We chose Robot framework for end to end tests but have used Cucumber-JVM for the separate projects. Here are my thoughts after using both in different contexts:

1. Robot Framework seems like it is catered towards testers who have no programming experience. At face value, there are many good libraries (SSHLibrary, Selenium2Library, Database, etc.) that can be utilized to run integration tests without knowing much about programming, as I'm sure you've seen in the examples.

2. Robot Framework outputs decent report documentation, though chasing down failures can sometimes be tricky. Its suite and test level setup and teardown methods are fairly straight-forward.

3. As a developer (and I imagine any QA engineer with decent programming skills), I find writing complex tests to be a chore in Robot. I'm not a fan of its text files, find IDE support to be lacking (w/ the exception of the RIDE framework, which crashed for me enough to abandon it - perhaps it has improved). "For" loops and conditionals are awkward, and finally writing keyword libraries in Robot as text files (and not Python/Java) makes them difficult to use with Python and Java keyword libraries w/o knowing a lot about the framework. For these reasons, I have often questioned the value of Robot Framework (which ends up being a pseudo programming language as soon as any degree of complexity is introduced) over something like nose (<https://nose.readthedocs.org/en/latest/>), where I can write simple Python, which will also generate decent reports and allows layered test setups. For this reason, we started bringing most or all of our keywords into Python code exclusively, but then lost the capability to write Gherkin-like Given/When/Then tests (<http://blog.codecentric.de/en/2009/11/givenwhenthen-and-example-tables-using-the-robot-framework/>) w/o using .txt files. In Cucumber, this is just an annotation over a Python function. Also, Robot's support for the Gherkin-like syntax is extremely limited; I often hit brick walls. Cucumber's use of regular expressions to translate between a natural language like English and any programming language is key.

4. Cucumber is intended for Acceptance testing only. While the line between high level business acceptance testing and lower level integration testing can be blurred, figuring out how to express lower level integration tests as Cucumber tests is challenging and time-consuming (so sometimes worthwhile, for the same reasons T.D.D. helps developers, by forcing you to think through what you're actually building). Cucumber shines at allowing feature discovery through communication and automated testing of those discovered features into the future. Your tests become documentation at the highest business value. I would argue that it is not the right choice for highly data-driven/bounds testing, as the more you test at this level, the less readable and interesting its resulting documentation will be for your stakeholders and new engineers coming on to a project.

I am working on our automation 2.0 vision at our company, and am hoping to write Python domain classes to encapsulate our business domain (and sub-domains, such as infrastructure), which we can then use in either Robot or nose for our integration tests, and Cucumber-JVM for our higher level acceptance tests.

Hope this helps. You may also find the <https://groups.google.com/forum/#!forum/robotframework-users> mailing list helpful for this same question. Any way you go, there will be challenges - multi-system testing automation is not a simple problem to solve (though it is a fun one!)