XSpec v0.5.0

Sandro Cirulli

XSpec and Oxford University Press

<sandro.cirulli@oup.com>

Abstract

XSpec is an open source unit test and behaviour driven development framework for XSLT and XQuery. XSpec v0.5.0 was released in January 2017 and included new features such as XSLT 3.0 support and JUnit report for integration with continuous integration tools. The new release also fixed long standing bugs, provided feature parity between the Windows and MacOS/Linux scripts, integrated an automated test suite, and updated the documentation. XSpec v0.5.0 is currently included in oXygen 19.0.

This paper highlights the importance of testing, provides a brief history of the XSpec project, describes the new features available in XSpec v0.5.0 and the work currently under development for future releases, and reports the effort of the XML community to revive this open source project.

Keywords: XSpec, Unit Testing, Behaviour Driven Development, Continuous Integration

1. The Importance of Testing

Testing is a fundamental part of writing software that aims to be robust, reliable, and maintainable. In fact, testing can be considered as a promise made to customers and users that the code behaves as intended. Writing tests regularly also improves the code base as it forces developers to write smaller units of code that can be more easily tested, debugged, and maintained. Finally, testing acts as self-documentation and can help other developers to understand and modify existing code.

Testing plays a central role in software development practices such as extreme programming (XP) and test-driven development (TDD) as well as in agile methodologies like Scrum and Kanban. For example, in test-driven development, unit tests (i.e. tests for individual units of code such as a function or a method) are usually written by developers as they write their code in order to make sure that new features work according to specifications and bug fixes do not break other parts of the code base. Unit tests increase the overall quality and maintainability of the code and it has been estimated that unit tests alone contribute to removing an average of 30% of defects present in software [1].

Although testing is important for any serious software developer, there aren't many testing tools for XSLT and XQuery when compared to other programming languages. Furthermore, their use is not yet very widespread. Back in 2009 Tony Graham [2] made an inventory of all the available testing frameworks available for XML technologies - most of which are unfortunately not actively developed any more. XSpec aims to fill this gap by offering a testing framework and raising awareness about testing in the XML community.

While any piece of XSLT and XQuery code can be tested using XSpec, the greatest return on investment occurs when testing code that gets called and reused frequently. Functions and named scenarios are perfect fits for unit testing as they are self-contained pieces of code upon which other parts of the code base may rely.

Integration with other testing and automation tools is also a key part of testing frameworks as unit tests are typically triggered automatically on events such as commits to the code base or software builds. Software development practices such as continuous integration (CI) popularized the importance of integrating development work into the code base on a daily basis, running test suites automatically, and providing developers with fast feedback when new code breaks tests or software builds. As a result, debugging and fixing bugs at the early stages of development improves the productivity of software developers and reduces the overall risk and cost of code releases and software maintenance.

2. A Brief History of XSpec

XSpec was created by Jeni Tennison in 2008 and is inspired by the RSpec testing framework for Ruby. Jeni Tennison presented XSpec at XML Prague 2009 [3] and released it as open source under the MIT license on Google Code. XSpec v0.3.0 was integrated in oXygen 14.0 in 2012 and this helped to spread its use and raise awareness about testing among XSLT developers.

doi:10.14337/XMLLondon17.Cirulli01

The project was maintained and expanded by Florent Georges who released v0.4.0-RC in 2012. Unfortunately active development stagnated between 2012 and 2015 with no further releases. The code base was moved from Google Code (now defunct) to GitHub in 2015.

I started contributing actively to XSpec in 2016 in order to fix an old bug and add a new feature I implemented at work. I forked the project and after few months I transferred my fork to the XSpec organisation repository under https://github.com/xspec/xspec. To my surprise, several people started contributing by raising issues and sending pull requests. This recreated an XSpec community that will hopefully sustain the project in the long term. This process culminated in release v0.5.0 in January 2017.

XSpec is under active development and new features and bug fixes are regularly merged into the master branch as soon as they are available and pass the test suite. For those who prefer a more stable version, the latest release is available as a zip file and can be retrieved from the official release page on GitHub [4].

3. New Features

A selection of the new features released in XSpec v0.5.0 is presented here. The full list of new features is available in the official release notes [4]. New features come with a test that makes sure that the feature behaves according to the specifications. Tests also act as documentation to show how the feature is implemented and are often used as examples in the documentation available on the official wiki [5].

3.1. XSLT 3.0 Support

XSpec now supports XSLT 3.0 [6]. This patch was provided by oXygen which first integrated it in its XML editor.

To illustrate XSLT 3.0 support, Example 1, "XSLT 3.0 Example" shows an example of XSLT that makes use of the inline function expression available in XPath 3.0 [7]:

Example 1. XSLT 3.0 Example

```
<xsl:stylesheet
  xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
  xmlns:xs="http://www.w3.org/2001/XMLSchema"
  exclude-result-prefixes="xs" version="3.0">

  <xsl:template name="supportXPath3">
    <root>
      <question>
        <xsl:text>Does XSpec
                  support XPath 3.0?</xsl:text>
      </question>
      <answer>
        <xsl:value-of select="
          let $answer := 'Yes it does'
          return $answer"/>
      </answer>
    </root>
  </xsl:template>

</xsl:stylesheet>
```

The template can be tested using the XSpec test in Example 2, "XSpec Test for XSLT 3.0". Note the use of `@xslt-version` specifying the version of XSLT (when `@xslt-version` is not provided, XSpec uses XSLT 2.0 by default).

Example 2. XSpec Test for XSLT 3.0

```
<x:description
  xmlns:x="http://www.jenitennison.com/xslt/xspec"
  stylesheet="xspec-xslt3.xsl" xslt-version="3.0">

  <x:scenario label="When testing the inline
                     function expression in XPath 3">

    <x:call template="supportXPath3"/>

    <x:expect label="it returns the expected answer">
      <root>
        <question>Does XSpec
                  support XPath 3.0?</question>
        <answer>Yes it does</answer>
      </root>
    </x:expect>

  </x:scenario>
</x:description>
```

3.2. JUnit Support

JUnit [8] is a popular unit testing framework for Java. JUnit reports are XML-based and are understood natively by popular continuous integration servers such as Jenkins.

In the past XSpec reports where only available in XML and HTML. XSpec v0.5.0 introduced JUnit reports which can be easily generated with the `-j` option from the command line as illustrated in Example 3, "Run XSpec with JUnit Option" (the sample file `escape-for-regex.xspec` is available in the tutorial folder on GitHub):

Example 3. Run XSpec with JUnit Option

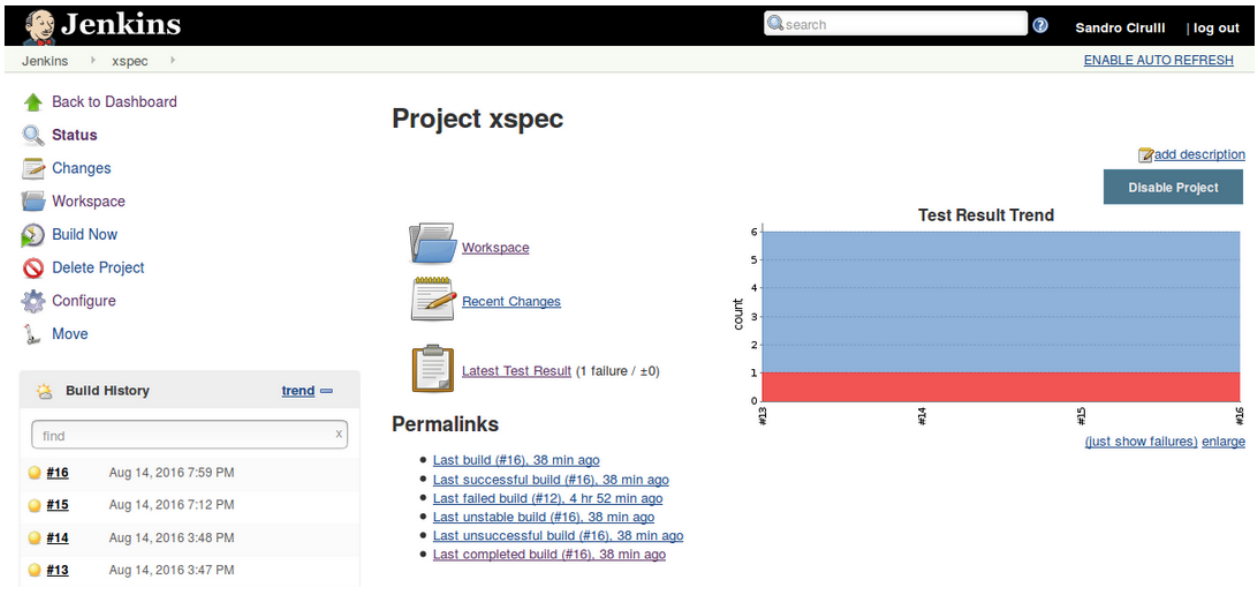
```
/bin/xspec.sh -j tutorial/escape-for-regex.xspec
```

Example 4, "JUnit Report" shows an example of the generated JUnit report with a successful and a failing test:

Example 4. JUnit Report

```
<testsuites>
  <testsuite name="When processing a list of phrases" tests="2" failures="1">
    <testcase name="All phrase elements should remain" status="passed"/>
    <testcase name="Strings should be escaped and status attributes should
                    be added" status="failed">
      <failure message="expect assertion failed">&lt;x:expect
        xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
        xmlns:xs="http://www.w3.org/2001/XMLSchema"
        xmlns:test="http://www.jenitennison.com/xslt/unit-test"
        xmlns:x="http://www.jenitennison.com/xslt/xspec"
        xmlns:functx="http://www.functx.com"&gt;
            &lt;phrases&gt;
               &lt;phrase status="same"&gt;Hello!&lt;/phrase&gt;
               &lt;phrase status="same"&gt;Goodbye!&lt;/phrase&gt;
               &lt;phrase status="changed"&gt;\(So long!\)&lt;/phrase&gt;
            &lt;/phrases&gt;
         &lt;/x:expect&gt;
      </failure>
    </testcase>
  </testsuite>
</testsuites>
```

Figure 1. Test Result Trend in Jenkins



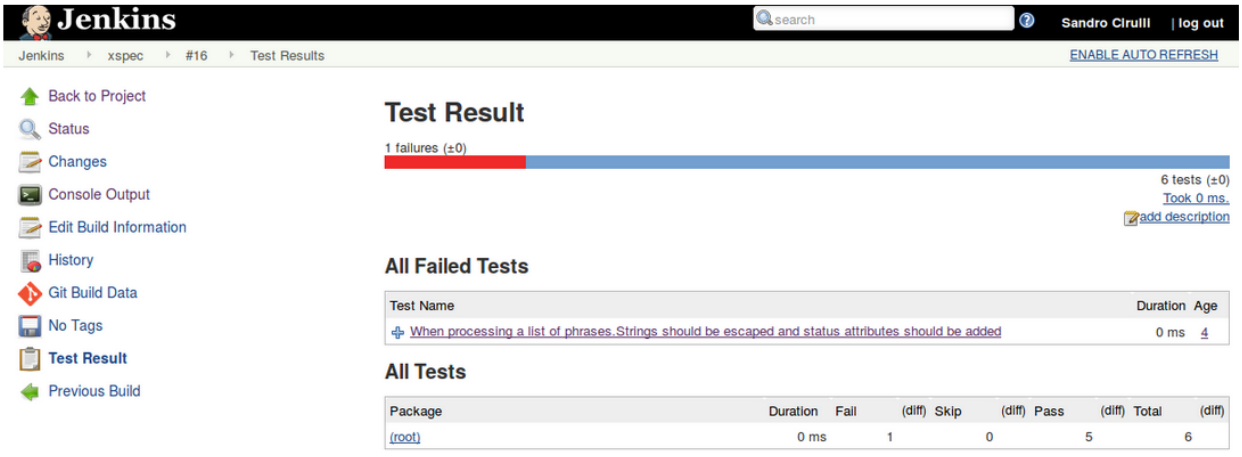
Note that the generation of JUnit reports requires Saxon 9 EE or Saxon 9 PE as the implementation makes use of Saxon extension functions.

JUnit reports can be easily plugged into continuous integration tools that understand JUnit natively such as Jenkins. The XSpec documentation describes how to configure Jenkins to run XSpec tests and generate JUnit reports [9]. Figure 1, "Test Result Trend in Jenkins" and Figure 2, "Test Result in Jenkins" show a test result trend and details of a failing test auto-generated by Jenkins from JUnit reports.

3.3. Testing XSpec

XSpec itself is tested using a mix of XSpec tests and shell and batch scripts. The test suite is executed automatically on online continuous integration servers (Travis for Linux and AppVeyor for Windows) every time a pull request or a code merge are initiated. This allows to spot regression bugs as soon as they appear and makes code reviews and approval of pull requests quicker and safer. In addition, testing XSpec with continuous integration tools such as Travis and AppVeyor provides example configuration and documentation for other projects that

Figure 2. Test Result in Jenkins



wish to use XSpec to run tests in continuous integration workflows.

3.4. Feature Parity between Windows and MacOS/Linux

XSpec can be executed from the command line using a batch script in Windows or a shell script in MacOS/Linux. Historically, the batch script lagged behind and did not provide all the options available in the shell script. XSpec 0.5.0 ships with a brand new version of the batch script that fully supports existing and new command line options available in the shell script. In addition, a test suite for the batch script is now executed on every new commit thus providing the same level of testing available for the shell script.

4. Bug Fixes

The full list of bug fixes is available in the official release notes [4]. Most bug fixes come with a test that makes sure that future code changes do not introduce regression bugs.

As example of defects fixed in this release, it is worth mentioning the code coverage bug. XSpec relies on Saxon extension functions for the implementation of the code coverage option that allows to check which parts of the XSLT code are covered by XSpec tests. This functionality was broken for several years due to a change in the implementation of the `TraceListener` interface between Saxon 9.2 and 9.3 and the bug was flagged by several users [10] [11].

This long standing issue has been fixed in v0.5.0 and the code coverage now works with the recent versions of Saxon EE and PE (extension functions are only available with these two versions of Saxon). Documentation on how to use the code coverage functionality is now available on the official wiki page [12].

5. Future Work

XSpec users can raise issues and feature requests on the official issue tracker on GitHub and contribute with pull requests. Some of the work that is currently under development or scheduled for the future releases of XSpec includes:

- **Schematron support**: A feature request was raised in order to have Schematron support in XSpec. This included use cases such as writing XSpec tests for Schematron rules and schemas. Vincent Lizzi provided a pull request for Schematron support in XSpec and demoed it during an open session at JATS-Con in April 2017. As I write these lines, the pull request has just been merged into the main XSpec code base and documentation will soon be available on the wiki. Schematron users are invited to test this new functionality and provide feedback.
- **Full XQuery support**: although XSpec allows to test both XSLT and XQuery, XQuery support is often lagging behind or untested. This work aims to bring full feature parity between XSLT and XQuery and to provide tests and documentation covering XQuery. A tutorial on how to write XSpec tests for XQuery is in the process of being written and will soon be available in the official documentation.
- **Harmonisation with oXygen**: XSpec is integrated by default in oXygen but some features such as the output report and the ant configuration are implemented differently. This work aims to harmonize XSpec so that the version provided in XSpec is the same version available on GitHub.

6. Conclusion

Testing is a crucial part of software development and XSpec aims to provide XSLT, XQuery, and Schematron developers with a testing framework for making their code more robust, reliable, and maintainable. After few years of stagnation, active development of XSpec restarted and culminated in the release of v0.5.0 in January 2017. This new release included several new features and fixed long standing bugs. Being an open source project, XSpec is developed and maintained by an active community gathering around the GitHub repository at https://github.com/xspec/xspec and welcomes new and existing users to contribute with issues, questions, and pull requests.

7. Acknowledgements

I would like to thank Jeni Tennison for creating XSpec back in 2008 and releasing it under an open source licence. I'm also deeply indebted to Florent Georges for maintaining the project in the past years and to Tony Graham for his support during the migration to GitHub. I own my deepest gratitude to the XSpec community who contributed to this release and provided me with encouragement and support, their GitHub user names are listed in the official release notes. A special thank you

to AirQuick - I ignore his real name - whose many pull requests, comments, and code reviews have been extremely valuable for the development of XSpec.

Bibliography

[1] Steve McConnell. 2006. *Software Estimation: Demystifying the Black Art*. Microsoft Press. Redmond, Washington.
ISBN 978-0735605350.

[2] Tony Graham. *Testing XSLT*. In Conference Proceedings of XML Prague 2009. March 21-22, 2009.
http://archive.xmlprague.cz/2009/presentations/XMLPrague_2009_proceedings.pdf#page=83

[3] Jeni Tennison. *Testing XSLT with XSpec*. In Conference Proceedings of XML Prague 2009. March 21-22, 2009.
http://archive.xmlprague.cz/2009/presentations/XMLPrague_2009_proceedings.pdf#page=105

[4] XSpec. *XSpec v0.5.0*.
https://github.com/xspec/xspec/releases/tag/v0.5.0
Accessed: 5 May 2017.

[5] XSpec. *XSpec Documentation Wiki*.
https://github.com/xspec/xspec/wiki
Accessed: 5 May 2017.

[6] World Wide Web Consortium (W3C). *XSL Transformations (XSLT) Version 3.0*. Michael Kay.
http://www.w3.org/TR/xslt-30/

[7] World Wide Web Consortium (W3C). *XML Path Language (XPath) 3.0*. Jonathan Robie, Don Chamberlin, Michael Dyck, and John Snelson. 8 April 2014.
http://www.w3.org/TR/xpath-30/

[8] JUnit. *JUnit*.
http://junit.org
Accessed: 5 May 2017.

[9] XSpec. *Integration with Jenkins*.
https://github.com/xspec/xspec/wiki/Integration-with-Jenkins
Accessed: 5 May 2017.

[10] XSpec Users Google Group. *XSpec 0.3.0*.
https://groups.google.com/forum/#!topic/xspec-users/0BIzNfFv4-Y
Accessed: 5 May 2017.

[11] Sandro Cirulli. *Continuous Integration for XML and RDF Data*. In Conference Proceedings of XML London 2015. June 6-7, 2015.
doi:10.14337/XMLLondon15.Cirulli01

[12] XSpec. *XSpec Code Coverage*.
https://github.com/xspec/xspec/wiki/Code-Coverage
Accessed: 5 May 2017.