Lessons Learned from Starting an Open Source Based Compliance Verification Program

Wenjing Chu
Senior Director, Open Source and Research
Futurewei Technologies, Inc.
Many industries gain tremendous value from standardization through a formal process, often overseen by well established institutions.

- E.g. 3GPP, IETF, IEEE etc. for the Networking Industry
- As these industries start to adopt Open Source Software that are originally developed elsewhere, significant challenges emerge as how to use general purpose Open Source while maintaining benefits from standardization.
- I have been involved in starting a Compliance Verification program since 2015 within, now known as, LF Networking and here are some tentative lessons that, I hope, could be applicable to other industries as well.
  - E.g. Financial Services, Automotive, Industrial IoT / Edge
- It’s an ongoing journey...
Applying Open Source to Industries

- The telco industry started embracing virtualization first with NFV through ETSI NFV ISG (Industry Specification Group) (2012)

- Network operator driven
- Try not to be a standard body
- Try to limit to a two year term
- Multifaceted and ambitious goals - “transform”
- The term “open source” was never used, but “software” was mentioned 32 times, “virtualisation”, 86 times, and “Openstack” and “OpenFlow” were cited.
Applying Open Source to Industries

- Two years later, OPNFV, an LF open source community was formed by many of the same network operators, participants of ETSI NFV, and broader technology providers (telecom/networking, IT, software) (2014)
The Beginning of a Compliance Verification Program

• March 2015: I proposed the initial idea to the Board of OPNFV. Formed a Board committee to drive the initiative.
• August 2015: Proposed community led bi-annual PlugFest events to bring developers together and facilitate integration and testing. First PlugFest was hosted in May 2016. Six PlugFests had taken place by now.
• September 2015: A technical project was created to develop/consolidate necessary test tools and test suites for use by the program.
• December 2016: The Board approved the program scope and governance.
• October 2017: The community completed its first test spec “2018.01”.
• January 2018: The OPNFV Verified Program (OVP) was launched with the first wave of products. Second version followed in September 2018.
• January 2018: Program expanded to ONAP following the formation of LF Networking umbrella.
• Today: ONAP is getting ready to launch its Compliance Verification to VNFs (Virtual Network Functions), probably soon this year (Check out ONS next month in San Jose).
What We Have Created

• An open source framework for verification testing
• A set of upstream test projects that develop quality test cases
• A community procedure to approve the official test suites, supervised by the TSC and approved by the Governing Board
• Software tools for automating all verification steps
• A volunteer committee to review all test results
• A branded mark and a public web site for listing certifications

[Diagram of verified products]

https://verified.opnfv.org/#/
What can we learn from this process?
Lesson #1: Have Clarity on Value Proposition

- Surprisingly more difficult than we expected
- Standard and open source communities formulate the “values” in different ways
- “Disruptive”, “Transformation”, means the ground rules are unclear
- Can we incrementally bootstrap?

- Help build the market for
  - LFN-based SDN/NFV/automation infrastructure
  - Applications or other devices designed to interface with that infrastructure
- Reduce adoption risks for end-users
- Decrease testing costs by verifying hardware and software platform interfaces and components
- Enhance interoperability
Lesson #2: Adapt to Collaborate in Open Source

• Instead of Specification, code and API
  • To accommodate, we “documented” code back to loose “description”
  • Avoiding code makes one “illiterate”
• Instead of RFP, Jira, issue, or better yet, pull request
  • Put requirements in a format that reaches developers, make them actionable
  • Scratch your own itch (e.g. operators can contribute lab qualification tests)
• Upstream, downstream, sidestream, or just confusion?
• Don’t try to repeal laws of economics, engineering, mathematics…
Lesson #3: A Rising Tide Lifts All Boats

• Except those that sink…
• It’s better to have a low bar initially, than no expectation at all
  • The trusted open process is more important
  • The expectation of interoperability and other goals
  • The consensus of adhering a community standard
• Unless you are willing to wait for a 5-10 year long industry process
  • Waterfall model vs. Continuous delivery
  • Fully automated test suites allow any time testing with minimum friction
• Get on board. Be part of the tide.
Lesson #4: Put the Horse before the Cart

• The Horse: the force unleashed by open source (free to innovate)
• The Cart: the goals an industry is trying to achieve via open source (e.g. standardization)
• OR,
• The Horse: open source, as I like to shape it to be
• The Cart: the goals (business goals) that I like to achieve

• Philosophizing: does the productive force dominate?
• Time to clarify the value proposition again
• This horse is a mighty force

https://tborash.wordpress.com/2012/10/14/always-put-the-cart-before-the-horse/
tborash: Learning to lead learning
Lesson #5: Verify Behavior. Code May Need A Different Mechanism.

• Verify behavior, because our “value proposition”, e.g. interoperability, is behavior based.
• We can “mandate” but it’s hard to “Verify” code.
  • Mandate without verification seems to mean very little
• Code needs a different mechanism. Overloading may not be helpful.
  • Commercial products have inherent incentive not to fork to avoid accruing debt
  • A community is healthier to always be reminded not to give others reason to fork
• Behavior can be tested to “cloud services” as well.
Lesson #6: Verify APIs and Use Cases

• Simple API verification leaves too big a gap
• Common use cases and design patterns are needed to meet the objectives. Examples,
  • High availability
  • Resource allocation, optimization, migration
  • Stress load
  • Security best practice
  • Performance characterization (predictability)
  • Networking diversity
  • Operational procedures
• This is where industry specialization can be adopted
Lesson #7: “Free as in Freedom” Is Not Enough

- Freedom to propose, code, test any part of the system behavior characteristics
- Having a freedom and effectively exercising the freedom are different things
- Who has a stake in it?
- Is it a present itch?
- Deployments?
- Tragedy of the commons.
- Skin in the game.
Lesson #8: Keep Calm and Carry On

- There will be many contentious moments and low points along the way.
- It could take a long time.
- It’s an ongoing journey...
“If you want to go quickly, go alone. If you want to go far, go together.”

*African Proverb*