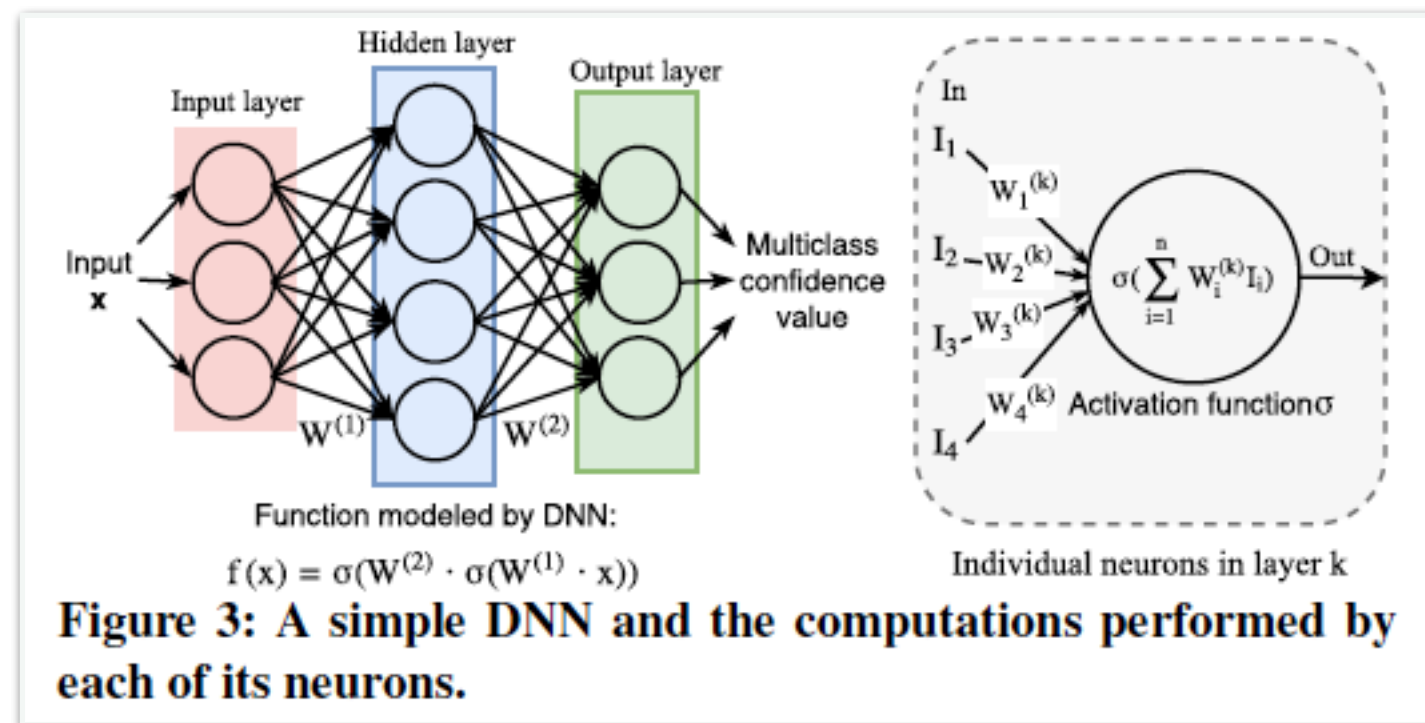


Testing AI-based systems

DeepXplore

[Pei et al, 2017]

- Automated Whitebox Testing of Deep Learning Systems
- Neuron coverage



Testing AI-based systems

DeepTest

[Tian et al, 2018]

- Applies image transformations such as contrast, scaling, blurring to generate synthetic test images



1.1 original



1.2 with added rain

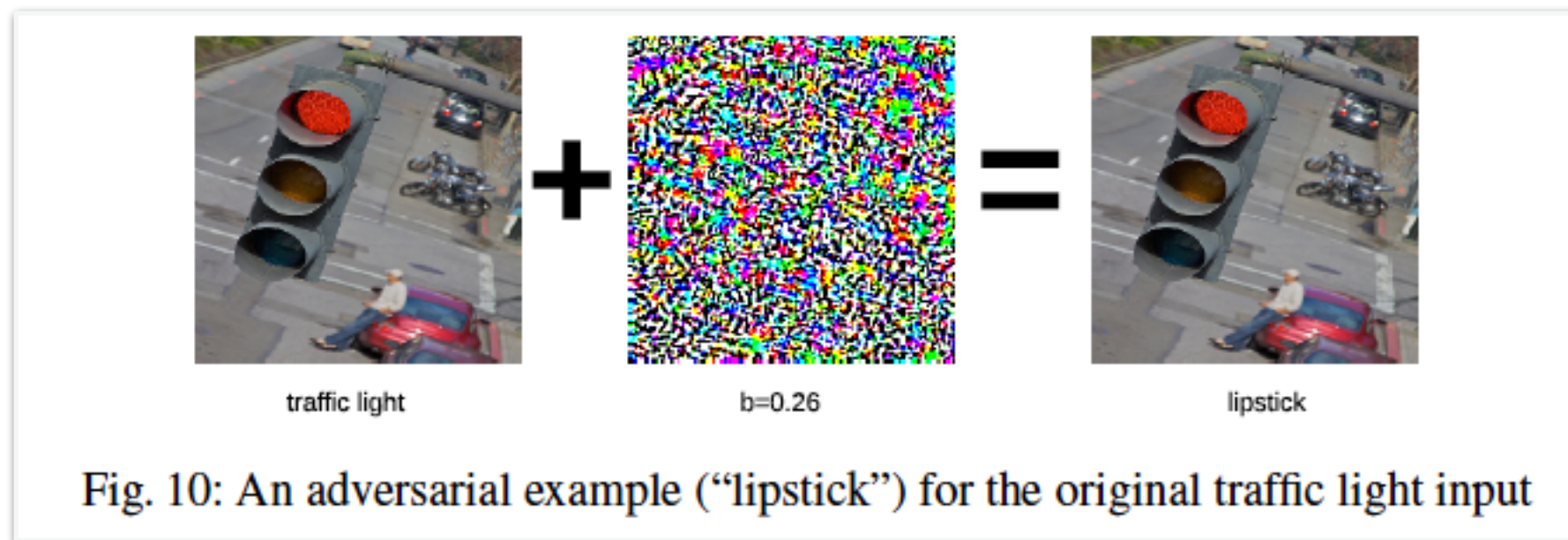
Figure 1: A sample dangerous erroneous behavior found by DeepTest in the *Chauffeur* DNN.

Testing AI-based systems

DeepCover

[Sun et al, 2018]

- Adaptation of combinatorial testing techniques for the systematic sampling of a large space of neuron interactions



Many

Thanks to

- Dusica Marijan, Arnaud Gotlieb, SIMULA Research Lab., Oslo, Norway