ENG Case Study

Keven T. Kearney

ENGINEERING

D4.4.3 “Assurance of the ENG Case Study”

Envisage Year 3 Plenary
Oslo, June. 29, 2016

SEVENTH FRAMEWORK PROGRAMME

ETICS - Assurance

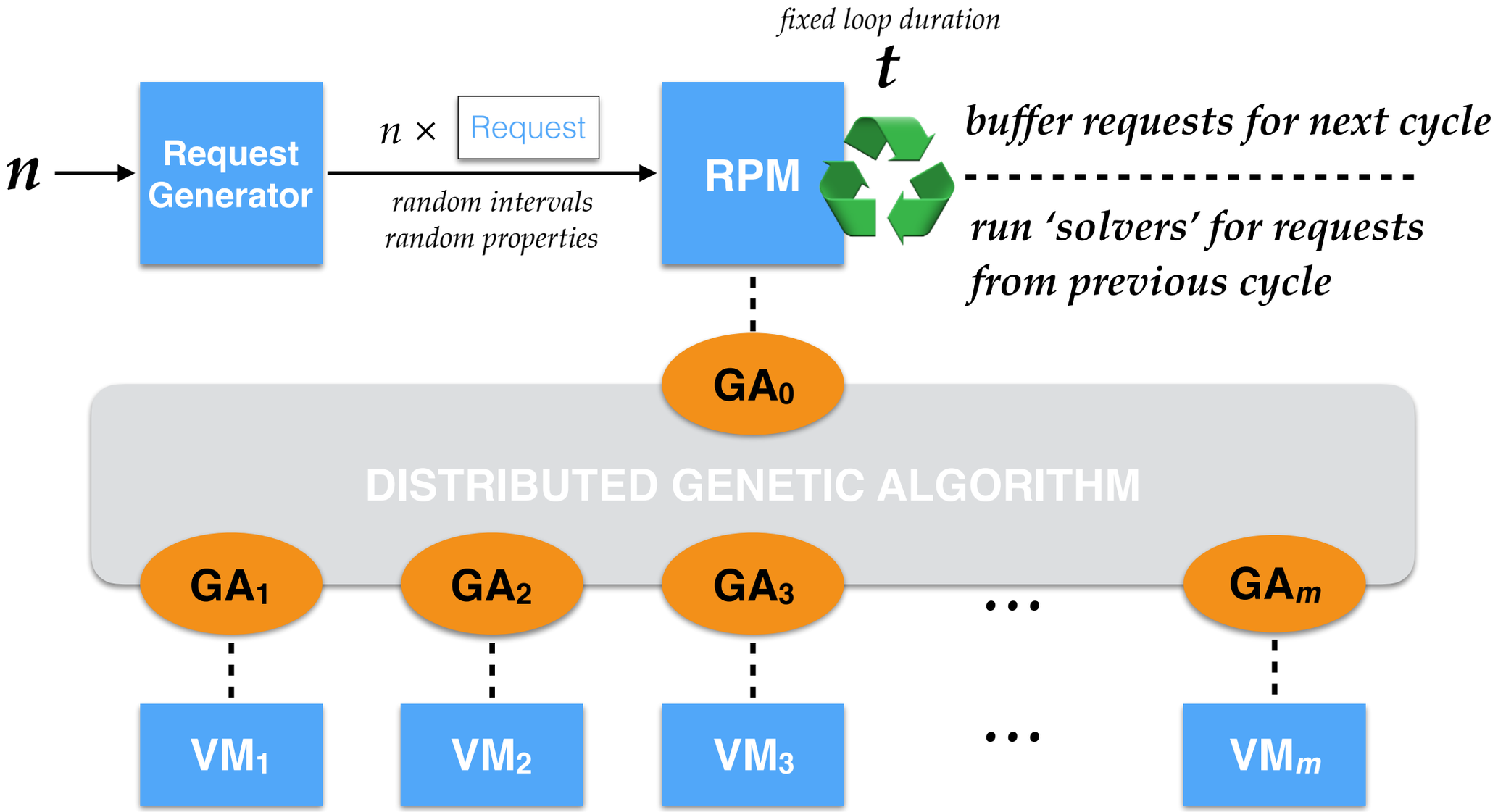
- **Code Simulation (Erlang)**
- **Deadlock Analysis**
- **Resource Analysis**
- **Java Code Generation**

ETICS - Assurance

- **Code Simulation (Erlang)**
- **Deadlock Analysis**
- **Resource Analysis**
- **Java Code Generation**

1: Code Simulation (Erlang)

- Recap of ETICS Simulator:



1: Code Simulation (Erlang)

(Mac OS 10.11 - Terminal App)

- **Full Code** (1672 lines of code)
  - 1..4 simulated requests → OK
  - 5+ → Stalls
  - 1000 → Error `{"init terminating in do_boot",badrpc}`

1: Code Simulation (Erlang)

- **output from 4 requests**

Launch VM#1 →
Assign Request#1 to VM#1 →
Launch VM#3 →
Assign Request#3 to VM#3 →
Launch VM#2 →
Assign Request#4 to VM#2 →
Assign Request#2 to VM#2 →
Kill VM#1 →
Kill VM#3 →
Kill VM#2 →

```
START
RECEIVED:RQ(id=1,size=117071231/5000000,pri=1/2)
RECEIVED:RQ(id=2,size=146926711/5000000,pri=1)
RECEIVED:RQ(id=3,size=14985807/2000000,pri=1)
END
RECEIVED:RQ(id=4,size=3798911/250000,pri=1)
LAUNCHED:VM(id=1,power=74/5,disk=32)
DEPLOYED:id=1
ENQUEUED:VM(id=1),RQ(id=1),D(dur=-1)
SETTING_UP:VM(id=1),RQ(id=1),D(dur=1287783541/1480000000)
LAUNCHED:VM(id=3,power=44/5,disk=8)
EXECUTING:VM(id=1),RQ(id=1),D(dur=1287783541/740000000)
INCOME=351213693/40000000
UTILITY=351213693/40000000
CLEANING_UP:VM(id=1),RQ(id=1),D(dur=1287783541/1480000000)
DEPLOYED:id=3
ENQUEUED:VM(id=3),RQ(id=3),D(dur=-1)
SETTING_UP:VM(id=3),RQ(id=3),D(dur=14985807/32000000)
LAUNCHED:VM(id=2,power=72/5,disk=32)
EXECUTING:VM(id=3),RQ(id=3),D(dur=14985807/16000000)
FINISHED:VM(id=1),RQ(id=1)
INCOME=501071763/40000000
UTILITY=501071763/40000000
CT_PENALTIES=176761942682583/176000000000000
UTILITY=2027953814517417/176000000000000
CLEANING_UP:VM(id=3),RQ(id=3),D(dur=14985807/32000000)
FINISHED:VM(id=3),RQ(id=3)
DEPLOYED:id=2
ENQUEUED:VM(id=2),RQ(id=4),D(dur=-1)
ENQUEUED:VM(id=2),RQ(id=2),D(dur=20059265831/480000000)
SETTING_UP:VM(id=2),RQ(id=4),D(dur=41788021/72000000)
EXECUTING:VM(id=2),RQ(id=4),D(dur=41788021/36000000)
INCOME=804984643/40000000
UTILITY=3365170486517417/176000000000000
CLEANING_UP:VM(id=2),RQ(id=4),D(dur=41788021/72000000)
FINISHED:VM(id=2),RQ(id=4)
SETTING_UP:VM(id=2),RQ(id=2),D(dur=1616193821/1440000000)
EXECUTING:VM(id=2),RQ(id=2),D(dur=1616193821/720000000)
INCOME=1686544909/40000000
UTILITY=7244035656917417/176000000000000
CLEANING_UP:VM(id=2),RQ(id=2),D(dur=1616193821/1440000000)
FINISHED:VM(id=2),RQ(id=2)
VM_COST=1104831/125000
UTILITY=5688433608917417/176000000000000
KILLED:id=1
VM_COST=1523781/125000
UTILITY=5098552008917417/176000000000000
KILLED:id=3
VM_COST=2819733/125000
UTILITY=3273851592917417/176000000000000
KILLED:id=2
```

€18.60

1: Code Simulation (Erlang)

(Mac OS 10.11 - Terminal App)

- **Full Code** (1672 lines of code)
  - 1..4 simulated requests → OK
  - 5+ → Stalls
  - 1000 → Error `{"init terminating in do_boot",badrpc}`

- **Skeleton Code** (333 lines) - for **Deadlock Analysis**
  - Removed data types + data processing
    - ONLY distributed/asynchronous control flow
  - 1..100 simulated requests → OK
  - 100+ → Stalls

ETICS - Assurance

- Code Simulation (Erlang)
- **Deadlock Analysis**
- **Resource Analysis**
- **Java Code Generation**

ETICS - Assurance

Tools → Collaboratory

- http://localhost:8888 (Vagrant VM / VirtualBox)
  - Empty examples
  - Multifile models?
  - Saving files?
  - Erlang simulator: no statistics

- http://ei.abs-models.org:8082/clients/web
  - Multifile models?
  - Saving files?

ETICS - Assurance

- Code Simulation (Erlang)
- **Deadlock Analysis**
- Resource Analysis
- Java Code Generation

2: Deadlock Analysis

- **Deadlock Problem**
  - Swift (v2) prototype stalls (zero CPU usage)

    Tests
    - Full code
    - “Inheritance Test” code
    - Skeleton code

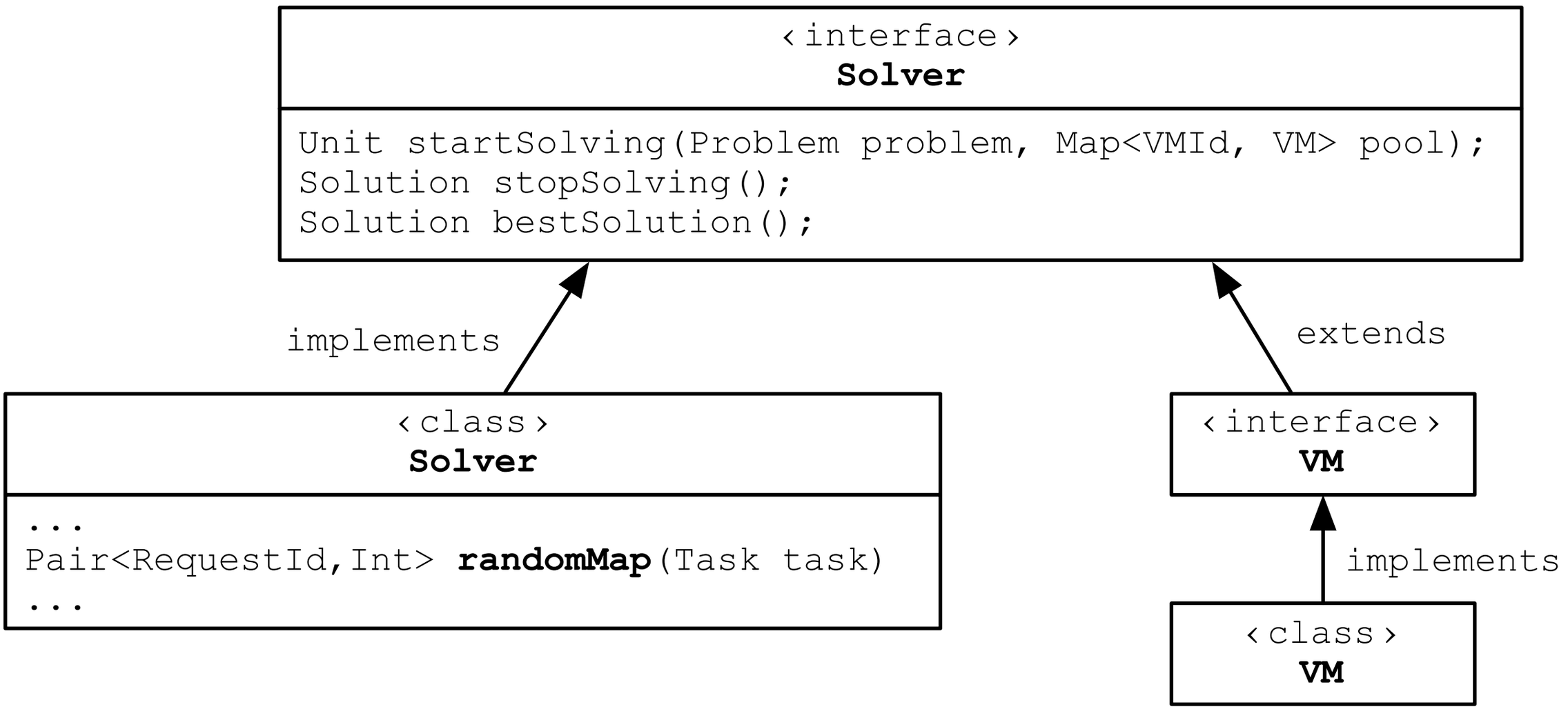
2: Deadlock Analysis

- **Full Code**

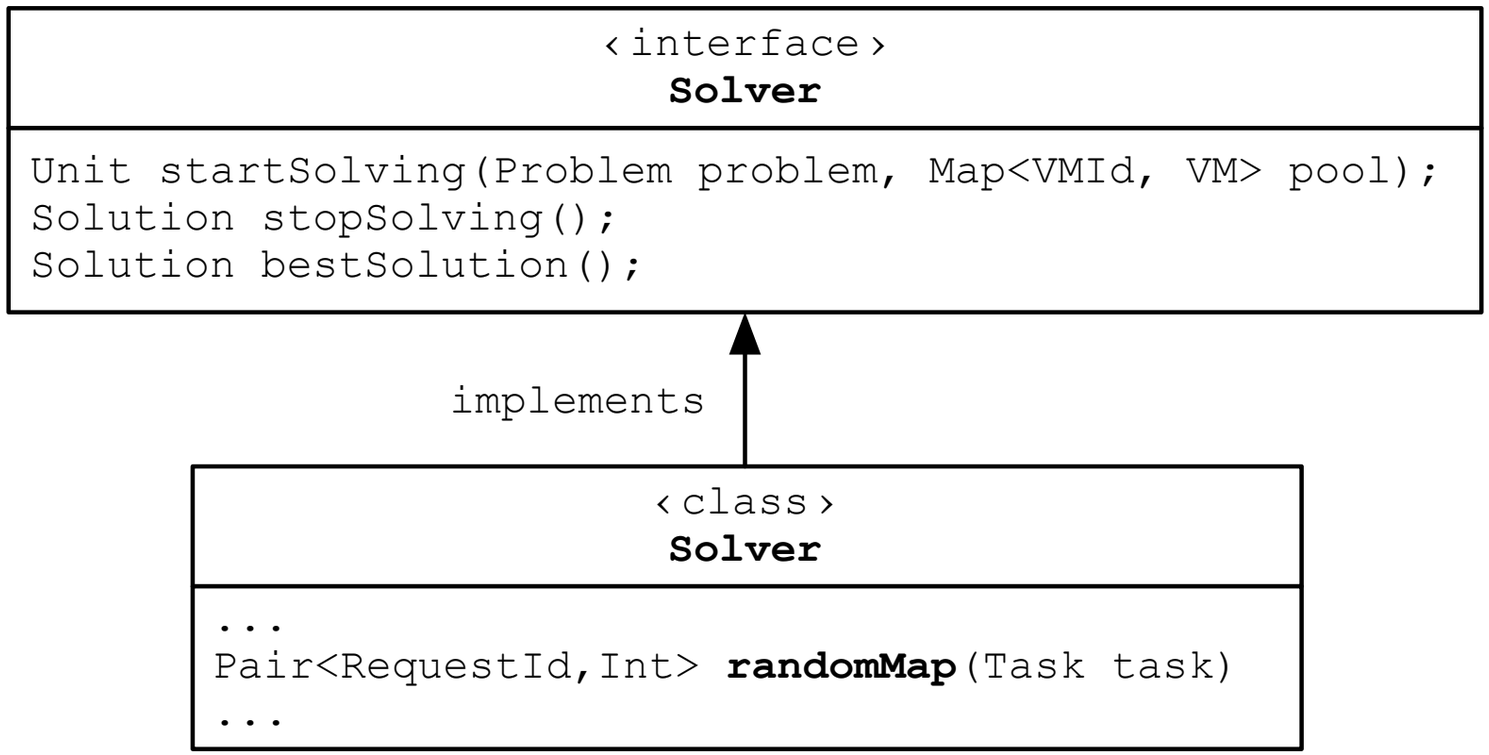
	SACO	DSA
local VM	**Error** `inexistent_entry(VM.randomMap)`	**no output**

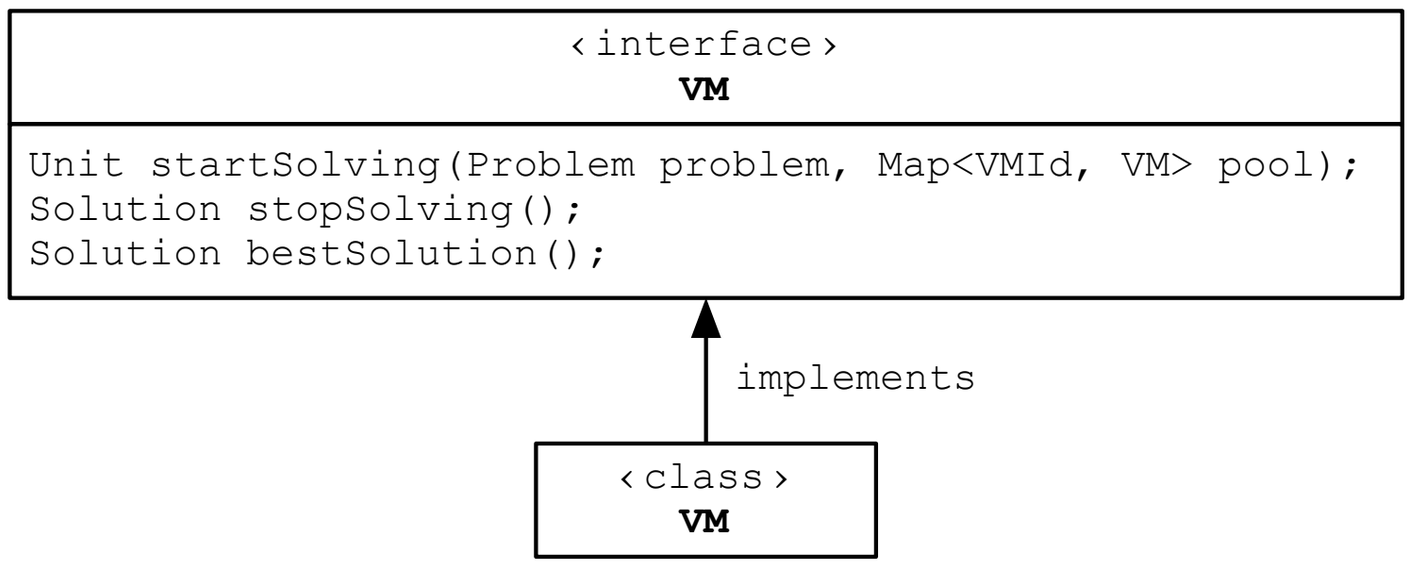
online	**Error**
	`inexistent_entry(VM.randomMap)`
		`checking with ..`
		`checking with ..`
		`checking with "ABS.Scheduler. ..`

2: Deadlock Analysis



2: Deadlock Analysis





2: Deadlock Analysis

- **Full Code**, without inheritance

	SACO	**DSA**
local VM	**Deadlock Free**	**no output**

online	**Deadlock Free**	`checking with ..`
		`checking with ..`
		`checking with "ABS.Scheduler. ..`

2: Deadlock Analysis

```
module XX;

interface X{
  Unit foo();
}

class X() implements X{
  Unit foo(){
    this.goo();
  }
  Unit goo(){ /* do nothing */ }
}

interface Y extends X{}

class Y() implements Y{
  X x;
  { // init
    x = new local X();
  }
  Unit foo(){
    x.foo();
  }
}
// main
{
  Y y = new Y();
  y.foo();
}
```

2: Deadlock Analysis

- **Inheritance Test Code**, with inheritance

	SACO	**DSA**
local VM	**Error** `inexistent_entry(Y.goo)`	**no output**
online	**Error** `inexistent_entry(Y.goo)`	**Deadlock Free**

2: Deadlock Analysis

```
module XX;

interface X{
  Unit foo();
}

class X() implements X{
  Unit foo(){
    this.goo();
  }
  Unit goo(){ /* do nothing */ }
}

interface Y extends X{}

class Y() implements Y{
  X x;
  { // init
    x = new local X();
  }
  Unit foo(){
    x.foo();
  }
}
// main
{
  Y y = new Y();
  // y.foo();
}
```

Don't call `y.foo()`

2: Deadlock Analysis

- **Inheritance Test Code**, with inheritance, `//y.foo()`

	SACO	DSA
local VM	**Deadlock Free**	**no output**
online	**Deadlock Free**	**Deadlock Free** **?**

2: Deadlock Analysis

- **Skeleton Code**, no inheritance

	SACO	**DSA**
local VM	**Deadlock Free**	**no output**

online	**Deadlock Free**	`checking with ..`
		`checking with ..`
		`checking with "ABS.Scheduler. ..`

2: Deadlock Analysis

- **Deadlock Problem**
  - Swift (2.0) prototype stalls (zero CPU usage)
  - ABS Model = deadlock free
    - No blocking ".get" expressions

```
X x = await obj!x();
```

    - **Is the ABS model a "good" model of the Swift version?**
- Swift (2.2) prototype **does not stall**

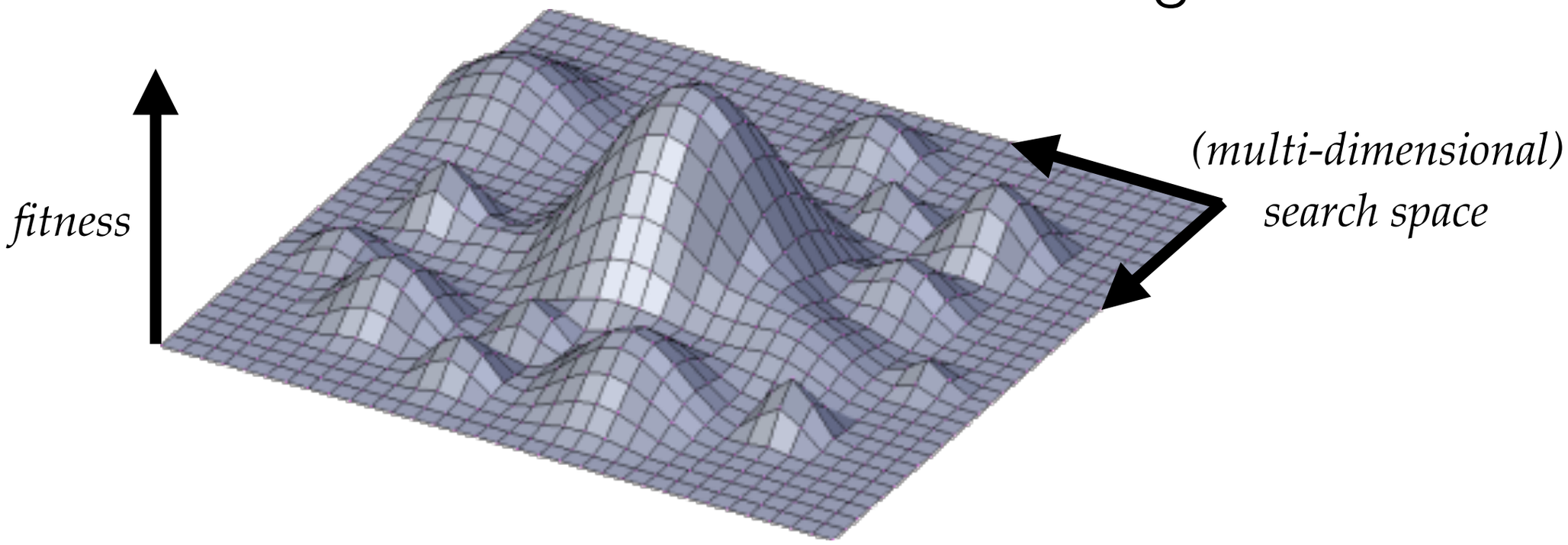
ETICS - Assurance

- Code Simulation (Erlang)
- Deadlock Analysis
- **Resource Analysis**
- Java Code Generation

3: Resource Analysis

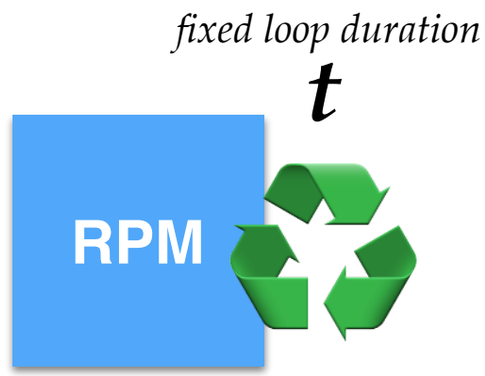
- **Problem**
  - Does the DGA scale (1000000s of requests/VMs)?
    - n = number of requests (in one RPM cycle)
    - m = number of VMs = $m_a + n$ [where m_a = number of VMs in pool]
    - s_p = number of possible solutions (exponential with $n + m$)
    - s_x = number of **tested** solutions?
    - How does s_x/s_p ("coverage") change with n and m

scales if: High% + Linear



3: Resource Analysis

- **Problem**
  - Does the DGA scale?
    - s_x = number of **tested** solutions = $f(m_a)$
    - For a single **`Solver`** instance:
      - find: g = number of GA generations per t
        - p = number of solutions per generation = FIXED



    - UB(s_x) = $gp(m_a + 1)$ (UB since some/many solutions are repeated)
    - To find g:

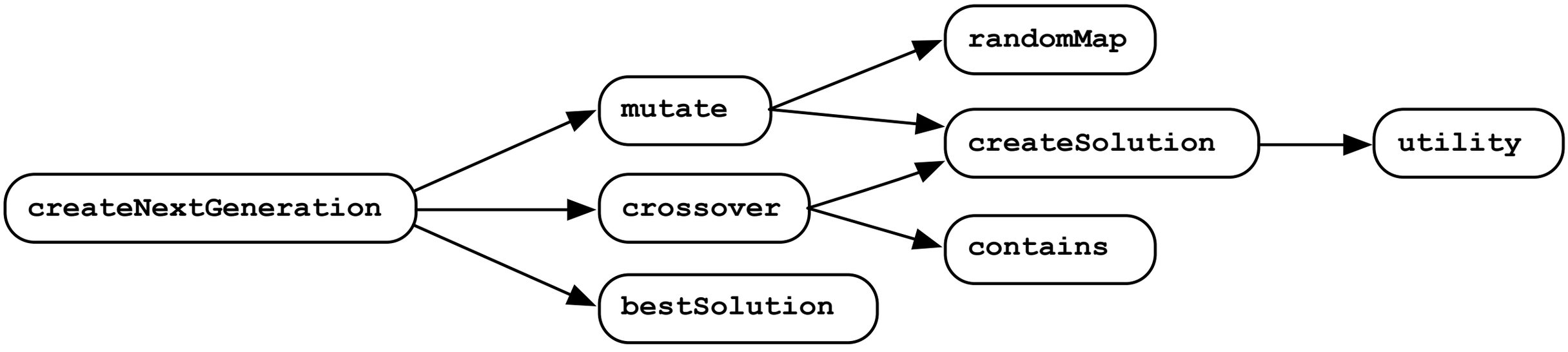
      How long does it take to create & test a single generation?

      ⇒ *computational cost* (steps) of the **`Solver`** method:

      `createNextGeneration(...)`

3: Resource Analysis

- Method invocations



- Lots of **`random()`** calls
- But, they are not "in principle" a problem .. e.g.

```
Int i = 0;
Int x = random_int(n);
while (i < x){ result[i] = a[i]; i++; }
while (i < n){ result[i] = b[i]; i++; }
```

3: Resource Analysis

```
List<Solution> createNextGeneration(List<Solution> previous, Int pop_count, Map<VMId, VM> pool){
    List<VM> vms = values(pool);
    Int vm_count = length( vms );
    Int top_count = ceiling(pop_count / 20);
    List<Solution> next_generation = Nil;
    Int i = 0;
    while (!cancelled && i < pop_count){
        Solution solution = nth(previous, random(top_count));
        Rat f = randomf();
        if (f < 1/4 || vm_count == 0){
            solution = this.mutate(solution);
        }else if (f < 1/2){
            Solution another = nth(previous, random(top_count));
            solution = this.crossover(solution, another);
        }else if (f < 3/4){
            // note: vm_count > 0 (see 1st case above)
            VM vm = nth(vms, random(vm_count));
            if (vm != null){
                solution = await vm!bestSolution();
            }
        }
        next_generation = Cons(solution, next_generation);
        i = i + 1;
    }
    return next_generation;
}
```

number of individuals per generation

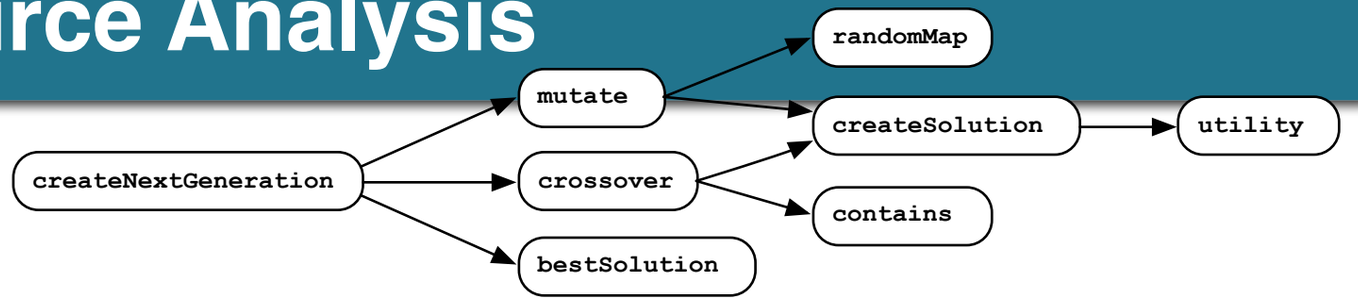
depend on size of solution(s) = calculable

problem

depends on
- VM activity
- comm.s latency

```
Solution best = NoSolution;
...
Solution bestSolution(){
  return best;
}
```

3: Resource Analysis



- **SACO** results (default settings)

  Same error for **`VM extends Solver`** so, using no inheritance version.

  `createNextGeneration(…).`

```
19+5*nat(pool/4-1/4)+6*nat(pool/2-1/2)+c(failed(no_rf,[scc=129,cr=entrywhile_0/9]))
```

  `mutate(…).`

```
45+6*nat(solution/2-3)+nat(solution/2-3)* (69+4*nat(solution/2-4)+6*nat(solution/
2-11/2)+c(failed(no_rf,[scc=104,cr=entrywhile_0/8]))+5*c(maximize_failed))
+6*c(maximize_failed)+c(maximize_failed)* (80+9*c(maximize_failed)+4*nat(solution/
2-4)+6*nat(solution/2-11/2)+9*c(maximize_failed)+4*c(maximize_failed)+6*c(maximize_failed))
+c(maximize_failed)* (12+8*c(maximize_failed))+c(failed(no_rf,[scc=92,cr=entrywhile_2/10]))
```

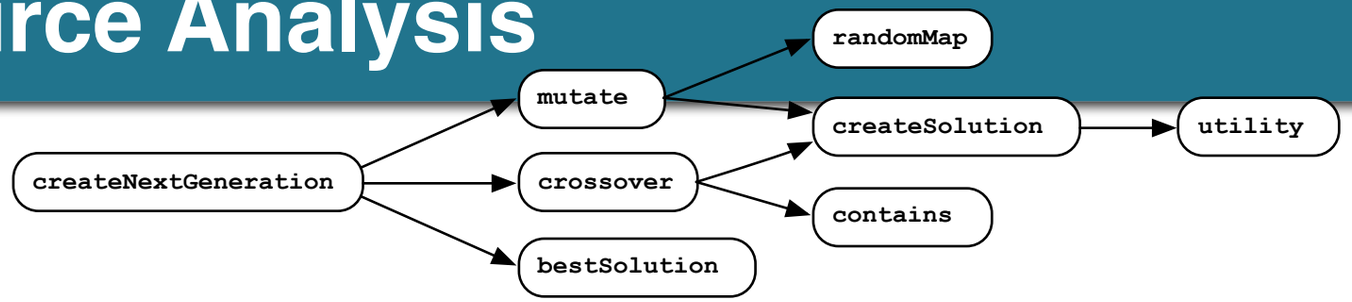
  `crossover(…).`

```
52+6*nat(s1/2-3)+c(maximize_failed)* (21+5*c(maximize_failed))+nat(s2/2-9/2)*
(37+20*c(maximize_failed)+5*c(maximize_failed))+6*c(maximize_failed)+c(maximize_failed)*
(80+9*c(maximize_failed)+4*nat(s1/2-4)+6*nat(s1/2-11/2)+9*c(maximize_failed)
+4*c(maximize_failed)+6*c(maximize_failed))+c(maximize_failed)* (12+8*c(maximize_failed))
+c(failed(no_rf,[scc=92,cr=entrywhile_1/10]))
```

  `bestSolution(…).`

```
Method Solver.bestSolution terminates?: YES
UB for 'Solver.bestSolution'(this) = 2
```

3: Resource Analysis



- **SACO** results

  continued …

  randomMap(…).

  ```
  26+6*nat(task/2-3/2)+c(failed(no_rf,[scc=4,cr=entrywhile_0/8]))
  ```

  createSolution(…).

  ```
  22+6*nat(maps/2-1/2)+nat(maps/2-1/2)* (80+9*nat(maps/2-5/2)+4*nat(problem/
  2-3/2)+6*nat(problem/2-3)+9*nat(maps-4)+4*c(maximize_failed)+6*c(maximize_failed))
  +c(maximize_failed)* (12+8*c(maximize_failed))+c(failed(no_rf,[scc=92,cr=entrywhile_1/10]))
  ```

  utility(…).

  ```
  35+243*nat(requests/2-13/2)
  ```

  contains(…).

  ```
  13+20*nat(list/2-1/2)
  ```

3: Resource Analysis

- **SACO** results

```
Pair<RequestId,Int> randomMap(Task task){
    Request request = taskRequest(task);
    RequestId request_uuid = requestId(request);
    List<VMInfo> vm_info_list = taskVMInfoList(task);
    Int n = length(vm_info_list);
    Int i = 0;
    Int selected = -1;
    while (selected < 0 && i < n){
        if (randomf() > (1/4)){
            selected = i;
        }
        i = i + 1;
    }
    if (selected < 0){
       selected = random(n + 1);
    }
    return Pair(request_uuid, selected);
}
```

3: Resource Analysis

- **SACO** results

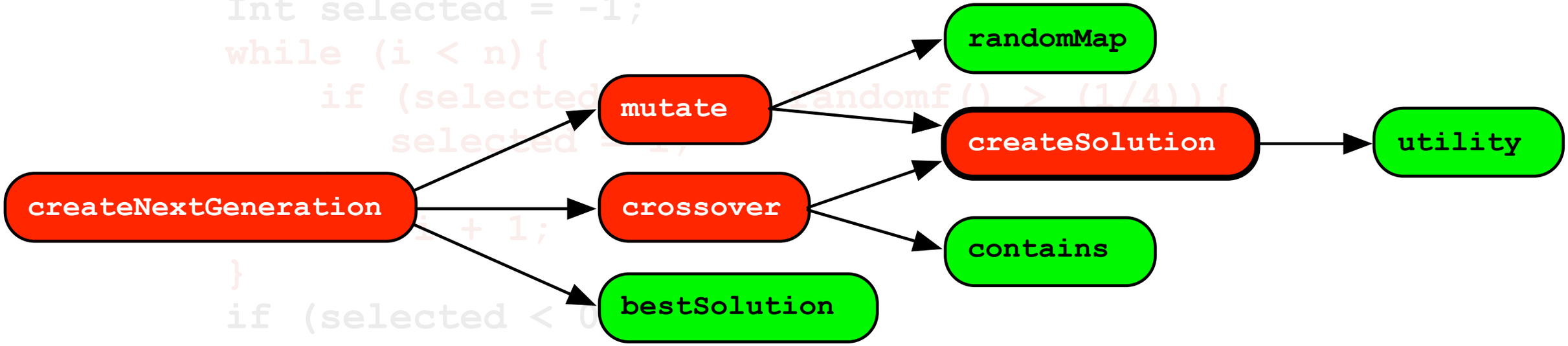
```
Pair<RequestId,Int> randomMap(Task task){
    Request request = taskRequest(task);
    RequestId request_uuid = requestId(request);
    List<VMInfo> vm_info_list = taskVMInfoList(task);
    Int n = length(vm_info_list);
    Int i = 0;
    Int selected = -1;
    while (i < n){
        if (selected < 0 && randomf() > (1/4)){
            selected = i;
        }
        i = i + 1;
    }
    if (selected < 0){
       selected = random(n + 1);
    }
    return Pair(request_uuid, selected);
}
```

3: Resource Analysis

- **SACO** results

randomMap()

```
20+6*nat(task/2-3/2)+5*nat(task/2-3/2)
```



3: Resource Analysis

- **SACO** results

```
Solution createSolution(Problem problem, List<Pair<RequestId,Int>> maps){
    ...
    List<Request> rejected = Nil;
    Map<VMId, Pair<VMInfo, List<Request>>> assigned = EmptyMap;
    Rat utility = 0;
    Int n = length(maps);
    Int i = 0;
    while (i < n){
        /* code to build rejected & assigned */
        i = i + 1;
    }
    Set<VMId> vm_ids = keys(assigned);
    while (hasNext(vm_ids)){
        VMId vm_id = take(vm_ids);
        /* code to calculate utility */
        vm_ids = remove(vm_ids, vm_id);
    }
    return Solution(problem, rejected, assigned, Price(utility), maps);
}
```

```
"... c(failed(no_rf,[scc=8,cr=entrywhile_1/10]))"
```

3: Resource Analysis

- **SACO** results

```
Solution createSolution(Problem problem, List<Pair<RequestId,Int>> maps){
    ...
    List<Request> rejected = Nil;
    Map<VMId, Pair<VMInfo, List<Request>>> assigned = EmptyMap;
    Rat utility = 0;
    Int n = length(maps);
    Int i = 0;
    while (i < n){
        /* code to build rejected & assigned */
        i = i + 1;
    }
    Set<VMId> vm_ids = keys(assigned);
    n = size(vm_ids);
    i = 0;
    while (i < n){
        VMId vm_id = take(vm_ids);
        /* code to calculate utility */
        vm_ids = remove(vm_ids, vm_id);
        i = i + 1;
    }
    return Solution(problem, rejected, assigned, Price(utility), maps);
}
```

3 × "c(maximise_failed)"

3: Resource Analysis

- **SACO** results

```
Solution createSolution(Problem problem, List<Pair<RequestId,Int>> maps){
    ...
    List<Request> rejected = Nil;
    Map<VMId, Pair<VMInfo, List<Request>>> assigned = EmptyMap;
    Rat utility = 0;
    Int n = length(maps);
    Int i = 0;
    while (i < n){
        /* code to build rejected & assigned */
        i = i + 1;
    }
    Set<VMId> vm_ids = keys(assigned);
    // n = size(vm_ids);
    i = 0;
    while (i < n){
        if (hasNext(vm_ids)){
            VMId vm_id = take(vm_ids);
            /* code to calculate utility */
            vm_ids = remove(vm_ids, vm_id);
        }
        i = i + 1;
    }
    return Solution(problem, rejected, assigned, Price(utility), maps);
}
```

1 × "c(maximise_failed)"

3: Resource Analysis

- **SACO** results

```
Solution createSolution(Problem problem, List<Pair<RequestId,Int>> maps){
    ...
    List<Request> rejected = Nil;
    Map<VMId, Pair<VMInfo, List<Request>>> assigned = EmptyMap;
    Rat utility = 0;
    Int n = length(maps);
    Int i = 0;
    List<VMId> vm_ids = Nil;
    while (i < n){
        /* code to build rejected & assigned & vm_ids */
        i = i + 1;
    }
    while (vm_ids != Nil){
        VMId vm_id = head(vm_ids);
        /* code to calculate utility */
        vm_ids = tail(vm_ids);
    }
    return Solution(problem, rejected, assigned, Price(utility), maps);
}
```

22+6*nat(maps/2-1/2)+5*nat(maps/2-1/2)

3: Resource Analysis

- **SACO** results

```
Solution createSolution(Problem problem, List<Pair<RequestId,Int>> maps){
    ...
    List<Request> rejected = Nil;
    Map<VMId, Pair<VMInfo, List<Request>>> assigned = EmptyMap;
    Rat utility = 0;
    Int n = length(maps);
    Int i = 0;
    List<VMId> vm_ids = Nil;
    while (i < n){
        /* code to build rejected & assigned & vm_ids */
        i = i + 1;
    }
    while (vm_ids != Nil){
        VMId vm_id = head(vm_ids);
        <<- UNCOMMENT: code to calculate utility
        vm_ids = tail(vm_ids);
    }
    return Solution(problem, rejected, assigned, Price(utility), maps);
}
```

22+6*nat(maps/2-1/2)+5*nat(maps/2-1/2)

3: Resource Analysis

- **SACO** results

```
Solution createSolution(Problem problem, List<Pair<RequestId,Int>> maps){
    ...
    List<Request> rejected = Nil;
    Map<VMId, Pair<VMInfo, List<Request>>> assigned = EmptyMap;
    Rat utility = 0;
    Int n = length(maps);
    Int i = 0;
    List<VMId> vm_ids = Nil;
    while (i < n){
        <<- UNCOMMENT: code to build rejected & assigned & vm_ids
        i = i + 1;
    }
    while (vm_ids != Nil){
        VMId vm_id = head(vm_ids);
        <<- UNCOMMENT: code to calculate utility
        vm_ids = tail(vm_ids);
    }
    return Solution(problem, rejected, assigned, Price(utility), maps);
}
```

8 × "c(maximise_failed)"

3: Resource Analysis

```
Solution createSolution(Problem problem, List<Pair<RequestId,Int>> maps){
   Map<RequestId, Task> task_map = problemTaskMap(problem);
   List<Request> rejected = Nil;
   Map<VMId, Pair<VMInfo, List<Request>>> assigned = EmptyMap;
   Rat utility = 0;
   List<VMId> vm_ids = Nil;
   Int n = length(maps);
   Int i = 0;
   while (i < n){
      Pair<RequestId,Int> map = nth(maps, i);
      RequestId request_id = fst(map);
      Int vmi_index = snd(map);
      Maybe<Task> maybe_task = lookup(task_map, request_id);
      if (maybe_task != Nothing){
         Task task = fromJust(maybe_task);
         Request request = taskRequest(task);
         List<VMInfo> vmis = taskVMInfoList(task);
         if (vmi_index >= length(vmis)){
            rejected = Cons(request, rejected);
            Rat penalty_fr = priceValue(requestPenaltyFR(request));
            Pair<VMInfo, List<Request>> p = fromJust(maybe_assigned);
            requests = snd(p);
         }
         requests = Cons(request, requests);
         assigned = put(assigned, vm_id, Pair(vm_info, requests));
         vm_ids = Cons(vm_id, vm_ids);
      }
   }
   i = i + 1;
}
while (vm_ids != Nil){
   VMId vm_id = head(vm_ids);
   Maybe<Pair<VMInfo, List<Request>>> maybe_assigned = lookup(assigned, vm_id);
   if (maybe_assigned != Nothing){
      Pair<VMInfo, List<Request>> p = fromJust(maybe_assigned);
      VMInfo vm_info = fst(p);
      List<Request> requests = snd(p);
      Triple<Rat, List<Request>, List<Request>> t = this.utility(vm_info, requests);
      utility = utility + fstT(t);
      List<Request> accepted = sndT(t);
      List<Request> rejects = trd(t);
      if (length(rejects) > 0){
         assigned = put(assigned, vm_id, Pair(vm_info, accepted));
         while (rejects != Nil){
            rejected = Cons(head(rejects), rejected);
            rejects = tail(rejects);
         }
      }
   }
   vm_ids = tail(vm_ids);
}
return Solution(problem, rejected, assigned, Price(utility), maps);
}
```

```
assigned = put(assigned, vm_id, Pair(vm_info, requests));
vm_ids = Cons(vm_id, vm_ids);
```

3: Resource Analysis

```
Solution createSolution(Problem problem, List<Pair<RequestId,Int>> maps){
   Map<RequestId, Task> task_map = problemTaskMap(problem);
   List<Request> rejected = Nil;
   Map<VMId, Pair<VMInfo, List<Request>>> assigned = EmptyMap;
   Rat utility = 0;
   List<VMId> vm_ids = Nil;
   Int n = length(maps);
   Int i = 0;
   while (i < n){
      Pair<RequestId,Int> map = nth(maps, i);
      RequestId request_id = fst(map);
      Int vmi_index = snd(map);
      Maybe<Task> maybe_task = lookup(task_map, request_id);
      if (maybe_task != Nothing){
         Task task = fromJust(maybe_task);
         Request request = taskRequest(task);
         List<VMInfo> vmis = taskVMInfoList(task);
         if (vmi_index >= length(vmis)){
            rejected = Cons(request, rejected);
            Rat penalty_fr = priceValue(requestPenaltyFR(request));
```

```
assigned = put(assigned, vm_id, Pair(vm_info, requests));
vm_ids = Cons(vm_id, vm_ids);
```

```
            Pair<VMInfo, List<Request>> p = fromJust(maybe_assigned);
               requests = snd(p);
            }
            requests = Cons(request, requests);
            assigned = put(assigned, vm_id, Pair(vm_info, requests));
            vm_ids = Cons(vm_id, vm_ids);
         }
      }
      i = i + 1;
   }
   while (vm_ids != Nil){
```

Note: cofloco …
Exception: error(existence_error(procedure,unfolding:member/2),context(system: <meta-call> / 1,_G991))

```
            while (rejects != Nil){
               rejected = Cons(head(rejects), rejected);
               rejects = tail(rejects);
            }
         }
      }
      vm_ids = tail(vm_ids);
   }
   return Solution(problem, rejected, assigned, Price(utility), maps);
}
```

3: Resource Analysis

- **SRA** results

UPDATE: not relevant to case study

Selected: **`createNextGeneration(...)`**.

Types

Equations

```
eq(cnew(VM), 1, [], [VM < 3]).
eq(cnew(VM), 0, [], [VM = 3]).

eq(crel(VM), -1, [],[VM = 1]).
eq(crel(VM), 0, [], [VM > 1]).
```

UBs

```
Warning: the following predicates are never called:[crel/1]
Partitioned cost of cnew(VM):
0
 if [VM=3]
1
 if [2>=VM]
```

Identical results selecting **`mutate(...)`**, **`crossover(...)`**, **`bestSolution(...)`** ...

ETICS - Assurance

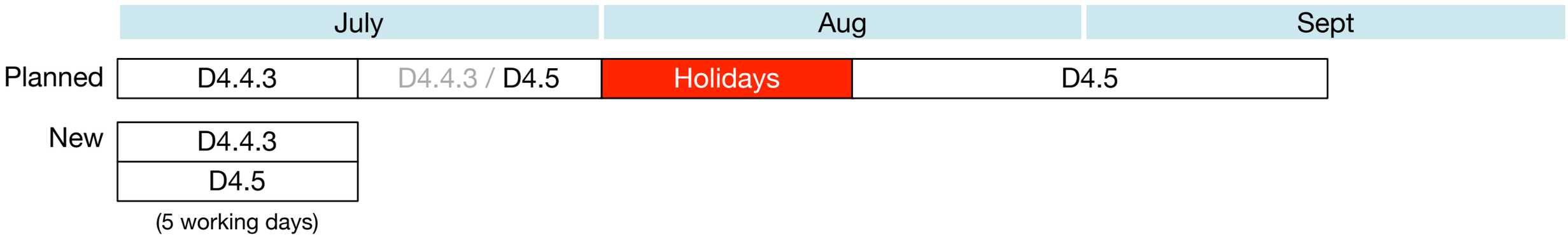
- Code Simulation (Erlang)
- Deadlock Analysis
- Resource Analysis
- **Java Code Generation**

4: Java Code Generation

- **Problem**
  - Automatically translate ABS model to Java

Tool not available

- Postpone the deliverable?



ETICS - Assurance

- **Summary**
  - Large scale tests with Erlang simulator**?**
  - Not all the tools worked / are available
    - Large + detailed code base**?**
  - Require “restricted” ABS:
    - Inheritance**?**
    - The way loops are expressed**?**

```
while (hasNext(set)){ … }
```

vs

```
while (i < n){ … }
```

    - Other “in principle” cases**?**
- How representative is the ABS Model of the original code**?**