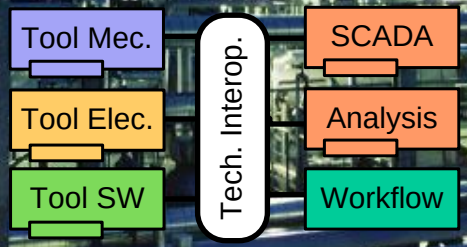
Flexible Engineering Process Automation Process: Continuous Integration & Test

Alexander Schatten Andreas Pieber Michael Handler Stefan Biffl
Christian Doppler Laboratory SE-Flex-AS
Institute of Software Technology and Interactive Systems (ISIS)
Vienna University of Technology
http://cdl.ifs.tuwien.ac.at



Motivation
Process „Continuous Integration & Test“

- **Engineering Service Bus** concept evaluation with real-world use cases
  - General-purpose use case for flexible engineering: CI&T
  - SE best-practice: continuous integration (CI) servers Continuum/Hudson
  - Limitation: CI servers are monolithic and hard to extend or integrate into a more complex tool landscape

- **„Continuous Integration and Test“** (CI&T)
  - Key part in an iterative systems development process
    - if part of a system or engineering model gets changed,
      the system has to be re-built and re-tested
      to identify defects early and
      to provide fast feedback on implementation progress
      to the project manager and the owners of the changed system parts.

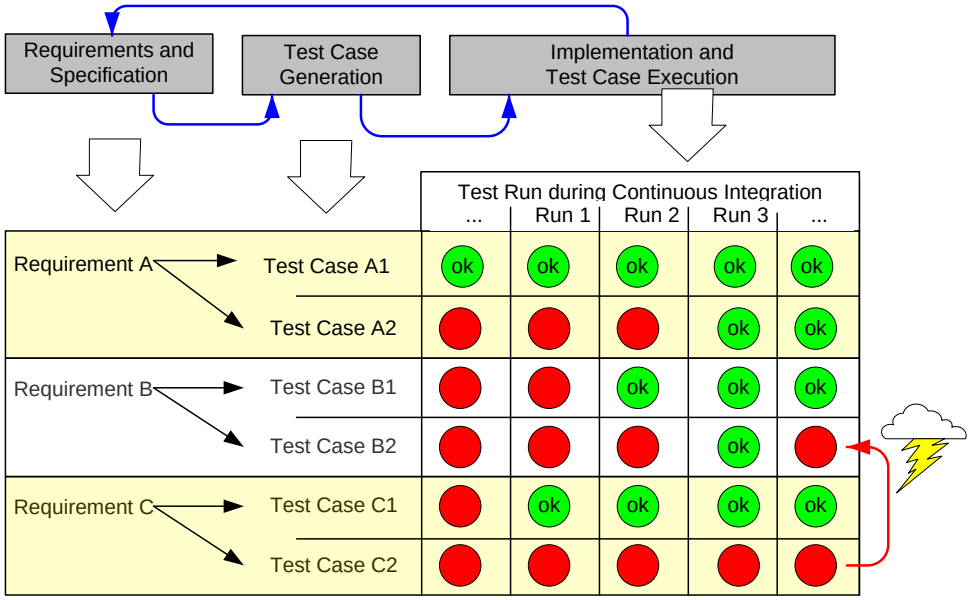
- **-> Feasibility study** with initial Engineering Service Bus prototype.
  - Sub process “Change, Test & Result Notification”
  - Technical integration of systems from several platforms.

Continuous Integration & Test Process for Iterative Quality Assurance

Continuous Integration & Test

- Frequent test runs
- Immediate feedback on test results (e.g., daily builds)
- Efficient regression testing.

- Needs **process automation** and tool support
  - **Build** system under test
  - **Test** automation
  - **Analysis** of test results
  - **Notification** on results

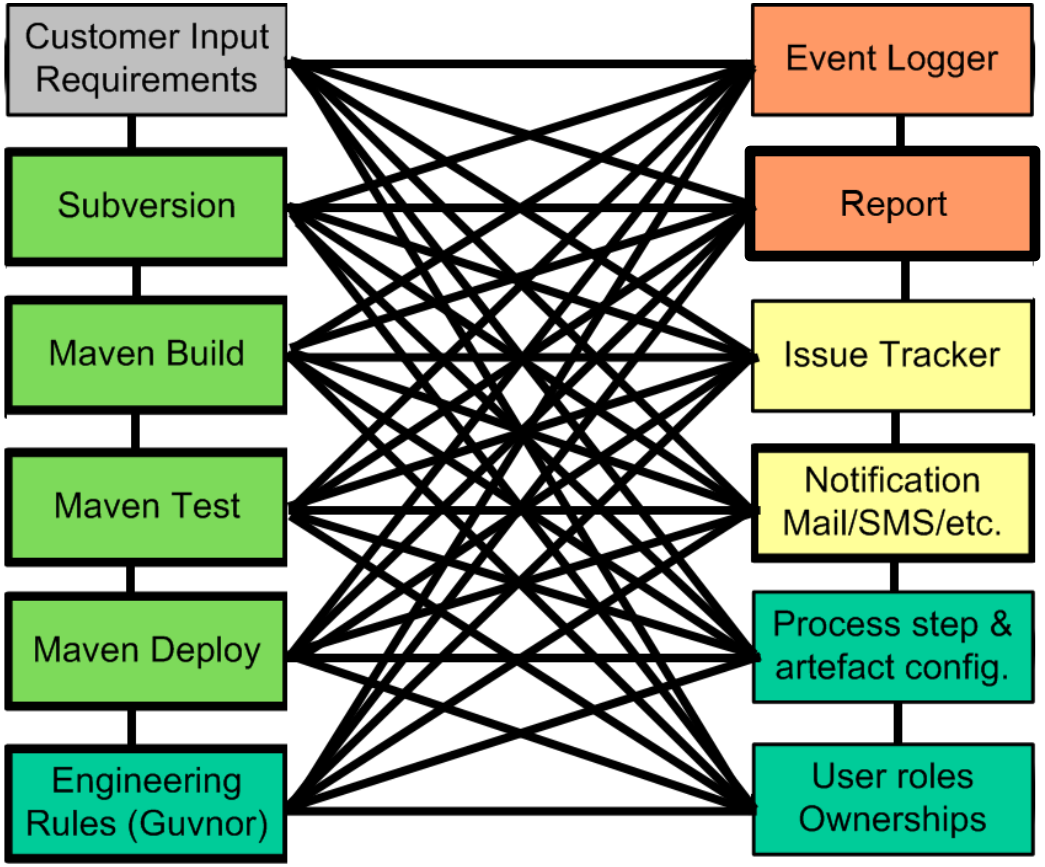


Current state of the practice

- Hudson
  - Open source software
  - Fixed process
  - Plugin structure for extensions
  - But takes considerable effort to gain the necessary knowledge
- Continuum
  - Open source software
  - Fixed process
  - Also considerable effort to adapt to specific requirements

Alternatives

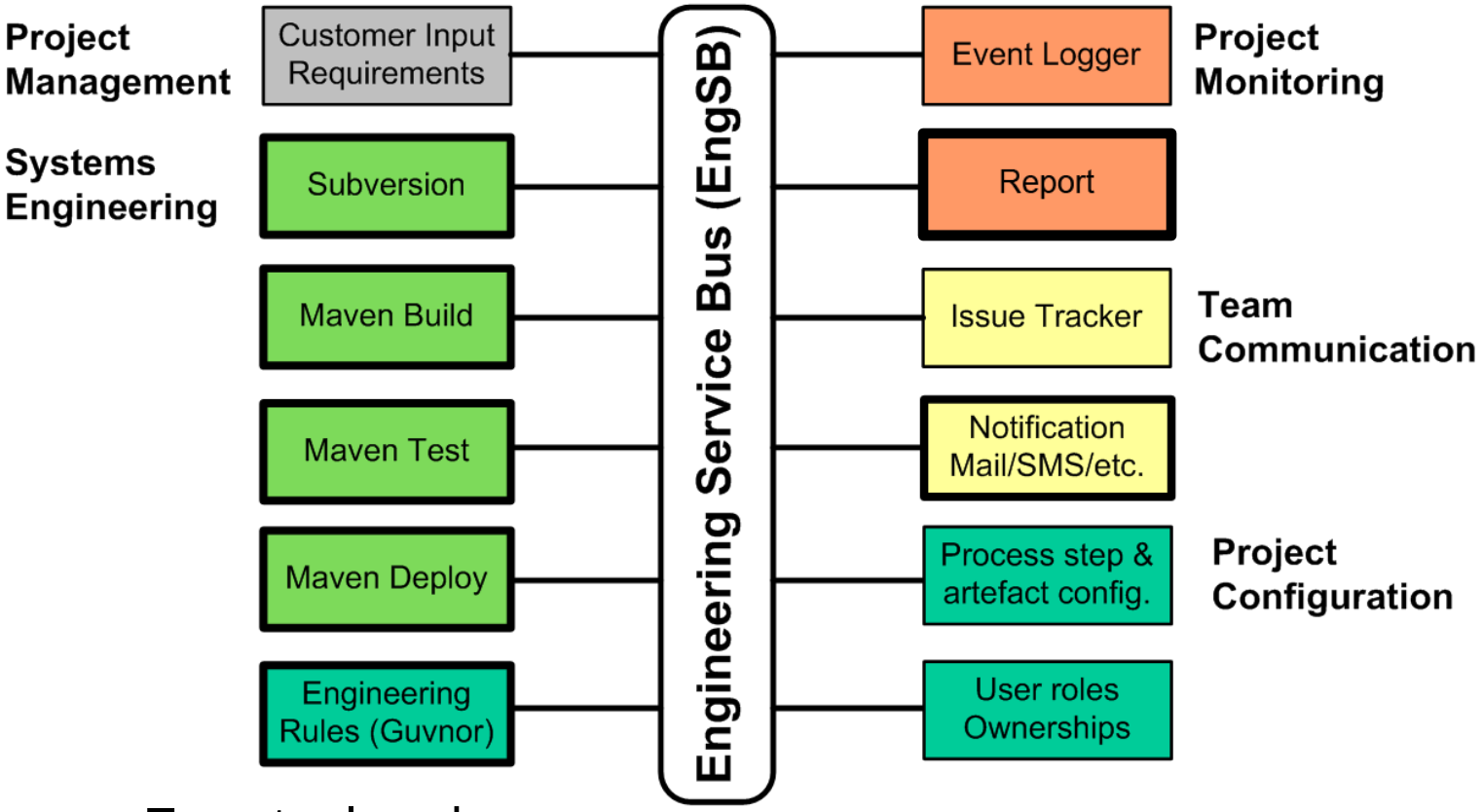
- Complex point to point integration



  - Very high maintenance effort needed
  - Hard to exchange a tool

Alternatives contd.

- Use OpenEngSB for CI & T Use Case



  - Easy tool exchange
  - Process is easy to adapt and extend

Tool Domains provide service interfaces

- Subversion for example is connected via the SCM domain
- Maven is connected via multiple domains (build, test, deploy)
- In Drools rules and workflows the process engineer uses the domains and their interfaces to connect to tools
- Domains can be configured to forward messages to a default tool
- It is possible to state explicitly to which tool a message shall be sent in a rule or workflow

Continuous Integration Process in Business Process Modeling Notation

Goal: **Flexible CI&T server** functionality.

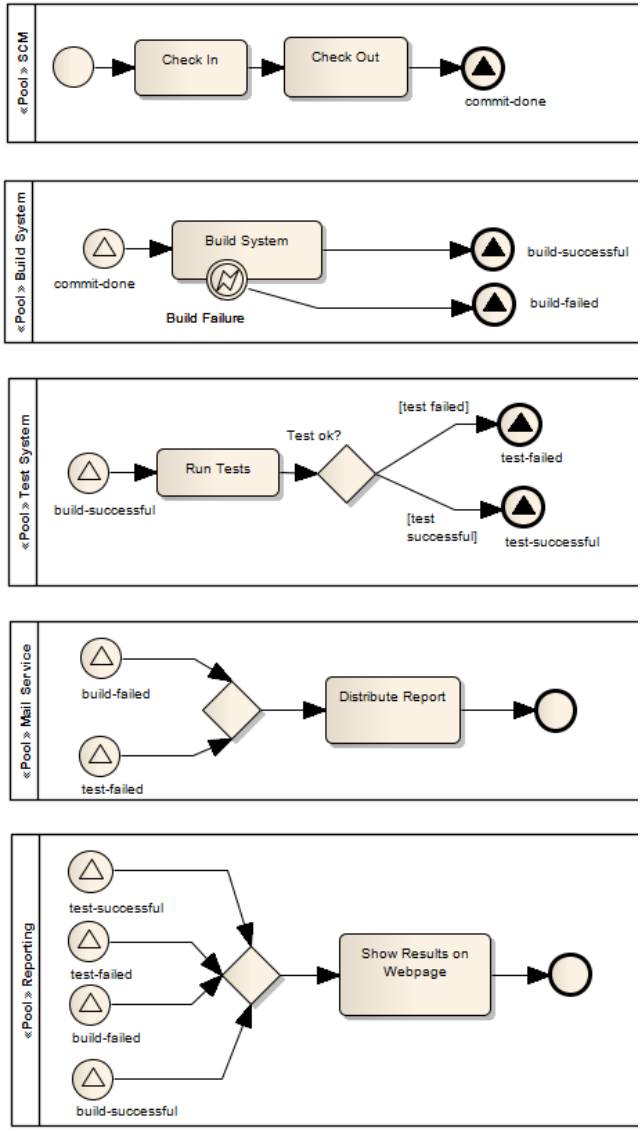
Continuous Integration (CI) Process

- 1. Build the source code,
- 2. Test the built source code,
- 3. Deploy the compiled source code
- 4. Send notification about result to a configured list of recipients.

Event-driven process definition (BPMN)

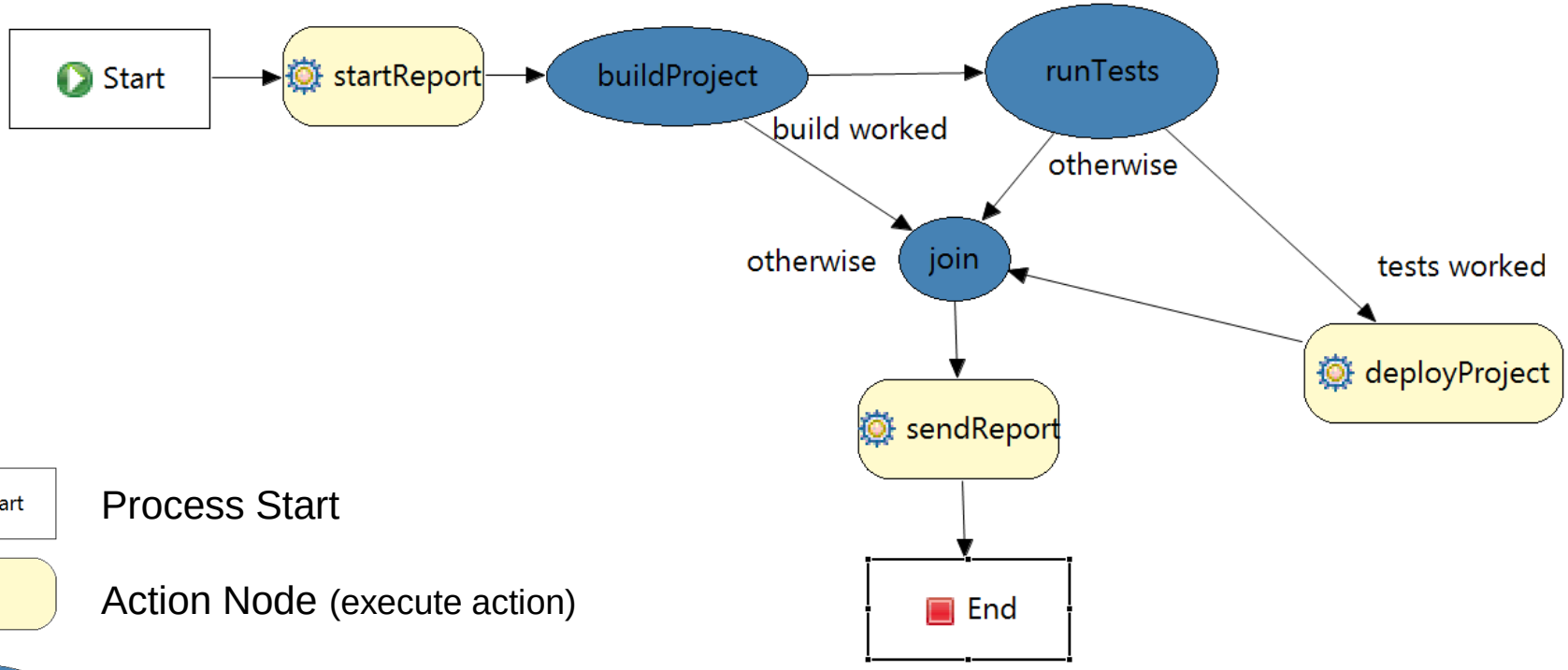
- Events
- Process steps
- Decisions
- Outgoing events

-> Decoupling of communication and tools.



The CI&T Process Model

- The CI&T process is defined with Drools Flow

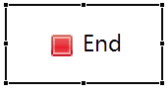


Process Start

Action Node (execute action)

Switch Node (execute action and decide further route through the process based on the result)

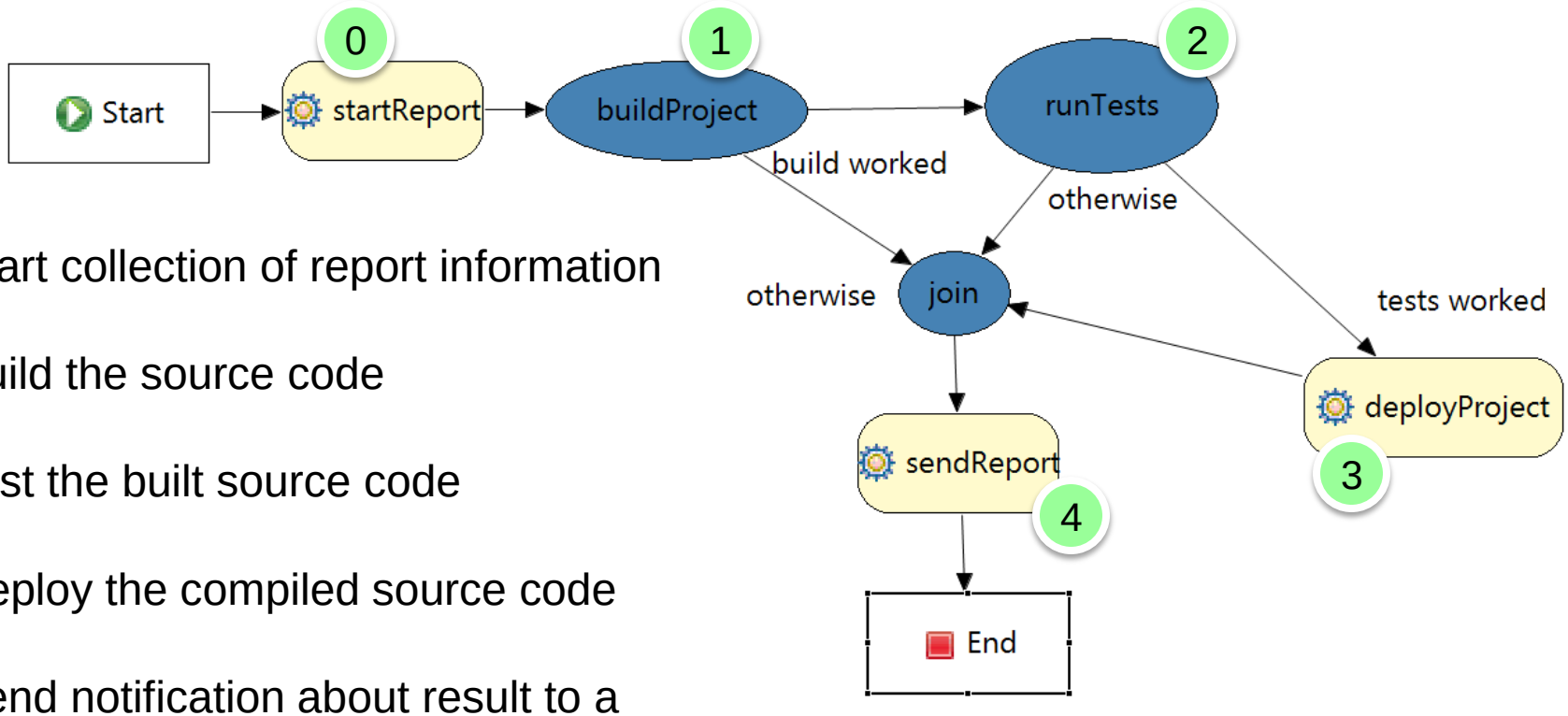
Join Node



Process End

The CI&T Process Model

- The CI&T process is defined with Drools Flow



0. Start collection of report information

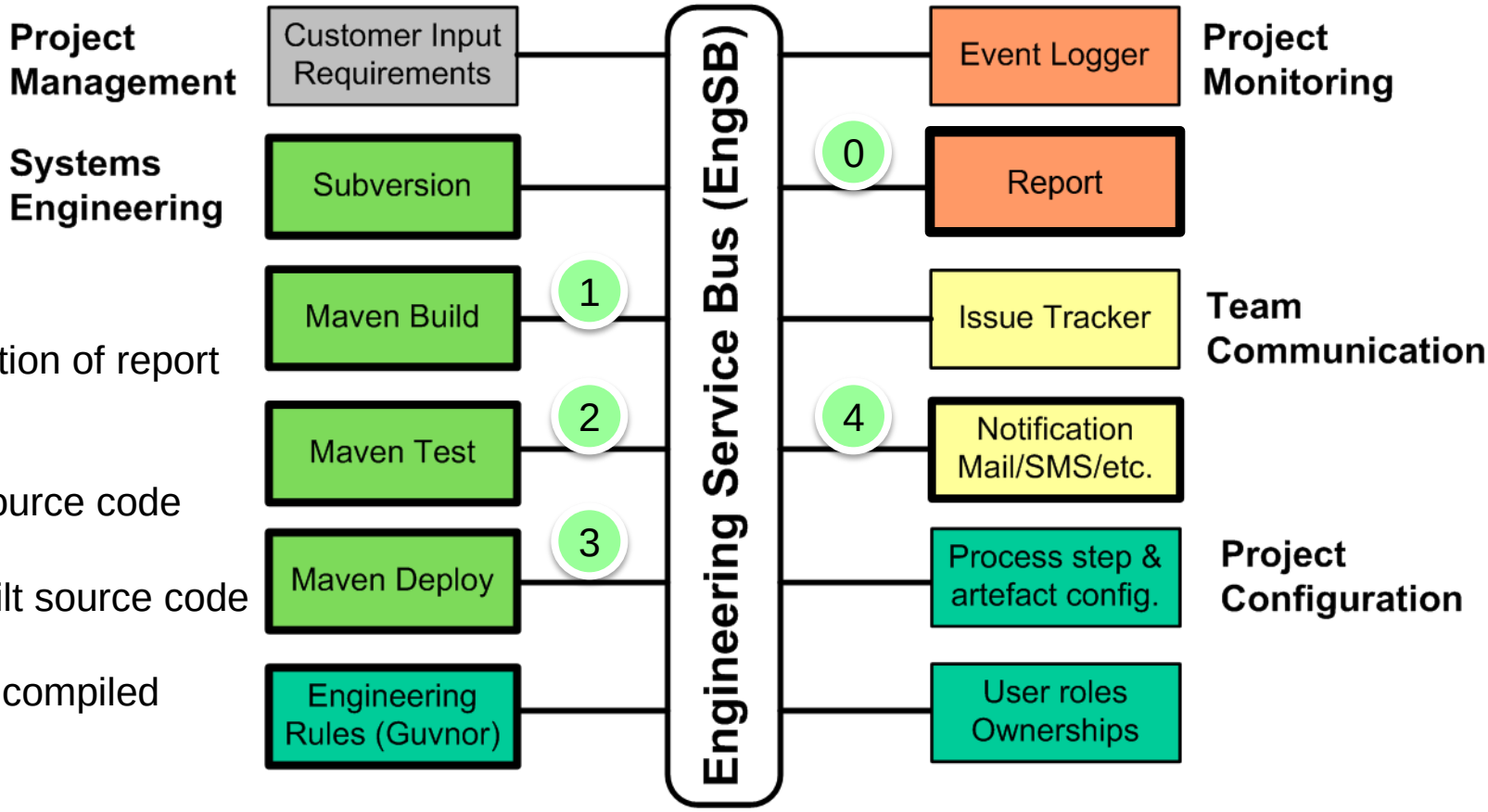
1. Build the source code

2. Test the built source code

3. Deploy the compiled source code

4. Send notification about result to a configured list of recipients.

Components of the OpenEngSB in the CI & T use case



0. Start collection of report information

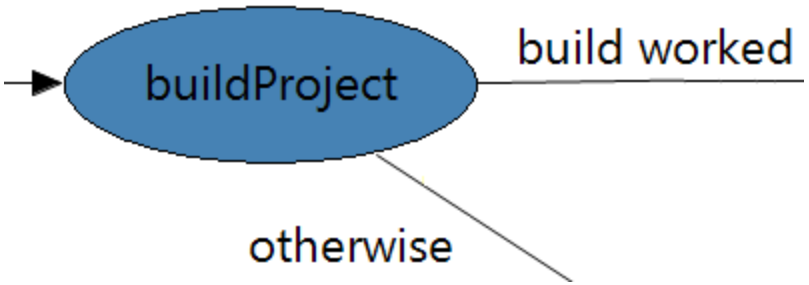
1. Build the source code

2. Test the built source code

3. Deploy the compiled source code

4. Send notification about result to a configured list of recipients.

Code examples

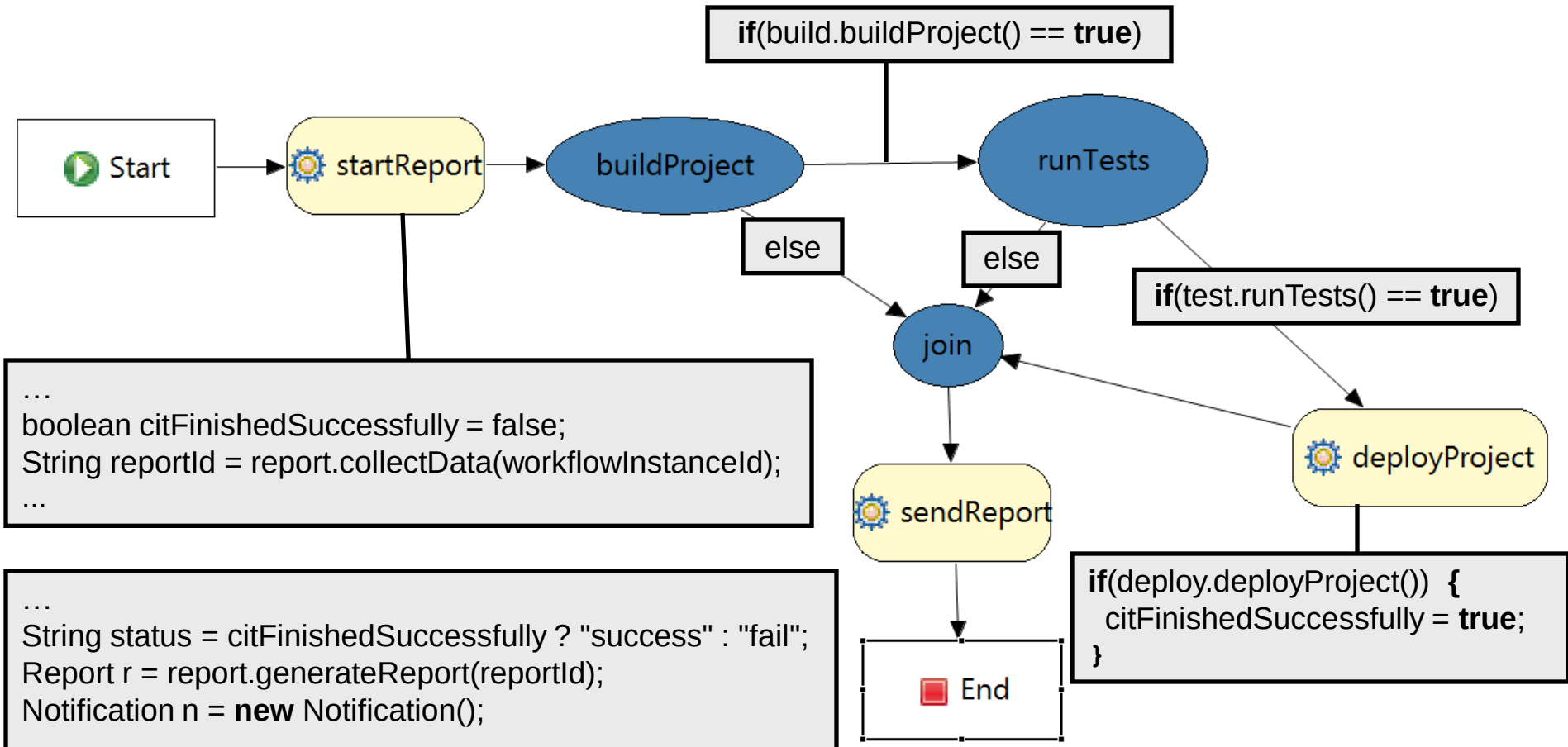


- Build-step code in switch node of Drools Flow graph

```
return build.buildProject();
```

  - Sends the service request to the build domain, which informs the responsible tool (in our case Maven)
  - Returns whether the build step was successful
  - Based on the result of the build step it can be decided whether the test and deploy steps should be performed or the CI & T process should be stopped.

The Ci & T Process in Drools Flow
Detailed description




```
…
boolean citFinishedSuccessfully = false;
String reportId = report.collectData(workflowInstanceId);
...
```

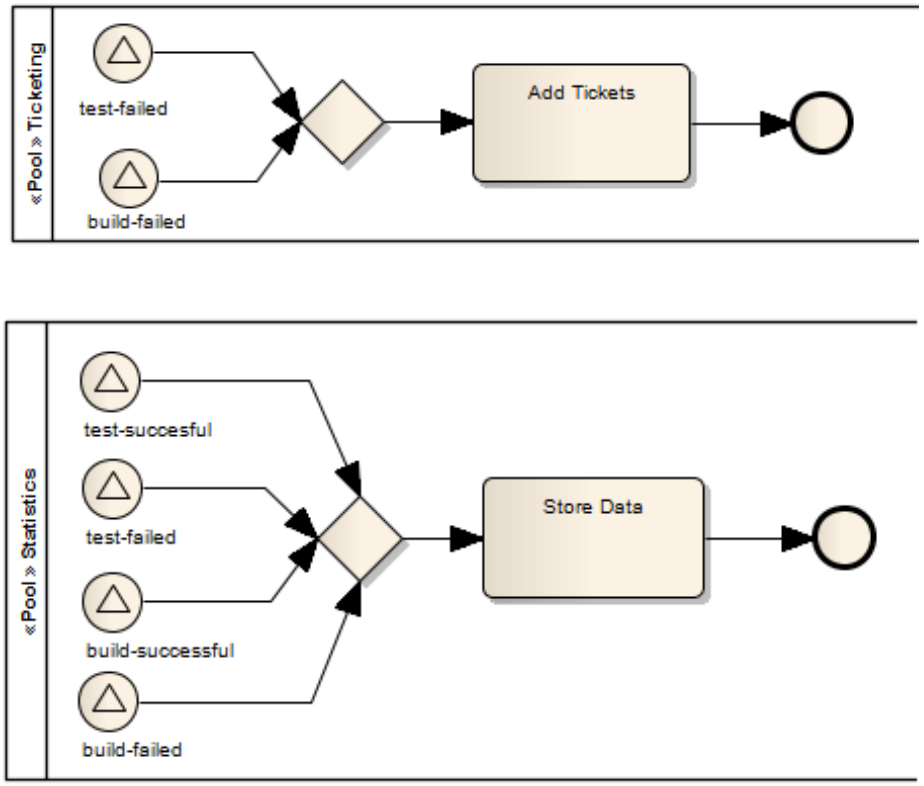
```
…
String status = citFinishedSuccessfully ? "success" : "fail";
Report r = report.generateReport(reportId);
Notification n = new Notification();
…
n.setMessage("Cit process finished with result: "+status);
n.addAttachment(r);
…
notification.notify(n);
```

```
if(deploy.deployProject()) {
  citFinishedSuccessfully = true;
}
```

Flexible Process Extension for CI&T

Process extensions

- Add **issue ticket** functionality
  - Trac tool instance
- Add **logging** functionality
  - Calculate project statistics over several projects.
- Add **“conditional build failure”**
  - Build should fail only if a failed test was successful before.



Technology-independent extension

- **Event-driven** extension
  - Add new event listeners
- Tool evolution behind **tool domain interface**
  - Different kinds of notification

Process Customization

- A process engineer can change the process by
  - editing the overall workflow using the graphical drools flow editor
  - editing what happens in each step, which means changing the code in the nodes of the workflow
  - defining rules that react to the events triggered by the workflow and are thus more independent from the workflow
  - configuring the tools and tool connectors
  - configuring the tool domain

Process Customization by Drools Rule Example

- Rule to create an issue if build, test or deploy fails

```
package org.openengsb
rule "createIssue"
when
  e : BuildEvent(buildSuccessful == false) or
  e : TestEvent(testRunSuccessful == false) or
  e : DeployEvent(deploySuccessful == false)
then
  issue.createIssue("cit step '" +e.getDomain() + "' failed");
end
```

  - If one of these three events happens and the process step was not successful create an issue.

Lessons learned

- Evidence from the prototype
  - **Successful reproduction** of continuous integration process on OpenEngSB.
  - OpenEngSB allows prototyping new variants of software engineering processes **more open, flexibly, and transparent** than rigid CI tools.

- Key benefits
  - **Tool domains** simplify exchanging tool instances
  - **Flexible extension of workflow** and tool instance logic

- Effort of integration
  - **Integration of a tool** with well-documented API took 1 to 2 days
  - **Process implementation** effort depends on process complexity; expect days for technical work for a sufficiently well-defined process.

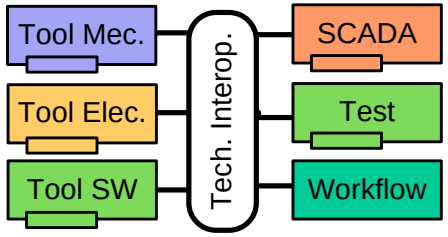
- Limitations
  - **Added complexity** to the tool environment from new middleware layer that needs **configuration and administration.**

Summary

- Complex software-intensive systems raise need for **engineering process automation**.
- **Flexible integration of engineering tools** and systems along the lifecycle is a foundation for better process automation and quality management.

- Even initial **Engineering Service Bus** (EngSB) implementations bring the foundation for
  - **Flexible (software+) engineering process prototyping**
  - **Awareness in the team** on relevant changes in the project environment
  - **Data collection and analysis** for quality assurance.

- Future Work
  - Collaboration of federated EngSBs
  - **Engineering model synchronization** and **defect detection** across tools.



Backup Slides

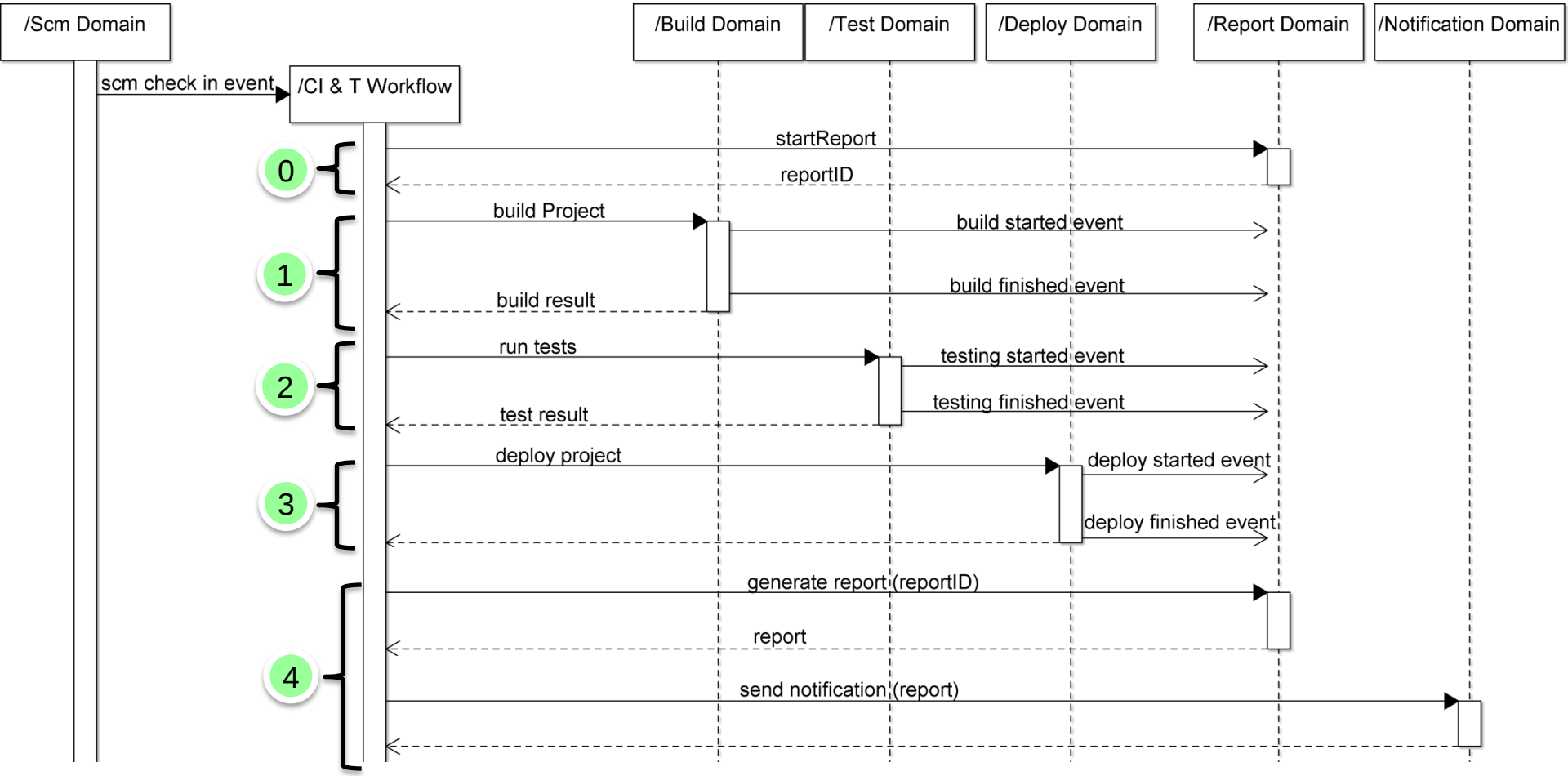
Design of Messages

- OpenEngSB uses XML as message format
- Predefined header that all message have to carry
  - Context-ID
  - Correlation-ID
  - Workflow-ID [only message in workflows]
  - Workflow Instance-ID [only message in workflows]
- Payload in a standardized format

ScmCheckInEvent Message

```
<?xml version="1.0" encoding="UTF-8"?>
<list xmlns="http://org.openengsb/util/serialization" name="event" … >
  <text name="event" … >org.openengsb.drools.events.ScmCheckInEvent</text>
  <list name="superclasses" … >
    <text name="superclass" … > org.openengsb.core.model.Event</text>
    <text name="superclass" … >java.lang.Object</text>
  </list>
  <text name="name" … >scmCheckInEvent</text>
  <text name="domain" … >scm</text>
</list>
```

Message Flow during the CI&T Process



0. Start collection of report information	1. Build the source code	2. Test the built source code	3. Deploy the compiled source code	4. Send notification about result to a configured list of recipients.