Spock & Groovy Test Code Style and Conventions

Use of Spock Blocks and Labels

1. The test 'def' should describe a high level scenario (not set up details or expected outcome)
2. Each label should have a description.The label plus description should form a readable English sentence preferring high level (Product Owner) type language over code level langue i.e. no method names etc.
   (since the label is the first word of the sentence the description should start with a lower case character)
3. All blocks should be separated by a blank line except and 'and: ' block which really is a continuation of the previous block
4. Given-when-then v. expect. Given-when-then is the preferred format but expect block is useful in situations where it 'is more natural to describe stimulus and expected response in a single expression' see also the Spock Primer

Expect and Labels example

```
@Unroll
def 'Celltrace scanner activation request with invalid ropPeriod value of  #ropPeriod'() {
    given: 'scanner details with an invalid rop period'
        eventHeaders = eventHeadersForCellTrace(ropPeriod, 'STREAMING')

    expect: 'the outbound request XML contains the default rop period of "FIFTEEN_MIN"'
        activateEventJobRequest.getRequest(eventHeaders).contains("<reportingPeriod>FIFTEEN_MIN<
/reportingPeriod>")

    where: 'rop period is any of these invalid values'
        ropPeriod << [ null, 0, -1, 123 ]

}
```

5. Expected outputs in Spock data tables should be separated with a double pipe symbol (||) to visually set them apart.

Some more tips from the Spock Primer :

- "Try to keep the number of conditions per feature method small. One to five conditions is a good guideline. If you have more than that, ask yourself if you are specifying multiple unrelated features at once. If the answer is yes, break up the feature method in several smaller ones. If your conditions only differ in their values, consider using a data table."
- Leverage Groovy JDK methods like any() and every() to create more expressive and succinct conditions.

Mixing of Java & Groovy

Groovy files should be 'pure' groovy as far as possible, which would imply some rules like

1. No access modifiers
2. No semi-colons
3. To prevent conflict between java and groovy style code groovy classes should use groovy style methods using 'def'.
   In some cases it might be useful (more readable) to specify the return type

Groovy method examples

```
def eventHeadersForCellTrace(ropPeriod, outputMode, eventDetails) {
    return [nodeAddress        : 'NetworkElement=LTE07dg2ERBS00001',
            SUBSCRIPTION_TYPE  : 'CELLTRACE',
            eventDetails       : eventDetails,
            ropPeriod          : ropPeriod,
            scannerId          : '10003',
            eventProducerId    : 'Lrat',
            pmEventMxmlns      : 'urn:com:ericsson:ecim:RcsPMEventM',
            managedElementxmlns: 'urn:com:ericsson:ecim:ComTop',
            outputMode         : outputMode,
            ueFraction         : 1000,
            isAsnEnabled       : true,
            streamInfoAddress  : '1.2.3.5',
            streamInfoPort     : '14']
}

def clickCreateSubscription(final String subscriptionType) {
    final String selector = getSubscriptionTypeSelector(subscriptionType)
    0..4.each { falseOnException { attemptClick(selector) } }
}

Map<String, String> getCreateSubscriptionDropdownOptions() {
    click(createSubscriptionDropdown)
    def options = [:]
    def elements = root.findElements(By.xpath("//div[contains(@class,'ebComponentList-item')]"))
    for (def i = 1; i < elements.size(); i++) {
        options.put(elements.get(i).text, elements.get(i).text)
    }
    click(createSubscriptionDropdown)
    return options
}
```

4. All strings should be enclosed using single apostrophes, unless it is a Groovy template (see https://stackoverflow.com/questions/6761498/whats-the-difference-of-strings-within-single-or-double-quotes-in-groovy)
   please note #parameter in Spock labels does not rely on GString functionality, so labels can be enclosed in single apostrophes too.

Label example

```
then: 'the task status is #expectedTaskStatus'
     taskStatus == expectedTaskStatus
where:
     subscriptionEvents | cellTraceCategory           | scannerStatus        || expectedTaskStatus
     events             | 'CELLTRACE_AND_EBSL_STREAM' | ScannerStatus.ACTIVE || TaskStatus.OK
```

Anti patterns

1. Test duplication instead of using of data table or data pipes
   see this (corrected) example in Gerrit: https://gerrit.ericsson.se/#/c/4939004/6..12/com-ecim-pm-eventsm-operation-handler-code-jar/src/test/groovy/com/ericsson/oss/mediation/pm/handlers/request/ActivateEventJobRequestSpec.groovy

2. Abuse of Spock tables; combining many tests (for different behavior) into one large table
   a. Bad behavior example: https://gerrit.ericsson.se/#/c/4767553/37/pm-flow-scheduler-handler-jar/src/test/groovy/com/ericsson/oss/mediation/pm/handlers/PmDataRetrievalHandlerSpec.groovy
   b. Bad behavior corrected: https://gerrit.ericsson.se/#/c/4809899/19/common-pm-mediation-router-policy-test-jar/src/test/groovy/com/ericsson/oss/mediation/pm/router/policy/rebalance/operators/PmRebalanceOperatorSpec.groovy

3. JUnit legacy behavior : All conditions and expectations in the test name/scenario
   Instead of using the given-when-then label descriptions the developer puts all those in the scenario/test title.

SonarQube Rule Candidates

SonarQube will be used to enforce some of the agreed conventions where (automation is) possible. The intention is that any violations in new or modified java code will 'break the build' i.e. give a '-1' score on the Gerrit commit like currently used for Java, Java Script etc.Some rules off lesser importance will be set to info-level so they will not break the build but the developer still gets informed about the preferred code style.

#	Description	Level	(related) Existing SQ Rule	Comments
1	Unnecessary modifiers	Break-the-build	https://sonarqube.lmera.ericsson.se/coding_rules#rule_key=grvy%3Aorg.codenarc.rule.unnecessary.UnnecessaryPublicModifierRule	**The existing SQ rule only covers 'public' we would have to extend that**
2	Unnecessary semicolon	Info		**Groovy classes do not need semi-colon line separators**
3	Unnecessary use of GString	Info	https://sonarqube.lmera.ericsson.se/coding_rules#rule_key=grvy%3Aorg.codenarc.rule.unnecessary.UnnecessaryGStringRule	**unnecessary GString usage does not incur any performance cost but its was agreed look and feel of the code is enough reason to flag it (at info level)**
3	Unnecessary return	Info	https://sonarqube.lmera.ericsson.se/coding_rules#rule_key=grvy%3Aorg.codenarc.rule.unnecessary.UnnecessaryReturnKeywordRule	**In Groovy, the return keyword is often optional**
4	All Spock labels should have descriptions	Break-the-Build	N/A	**Would have to code a new SQ Rule**
5	All Spock blocks should be followed by a blank line ("and:" does not start a new block)	Info	N/A	**Would have to code a new SQ Rule**
6	Spock test definitions annotated with @Unroll should have at least one #parameter to distinguish each run	Break-the-Build	N/A	**Would have to code a new SQ Rule**

Some Complete Examples

Complete Example 1

```
    @Unroll
    def 'Activation tasks with #scannerName scanner'() {
        given: 'one RadioNode'
            createNode()
        and: 'CCTR subscription ContinuousCellTraceSubscription for Lrat with 1 RadioNode and has 2 events with
no event producerId'
            createCctrSubscription(events, 'ContinuousCellTraceSubscription',[radioNodeMO])
            def expectedSubscriptionId = expecedNumberOfNotifications == 0 ? 0 : subscription.id
        and: 'one INACTIVE HIGH_PRIORITY_CELLTRACE scanner'
            scannerUtil.builder(scannerName,radioNodeMO.getName())
                       .status(ScannerStatus.INACTIVE)
                       .processType(ProcessType.HIGH_PRIORITY_CELLTRACE).build()

        when: 'createMediationTaskRequests is invoked'
            def tasks = cctrSubscriptionActivationTaskRequestFactory.createMediationTaskRequests(subscription.
nodes, subscription, true)

        then: '#expectedNumberOfTasks mediation tasks created'
            expectedNumberOfTasks == tasks.size()
        and: 'Verify size of the tracker cache is 1'
            expectedTrackerCacheSize == pmicInitiationTrackerCache.allTrackers.size()
        and: 'Verify there is one notification for every scanner to be activated'
            expecedNumberOfNotifications == (pmicInitiationTrackerCache.allTrackers.size() > 0
                    ? pmicInitiationTrackerCache.getTracker(subscription.idAsString).
totalAmountOfExpectedNotifications : 0)
        and: 'Scanner assigned correct subscription id'
            expectedSubscriptionId == scannerDao.findAll()[0].subscriptionId


        where: 'the scannername is one of the following'
             scannerName                || expectedNumberOfTasks | expecedNumberOfNotifications   |
expectedTrackerCacheSize
            'PREDEF.10005.CELLTRACE'    || 1                     | 1                              |   1
            'PREDEF.DU.10005.CELLTRACE' || 0                     | 0                              |   0
    }
```

External References

- http://groovy-lang.org/documentation.html
- http://spockframework.org/
- http://groovy-lang.org/style-guide.html