Thoughts on Software Quality

Is it just a cultural thing?
Are we doomed?

Bertrand Delacrétaz
@bdelacretaz, grep.codeconsult.ch
Principal Scientist, Adobe Basel
Apache Software Foundation Member and Director
ApacheCon:Core Europe, November 2016
created in 2007
powers "a few" major websites
still going strong
something must be right
(and it’s just one example among many)
how can we create a culture of quality in our teams?
Quality is usually visible from far away

Engineering culture?
Good tools for good craftsmen
Defensive & Conservative
Avoid broken windows
Do one thing and one thing well
Cultural traits? you can’t measure those…

automate!

defensive

no broken windows

done one thing and one thing well

conservative

obvious quality, measurable where possible
so, quality is a cultural thing?
maybe a few axes will help us measure
Code coverage? Not a silver bullet.

```java
public class MathRandomTest {

    @Test
    public void testRandomIntGenerator() {
        final Random r = new Random();
        for(int i=0; i < 100000000; i++) {
            r.nextInt();
        }
    }
}
```

Code coverage: 100%
Semantic coverage: zero!
A generator with `return 42;` would pass.
Not useless...but just one tool among others.
quality is much wider than that
Happy users!
Happy Developers!
“Conceived to help motorize the large number of farmers still using horses and carts in 1930s France” (Wikipedia: Citroën 2CV)
Simplicity & adequation

```
$ cat $(find bundles/engine -name *.java) | wc -l
  13841
$ cat $(find bundles/resourceresolver -name *.java) | wc -l
  18273
$ cat $(find bundles/servlets -name *.java) | wc -l
  19404
$ cat $(find bundles/api -name *.java) | wc -l
  13938
```

```
Sling Core
standard
custom
resourc
servlet
JSR
Felix
OSGi
webD
OSGi framework
(Apache Felix)
```

**Sling**

**Discover Sling in 15 minutes**

The Sling Launchpad is a ready-to-run Sling configuration, providing an embedded JCR content repository and web server, a selection of Sling components, documentation and examples. The Launchpad makes it easy to get started with Sling and to develop script-based applications.

This page will help you get started with the Launchpad. Fifteen minutes should be enough to get an overview of what Sling does.

While simple to run and understand, the Launchpad is a full-featured instance of Sling, an example configuration that we have created with the most common modules and configurations. The full functionality of Sling is available by loading additional Sling (or custom) OSGi bundles as needed, using the Launchpad's web-based OSGi management console.
Robustness
Building Apache Sling Launchpad Testing 7-SNAPSHOT

Tests run: 529, Failures: 0, Errors: 0, Skipped: 1

BUILD SUCCESS

Total time: 02:08 min
the best way to avoid risk is to do nothing, of course. Nobody will steal this picture, but it’s quite a boring slide.
Security

Reporting New Security Problems with Apache Sling

The Apache Software Foundation takes a very active stance in eliminating security problems and denial of service attacks against Apache Sling.

We strongly encourage folks to report such problems to our private security mailing list first, before disclosing them in a public forum.

Please note that the security mailing list should only be used for reporting undisclosed security vulnerabilities in Apache Sling and managing the process of fixing such vulnerabilities. We cannot accept regular bug reports or other queries at this address. All mail sent to this address that does not relate to an undisclosed security problem in the Apache Sling source code will be ignored.

If you need to report a bug that isn’t an undisclosed security vulnerability, please use our public issue tracker.

Questions about:
- how to configure Sling securely
- whether a published vulnerability applies to your particular application
- obtaining further information on a published vulnerability
- availability of patches and/or new releases

should be addressed to our public users mailing list. Please see the Project Information page for details of how to subscribe.

The private security mailing address is: security(at)sling.apache.org.

Note that all networked servers are subject to denial of service attacks, and we cannot promise magic workarounds to generic problems (such as a client streaming lots of data to your server, or re-requesting the same URL repeatedly). In general our philosophy is to avoid any attacks which can cause the server to consume resources in a non-linear relationship to the size of inputs.
scarce resources help to focus!
Backwards compatibility
Transparency

REST culture!
Transparency

REST -> naturally transparent
Good names
Helpful messages
Introspection tools

```bash
$ curl -u admin:admin -Ffoo=bar http://localhost:8080/testing12
<html>
<head>
  <title>Content created /testing12</title>
</head>
<body>
  <h1>Content created /testing12</h1>
</body>
```
Good names?

AbstractResource
AbstractResourceVisitor
AccessSecurityException
Adaptable
AdapterFactory
AdapterManager
AttributableResourceProvider
Authenticator
CompositeValueMap
DeepReadValueMapDecorator
DynamicResourceProvider
ExternalResourceListener
HtmlResponse
SlingAllMethodsServlet
SlingBindings
SlingConstants
SlingException
SlingHttpServletRequest
SlingHttpServletRequestWrapper
SlingHttpServletResponse
SlingHttpServletResponseWrapper
SlingIOException
LoginException
ModifiableValueMap
ModifiableValueMapDecorator
ModifyingResourceProvider
NoAuthenticationHandlerException
NonExistingResource
ObservationReporter
ObserverConfiguration
OptingServlet
ParametrizableResourceProvider
PersistableValueMap
PersistenceException
PropertyBuilder
ProviderContext
RefreshableResourceProvider
RequestDispatcherOptions
RequestParameter
RequestParameterMap
RequestPathInfo
RequestProgressTracker
RequestUtil
ResolveContext
Resource
ResourceAccessSecurity
ResourceChange
Low coupling

sailboats are very cool but we shouldn’t take their high coupling as an example…
quality has many dimensions often hard to measure
so, how is Sling doing?
I’d say not bad
thanks to a strong community with a culture of quality in design, implementation and testing.
You can’t measure it all: quality is multi-dimensional and needs to be part of your team’s culture.

Thinking and reviewing along those axes can help build such a culture.

And in the end the only thing that really matters is happy users…