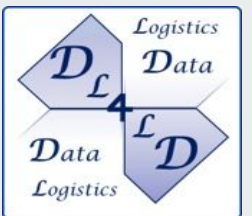




# Seamless Integration and Testing for MAS Engineering

Mostafa Mohajeri (m.mohajeriparizi@uva.nl),  
Giovanni Sileno, Tom van Engers  
*University of Amsterdam*

3-4 May, 2021, EMAS2021 @ AAMAS



# Introduction



- In this paper we elaborate on the development toolbox available for MAS engineering
  - It is not compatible with nor as mature as mainstream software systems
  - Particularly for DevOps operations such as tests and integration
  - Resulting in complicated and ineffective manual development cycles
  - Especially in situations where MAS is part of a bigger project
- Propose a practical approach for testing and integration in MAS
  - Enhance the AgentScript (ASC2) Agent-Oriented Programming framework
  - With an effective development cycle
    - Automated builds
    - Test frameworks and Code coverage tools
    - Continuous Integration (CI) services

# Introduction
















- Our main research: development of Data-Sharing Infrastructures, e.g, Data Marketplaces
- Typically include computational and institutional actors
  - Distributed over multi-domain networks
  - Across several jurisdiction with distinct norms
  - Also subjected to infrastructural, domain and ad-hoc policies
- For such systems, agent-based programming provides an intuitive way to model and program actors
  - Particularly the Belief-Desire-Intention (BDI) model
    - Defines actors as *intentional agents*
    - The mapping between institutional actors and intentional agents is well-studied in the MAS literature
- The problem: Our MAS development toolbox is not on par with our colleagues

**Background**

# Software Test and Integration

- Set of development tools we aim to bridge into MAS

Code Repositories	Build Tools	Test Frameworks	Continuous Integration	Source analysis
 	  	 	  	  

# Levels of Testing



## Unit

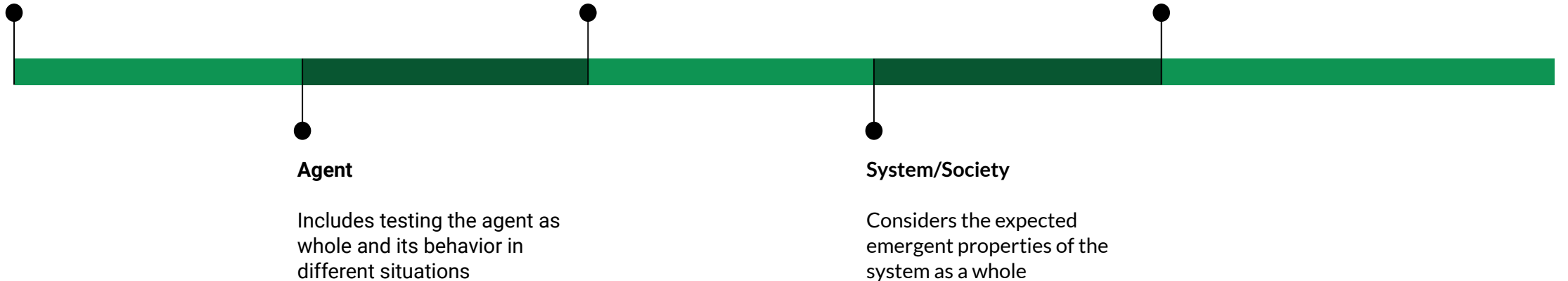
Testing internal components of an agent: Belief Base, Communication Layer, etc.

## Group/Integration

Aims at the interactions between agents in the system, e.g, communications protocols

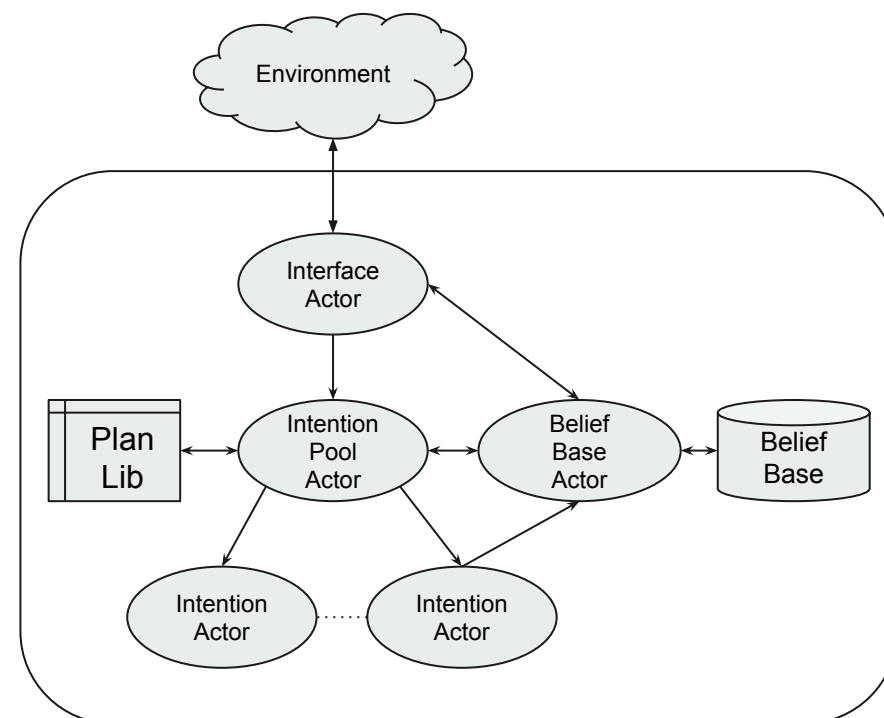
## Acceptance

Check the compliance of the software with given stakeholders' requirements.



# AgentScript Cross-Compiler

- Target MAS framework: *AgentScript Cross-Compiler (ASC2)*
  - A source-to-source compiler
  - Translates agent specifications defined in a high level language
    - Inspired by AgentSpeak(L)/Jason
  - To executable code written in a lower level language
    - The underlying executable language is *Scala*
    - ASC2 utilizes *Akka* actor framework for concurrency
    - Each agent is translated to an actor-oriented micro-system

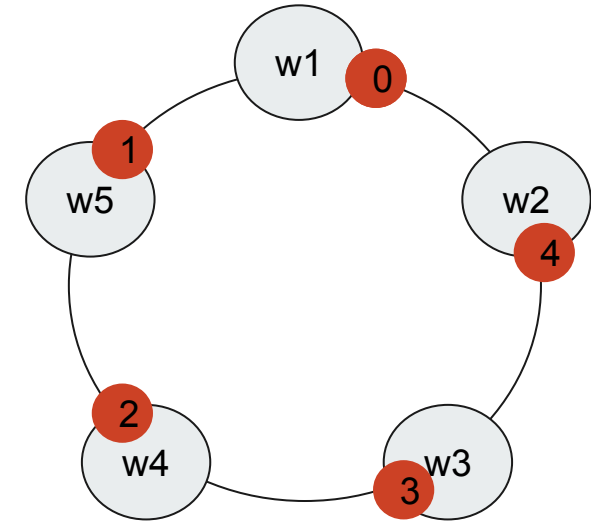


# Method



# Example Project: The Token Ring

- $w$  worker agents and  $1$  master agent
  - at the start the master sends the number of workers to all
  - each worker finds its neighbor to form a ring
- $T$  tokens are distributed evenly between agents
  - by the **master** agent
- When an agent receives a token it passes it on to its neighbor
- Program finishes when all tokens hop  $N$  times
  - and this is reported to the master
- Example:
  - $w = 5, T = 1, N = 5$



# Worker Test Suite

---

Entity under test: A worker agent named **worker1**

Mocked entities:

1. A master agent named **master**
2. A worker agent named **worker2**

Injected entities

1. Belief Base of the **worker1**

Invocation conditions	Expected state/behavior
<b>worker1</b> receives an <b>init(N)</b> message with $N > 2$	<b>worker1</b> has <b>neighbor(worker2)</b> in its belief base
<b>worker1</b> has <b>neighbor(worker2)</b> in its belief base <b>worker1</b> receives a <b>token(T)</b> message with $T > 0$	<b>worker1</b> sends <b>token(T-1)</b> message to <b>worker2</b>
<b>worker1</b> receives a <b>token(0)</b> message	<b>worker1</b> sends <b>done</b> message to <b>master</b>

```

+!init(W) : W > 1 =>
  Nbr = "worker" +
    ((#name.replaceAll("worker","").toInt % W) + 1);
+neighbor(Nbr).

+!token(0) =>
  #coms.achieve("master", done).

+!token(N) : neighbor(Nbr) =>
  #coms.achieve(Nbr, token(N - 1)).

```

## Worker Agent Script

```

1 class TokenRingWorkerSpec extends ... {
2
3   val mas = new MAS()
4   val master = testKit.createTestProbe[IMessage]()
5   val neighbor = testKit.createTestProbe[IMessage]()
6   val worker1 = new worker()
7
8   override def beforeAll(): Unit = {
9     mas.registerAgent(worker1, name = "worker1")
10    mas.registerAgent(master, name = "master")
11    mas.registerAgent(neighbor, name = "worker2")
12  }
13
14  "A worker agent" should {
15    "have its neighbor in its belief base after `!init(N)`" in {...}
16    "send a `!done` to master on `!token(0)`" in {...}
17
18    "send a `!token(N-1)` to its neighbor on `!token(N)`" in {
19      worker1.event(achieve,"token(10)").send()
20      neighbor.expect(event(achieve,"token(9)").source(worker1))
21    }
22  }
23 }

```

## Worker Agent Test Suite

```

+!init(W) : W > 1 =>
  Nbr = "worker" +
    ((#name.replaceAll("w", "1")) + 1)
  +neighbor(Nbr).

+!token(0) =>
  #coms.achieve("master", done).

+!token(N) : neighbor(Nbr) =>
  #coms.achieve(Nbr, token(N)).

```

## Worker Agent Script

```

$ sbt compile test coverageReport
...
[info] A worker agent should
[info] - have its neighbor in its belief base after `!init(N)`
[info] - send a `!done` to master on `!token(0)`
[info] - send a `!token(N-1)` to its neighbor on `!token(N)`
...
[info] All tests passed.
...
[info] Coverage report
[info] - statement coverage: 90%
[info] - branch coverage: 10/11

```

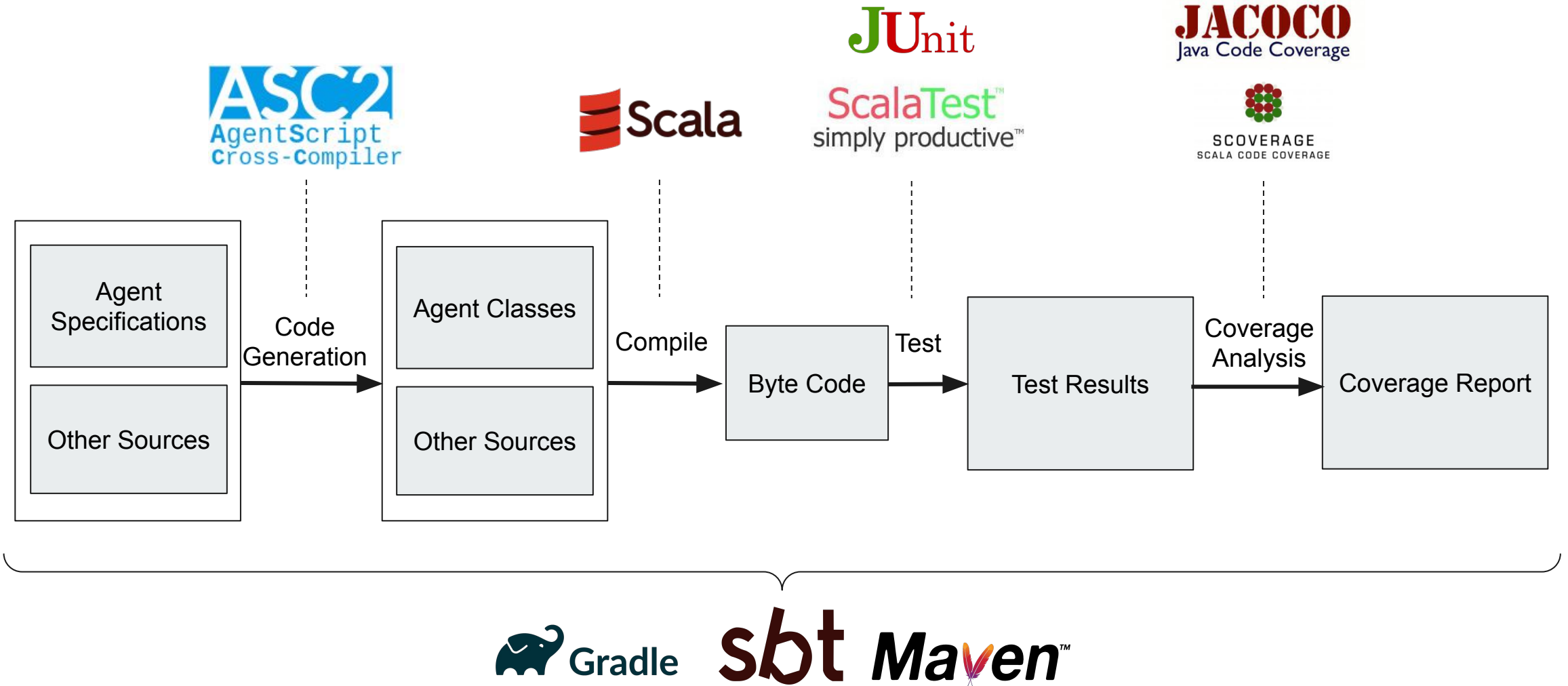
```

1 class TokenRingWorkerSpec extends ... {
...
  testProbe[IMessage]()
  testProbe[IMessage]()
...
  val c = {
    name = "worker1"
    name = "master"
    name = "worker2"
  }
...
  "send a `!token(N-1)` to its neighbor on `!token(N)`" in {
18     worker1.event(achieve, "token(10)").send()
19     neighbor.expect(event(achieve, "token(9)").source(worker1))
20   }
21 }
22 }
23 }

```

## Worker Agent Test Suite

# Automated Build/Test Process



# Group/Society Test Suite

---

Entity under test: A token ring system with  $W = 100$ ,  $T = 50$  and  $N = 4$

Injected entities:

1. The communication layer of agents that stores all messages

Invocation conditions	Expected state/behavior
<b>1 master</b> agent and <b>100 workers</b> are in the system	The System should stop in under 10 seconds
<b>master</b> receives a <b>start(50,4)</b> message	There should be 250 <b>token(X)</b> and 50 <b>done</b> messages sent in the system

```

1 class TokenRingIntegrationSpec extends ... {
2
3   //a communication layer that records a trace of the interactions
4   object recordedComs extends AgentCommunicationsLayer { ... }
5
6   val token_pattern = "token\\([0-9]+\\)".r
7   val done_pattern = "done".r
8
9   "A token ring MAS with W = 100, T = 50 and N = 4" should {
10    "have 250 `token(X)` and 50 `done` message" in {
11      // create the agents
12      mas.registerAgent(new worker(coms = recordedComs), num = 100)
13      mas.registerAgent(new master(coms = recordedComs), name = "master")
14      // invoke the system
15      mas.getAgent("master").event(achieve, "start(50,4)").send()
16      // verify the interactions
17      watchdog.expectTerminated( mas, 10.seconds )
18      assert(recordedComs.trace.count(token_pattern.matches) == 250)
19      assert(recordedComs.trace.count(done_pattern.matches) == 50)
20    }
21  }
22 }

```

## Token Ring Test Suite

```

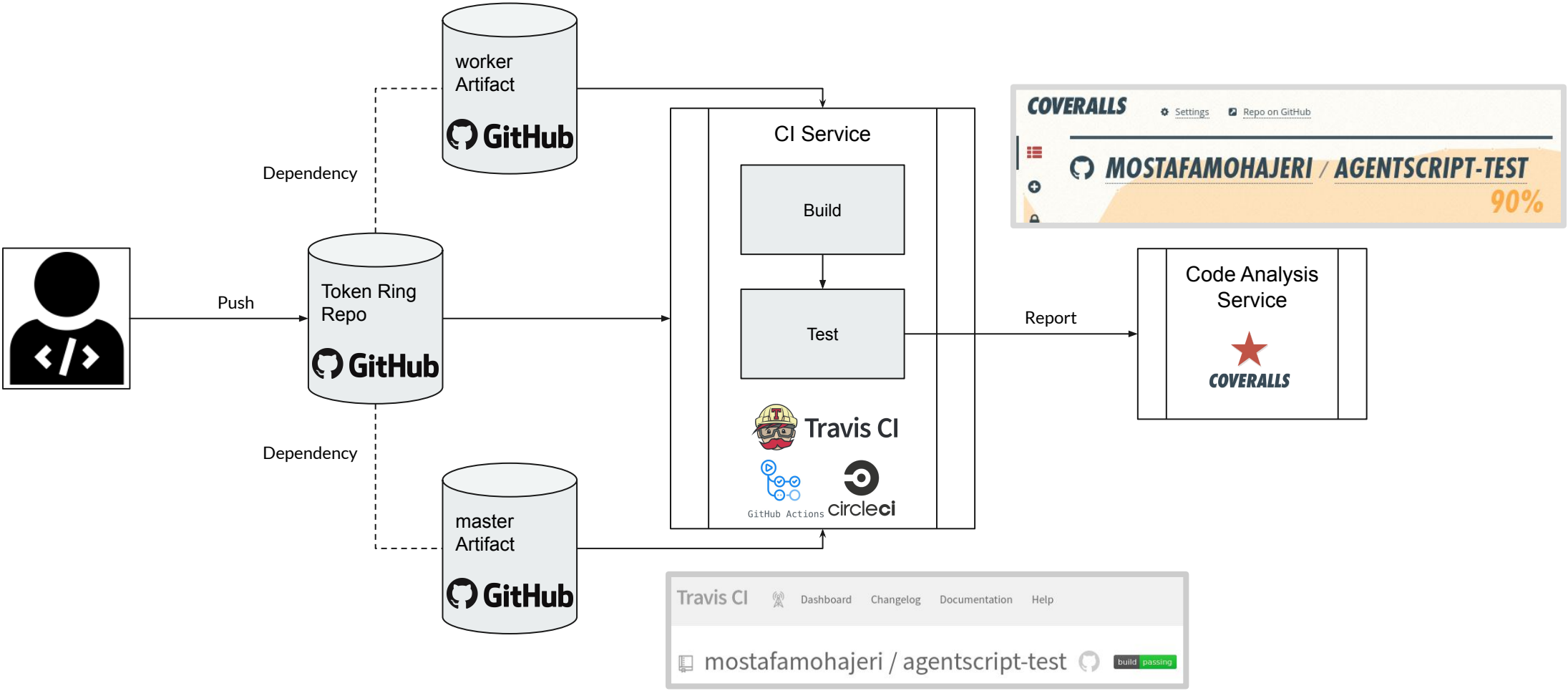
1 class TokenRingIntegrationSpec extends ... {
2
3   //a communication layer that records a trace of the interactions
4   object recordedComs extends AgentCommunicationsLayer { ... }
5
6   val token_pattern = "token\\\[0-9]+\\".r
7   val done_pattern = "done".r
8
9
10  [info] A token ring MAS with W = 100, T = 50 and N = 4 should
11  [info] - have 250 `token(X)` and 50 `done` message
12  ...
13  [info] All tests passed.
14
15
16  // verify the interactions
17  watchdog.expectTerminated( mas, 10.seconds )
18  assert(recordedComs.trace.count(token_pattern.matches) == 250)
19  assert(recordedComs.trace.count(done_pattern.matches) == 50)
20  }
21 }
22 }

```

## Token Ring Test Suite



# Continuous Integration



✓ main fix #16.1 passed Restart job

Commit 767f8c4 Ran for 2 min 15 sec

Compare 57ea90b...767f8c4 2 months ago

Branch main

Mostafa Mohajeriparizi

</> JDK: oraclejdk11 Scala: 2.13.3

AMD64

Job log View config

```

1 Worker information
6
7 Build system information
161
162
163 Installing oraclejdk11
179 Updating sbt
180
181 $ git clone --depth=50 --branch=main
191
192 $ export JVM_OPTS=@/etc/sbt/jvmopts
193 $ export SBT_OPTS=@/etc/sbt/sbtopts
  
```

DEFAULT BRANCH: MAIN

REPO ADDED 11 FEB 2021 08:55AM UTC

TOTAL FILES 2

# BUILDS 13

BADGE coverage 90%

TOKEN gL2S6nTwq7uzFnqsU96

LAST BUILD ON BRANCH MAIN

COMMITTED 1 MAR 2021 - 21:32 COVERAGE DECREASED (-6.7%) TO 90.476%

BUILD #	BUILD TYPE	COMMITTED BY	COMMIT MESSAGE	RUN DETAILS
16	push travis-ci-com	mostafamohajeri	fix	76 of 84 relevant lines covered (90.48%) 0.9 hits per line



SOURCE FILES ON MAIN

TREE LIST 2 CHANGED 1 SOURCE CHANGED 1 COVERAGE CHANGED 1

- 100.0 src/
- 90.12 target/

# **Conclusion and Future Works**

# Conclusion

---

- We showed how a MAS engineer that uses a logic-based Agent-Oriented Programming framework can utilize and benefit from modern mainstream DevOps tools
- Advantages:
  - Access to vast number of tools and services developed by the SE community
  - More effective and scalable development cycle for MAS
  - Compatible test and CI process with the non-MAS parts of the project
  - The approach does not replace, but complements other verification/test/debug approaches for MAS
    - Formal methods
    - Mind inspectors
    - MAS Debuggers

# Future Works



- Defining test suite in the same language as the agent script
  - Disparity between agent scripts and tests can be counter-productive
- Applying the same pattern to (JVM) debuggers
  - Now the debuggers only identify the generated code and is not linked to the original script
- Applying the same pattern to Deployment and Continuous Delivery
  - Deploying the Agent Systems seamlessly and automatically
- Studying the feasibility of the same approach for other BDI/MAS frameworks

**Thank you :-)**





# Seamless Integration and Testing for MAS Engineering

Mostafa Mohajeri (m.mohajeriparizi@uva.nl),  
Giovanni Sileno, Tom van Engers  
*University of Amsterdam*

May 4th, 2021, EMAS2021 @ AAMAS

