CASTOR

Software Research Centre

CASTOR

- Academia-industry joint research center
- Software technology
- Launched in May 2018

CASTOR objectives

- Relevant software technology research
- Graduates in software technology
- Open science and innovation
- Structure the Software Area at KTH

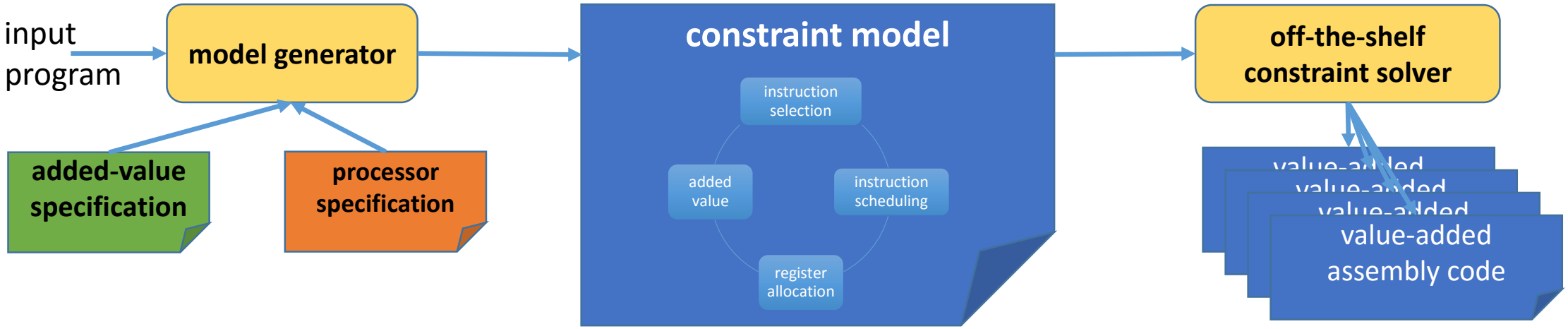
CASTOR – software industry

- Telecom infrastructure
  - Massive scale distributed systems
  - SW technology: languages, optimized compilers, machine learning, auto-tuning, cloud/edge computing, DevOps, security
- Defense systems
  - Multiple areas ranging from mechanics to software
  - SW technology: embedded systems, architecture, languages, cloud, security, DevOps

CASTOR research areas

- 20 KTH faculty
- Reliability
  - Repair, fault injection, continuous integration, testing
- Security
  - Formal verification, SW/HW, diversity, information flow
- Data-driven systems
  - Distributed systems, analytics, machine learning
- Embedded systems
  - Languages, verification, compilation

Unison: Value-added compiler



- Express **how** programs should compute, not only what!
  - pry open the compiler: thermal budget, power, timing, user annotations, …
  - generate multiple, different assembly codes for software diversification
  - …
- Using combinatorial optimization (constraint programming)
  - based on Unison (code generator developed in joint project KTH, RISE SICS, Ericsson)
  - generate **model for generating code** rather than code (simple, flexible, optimal)

Unison: Value-added compiler

ERICSSON

Networks Digital Services Managed Services Portfolio Tech Innovation Trends & Insights

Ericsson Research Blog

Research, insights and technology reflections

Filter by topic Filter by author

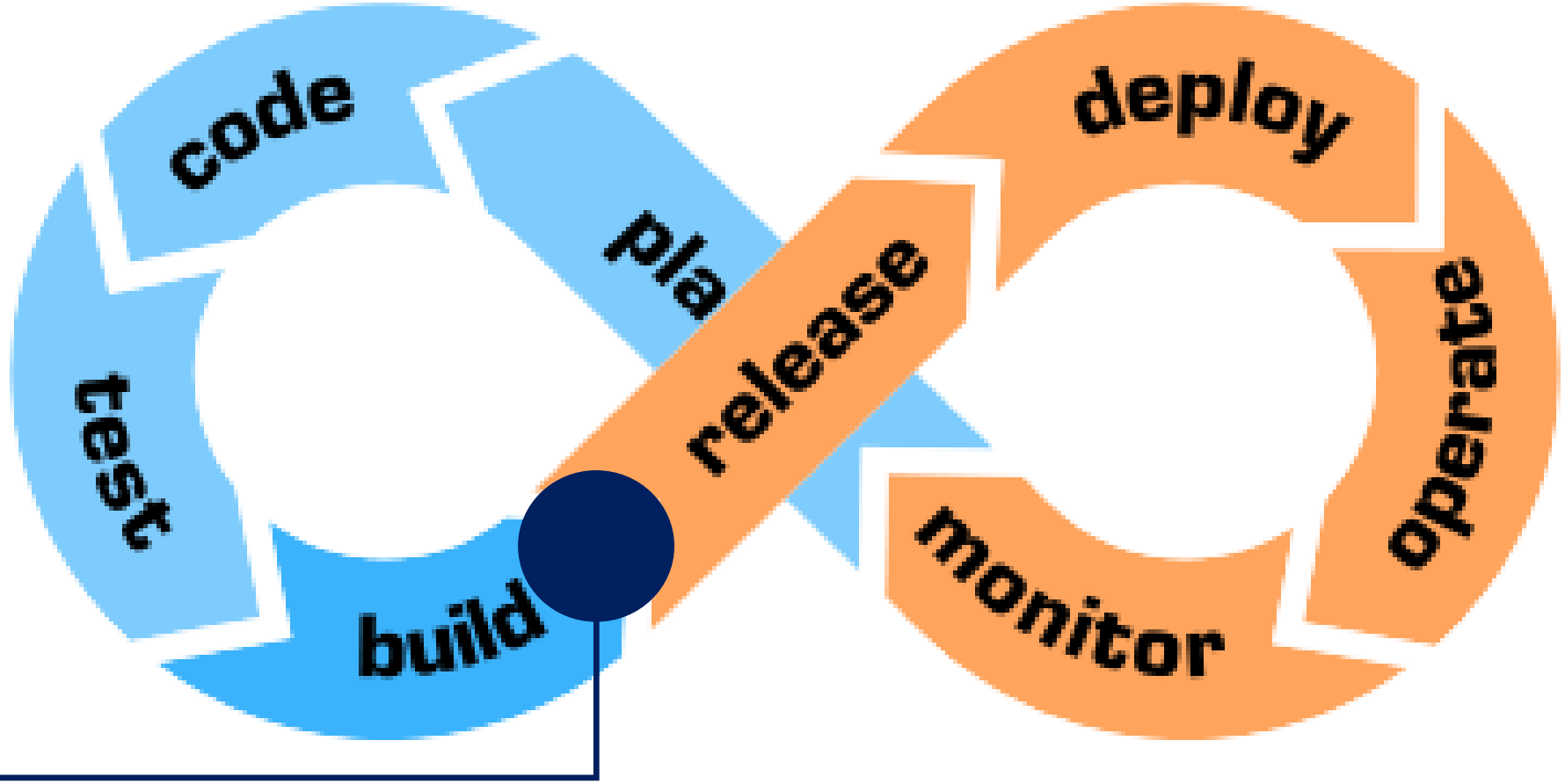
Outperforming state-of-the-art compilers in Unison

Unison's source code http://unison-code.github.io/

DevOps for software research

B. Baudry, M. Monperrus

DevOps



Continuous integration

The pull request loop

pull req.

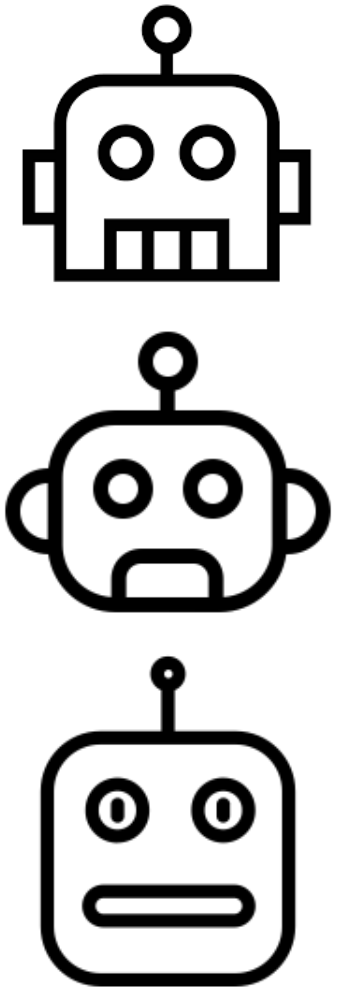
Collaboration platform

The pull request loop

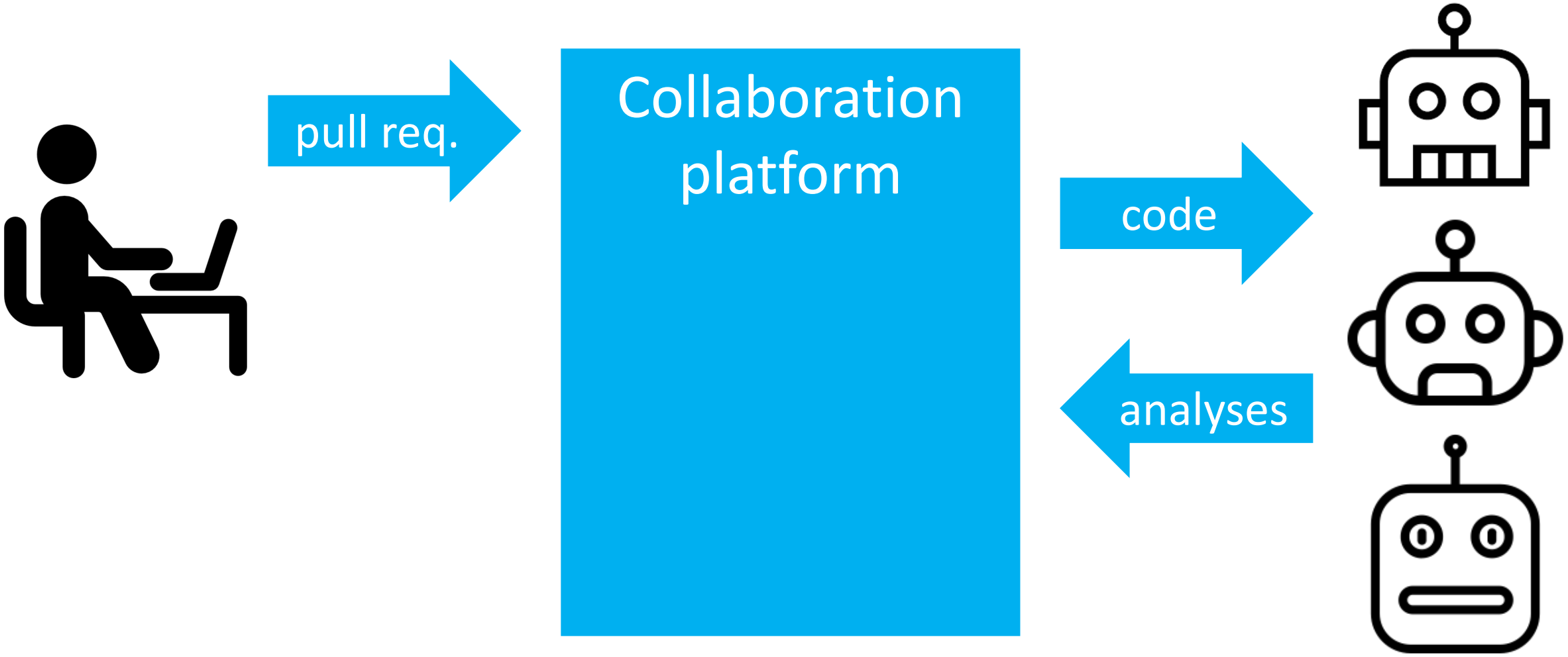
pull req.

Collaboration platform

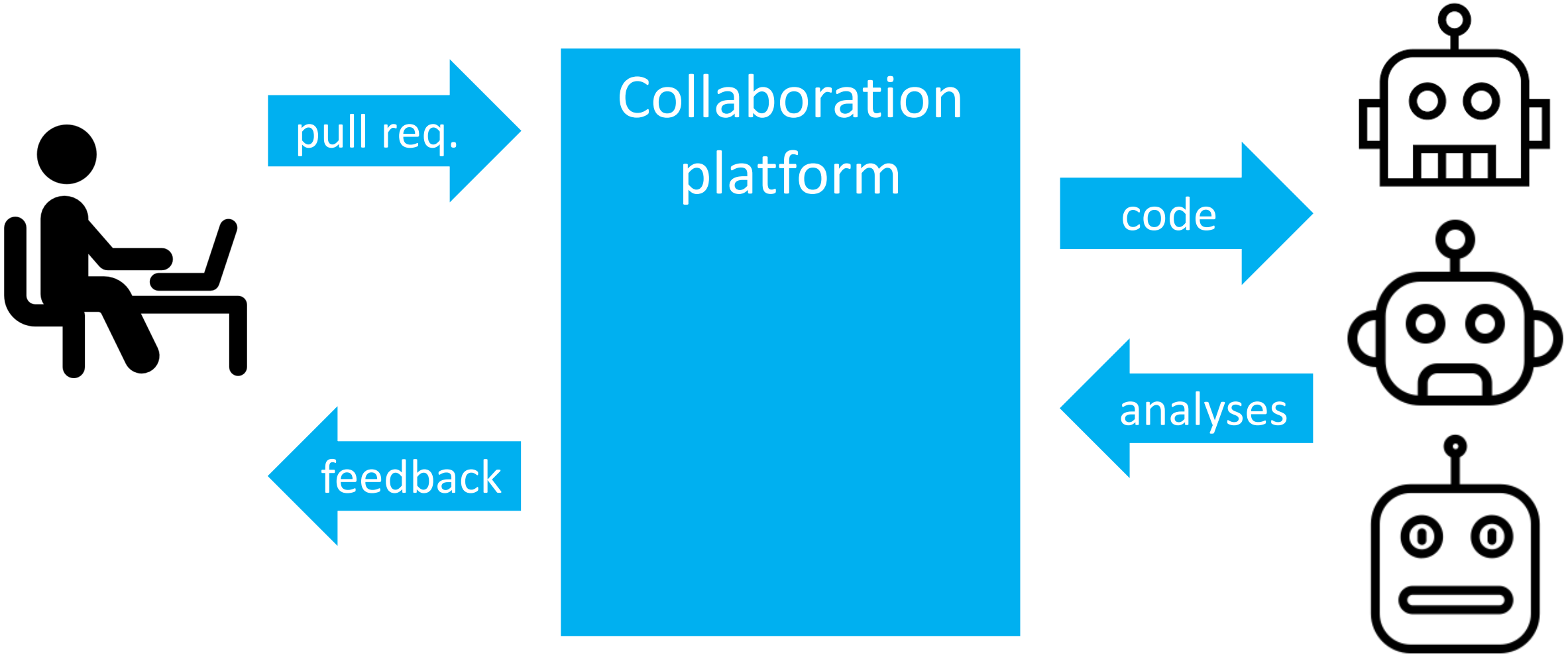
code



The pull request loop

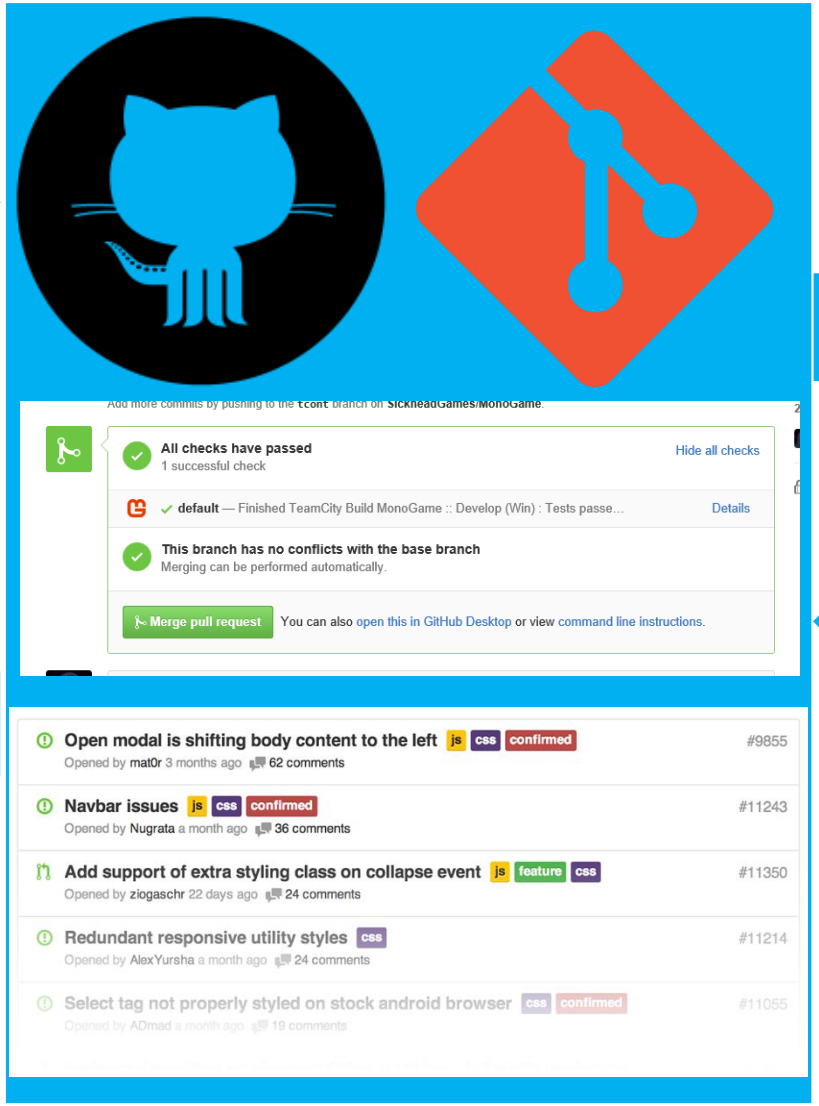


The pull request loop



The pull request loop

pull req.



code

analyses

feedback

maven

The pull request loop for research

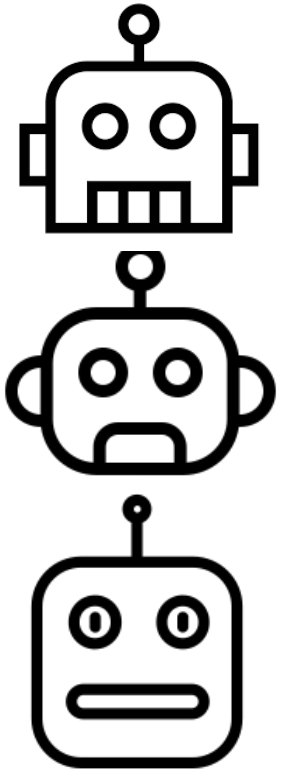
pull req.

feedback

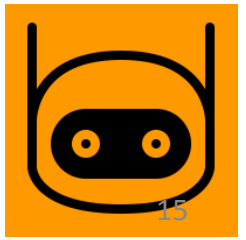
Collaboration platform

code

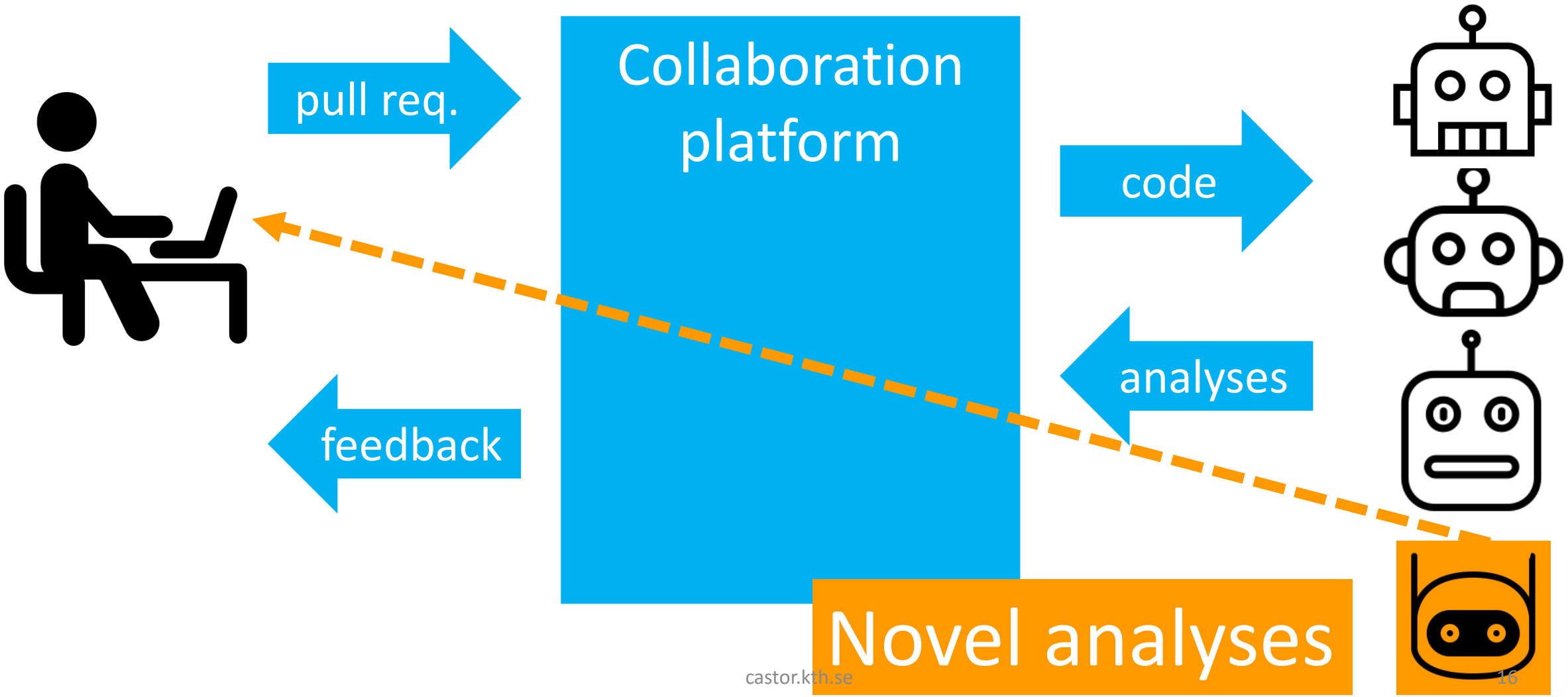
analyses



Novel analyses



The pull request loop for research



DSpot

```
@Test
public void html() {
 Attribute attr = new Attribute("key", "value &");
 assertEquals("key=\"value &\"", attr.html());
 assertEquals(attr.html(), attr.toString());}
```


Automatic Test Improvement with DSpot: a Study with Ten Mature Open-Source Projects. B. Danglot, O. Luis Vera-Pérez, B. Baudry, M. Monperrus. Submitted to EMSE.

DSpot

```
@Test
public void html() {
 Attribute attr = new Attribute("key", "value &");
 assertEquals("key=\"value &\"", attr.html());
 assertEquals(attr.html(), attr.toString());}

@Test
public void html_add33() throws Exception {
 Attribute attr = new Attribute("key", "value &");
 Assert.assertEquals("key=\"value &\"", attr.html());
 Assert.assertEquals("key=\"value &\"", attr.toString());
 Assert.assertEquals("key", attr.getKey());
 Assert.assertEquals("value &", attr.getValue()); }
```


Automatic Test Improvement with DSpot: a Study with Ten Mature Open-Source Projects. B. Danglot, O. Luis Vera-Pérez, B. Baudry, M. Monperrus. Submitted to EMSE.

DSpot

```
146         ByteArrayOutputStream out = new ByteArrayOutputStream();
    -       writeListTo(out, foos, SerializableObjects.foo.cachedSchema());
147 +       final int bytesWritten = writeListTo(out, foos, SerializableObjects.foo.cachedSchema());
148 +       assertEquals(0, bytesWritten);
149         byte[] data = out.toByteArray();
```


Automatic Test Improvement with DSpot: a Study with Ten Mature Open-Source Projects. B. Danglot, O. Luis Vera-Pérez, B. Baudry, M. Monperrus. Submitted to EMSE.

DSpot

project	# opened	# merged	# closed	# under discussion
javapoet	4	4	0	0
mybatis-3	2	2	0	0
traccar	2	1	0	1
stream-lib	1	1	0	0
mustache	2	2	0	0
twilio	2	1	0	1
jsoup	2	0	1	1
prostostuff	2	2	0	0
logback	2	0	0	2
retrofit	0	0	0	0
total	19	13	1	5

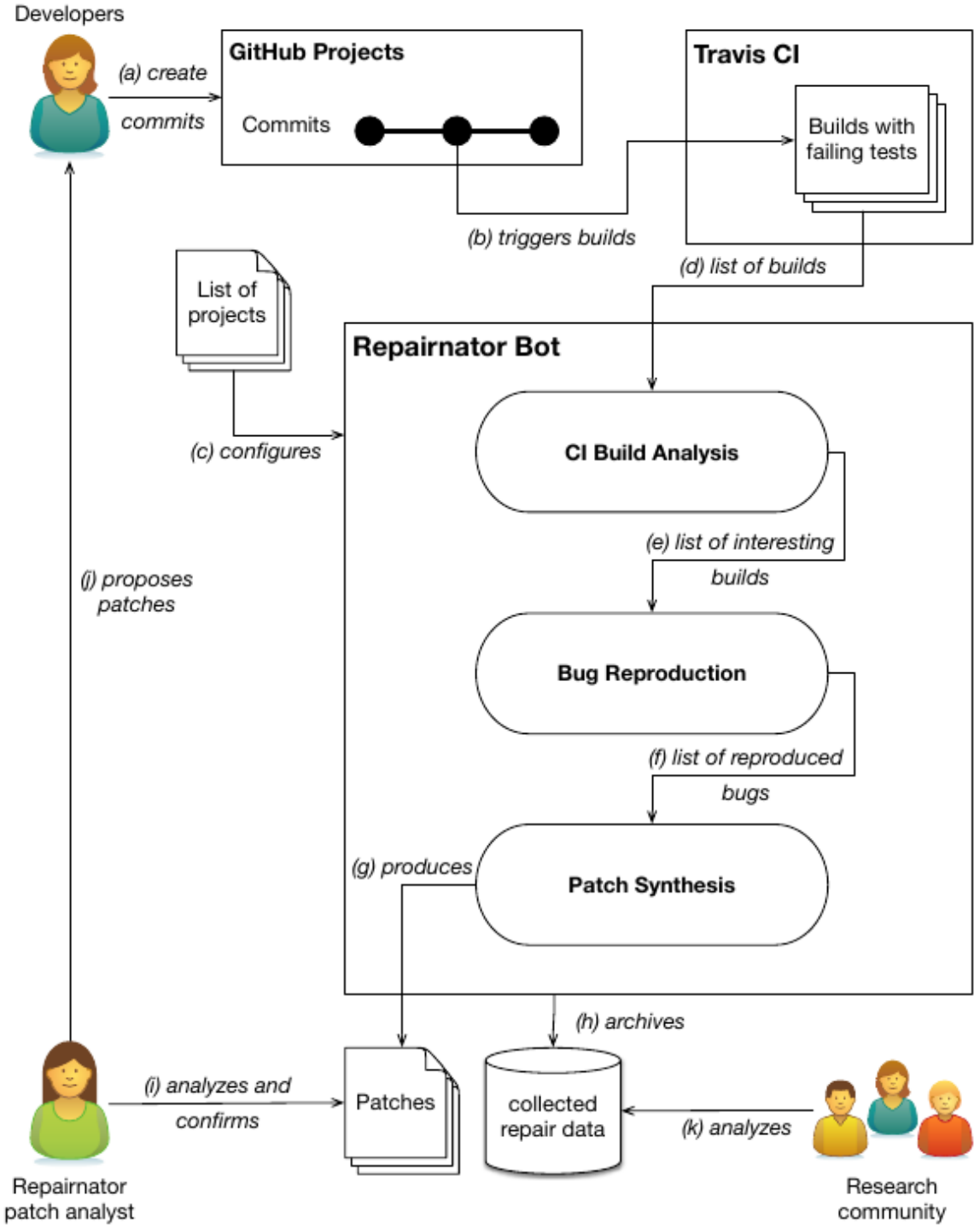
Automatic Test Improvement with DSpot: a Study with Ten Mature Open-Source Projects. B. Danglot, O. Luis Vera-Pérez, B. Baudry, M. Monperrus. Submitted to EMSE.

Repairnator

- Automatic repair bot to target build failures
- Runs since February 2017
- ICSE SEIP paper reports on
  - 11 523 test failures over 1 609 open-source software projects hosted on GitHub
  - generated patches for 15 different bugs

How to Design a Program Repair Bot? Insights from the Repairnator Project. S. Urli, Z. Yu, L. Seinturier, M. Monperrus. ICSE SEIP 2018.

Repairnator



How to Design a Program Repair Bot? Insights from the Repairnator Project. S. Urli, Z. Yu, L. Seinturier, M. Monperrus. ICSE SEIP 2018.

Repairnator

Project	Builds w/ patches	Nopol patches	NPEFix patches
jamesagnew/hapi-fhir	1	35	0
spotify/cassandra-reaper	1	1	0
xmlunit/xmlunit	1	145	0
apache/pdfbox	1	120	0
LiveRamp/hank	1	4	0
spring-cloud/spring-cloud-dataflow	1	0	1
IQSS/dataverse	2	0	16
bonigarcia/webdrivermar	3	30	0
GeoWebCache/geowebca	1	0	2
timmolter/XChange	1	0	4
phax/jcodemodel	1	624	0
phoenixnap/springmvc-raml-plugin	1	348	0
Total	15	1 307	23

How to Design a Program Repair Bot? Insights from the Repairnator Project. S. Urli, Z. Yu, L. Seinturier, M. Monperrus. ICSE SEIP 2018.

Conclusion

CASTOR

- Research center
- Industry – academia joint software research
- Focus on software technology
- castor.kth.se