Component criteria

> "Picking your team is 90 percent of the battle" — Stewart R. Levine

A **component** is a Free / Libre / Open Source software (FLOSS) project which is selected to be part of WikiSuite. Please see: Software Components. While it's subjective to pick a component, here are some general guidelines that are thought to be a good fit. If you have some ideas, please do share them.

The premiere place to evaluate and compare FLOSS projects is Black Duck Open Hub (formerly known as Ohloh.net). Other great sources include

- AlternativeTo
- Social Source Commons

And for libraries to be included in apps (in our case, mostly PHP / JavaScript)

- https://packagist.org/explore/popular
- https://asset-packagist.org/

Fits the use case

As described at use cases

Multi-functional

As we want to reduce the number (complexity) of interactions between components, it's best that they are handled within each app. So "suite mentality" is preferred. Having all the desired sub-functionality which complements the other elements of the WikiSuite is of course important. However, this is a long term project and if a feature is missing, we can just add later on as we complete the puzzle.

Integration / All-in-one

Let's avoid fragmentation of efforts.

The Tiki Model has permitted it to become the Free / Libre / Open Source Web Application with the most built-in features

- Please see: http://pluginproblems.com/

Free / Libre / Open Source software (FLOSS)

This should be obvious but just to be sure! Preferred license is LGPL v2 because that is what Tiki uses.
Any OSI license is fine. Be wary of situations where the license could be closed up or the FLOSS / "community edition" abandoned, like MindTouch or Socialtext

Activity level

Number of commits and number of contributors reported by Open Hub. See Black Duck Open Hub.

Longevity

WikiSuite is a long-term project and many many organizations will depend on it. Thus, a project should ideally already have a history.

Leadership

Projects with good leadership and good community vibe. No one wants to be stuck in a non-fun place!

Community

Projects with larger/vibrant communities have more to gain from and more to contribute to WikiSuite. And the diversity of the community is a definite strength. Diversity leads to different ways to approach a challenge and improves the solution.

Roadmap

Where is the project headed? What is the plan/vision?

Release lifecycle

There should be frequent and / or scheduled releases. Ex: Tiki has a major release every 8 months, which has all kinds of benefits. Ideally, the project has **scheduled releases** so we can have a good cadence.

Ease of use

Ideally, people can use with little or no training and documentation is available. If the features are there but are not easy enough to use, this is something that we can improve together. As an example: Manticore Search not only has great documentation but also interactive courses to learn Manticore Search without any downloads or configuration.

Interoperability

There should be a commitment to being as interoperable as reasonably possible to projects outside the WikiSuite, by using (and enhancing if need be) open standards and protocols.

Deployment

Is it easy to install & upgrade?

- PHP lib: Packagist / Composer
- Mobile: F-Droid
- Desktop apps
  - Windows: http://portableapps.com/
  - Linux: the package manager of your distro

Multi-platform

Especially client side, we have to assume organizations will have eclectic hardware, including mobile devices. WikiSuite is (mostly) a server solution so users will be able to do the vast majority with a modern browser. However, there are some times where the browser is not (yet) ideal, and client software should be recommended. Ideally, this client software is

- Available in a portable mode. This means there is no "installation". The user can just use a USB key/portable disk.
- Cross platform (Win / Mac / Lin and ideally Android as well)
- Can start up and log in when user boots up computer.

Open development model

The project must be open to contributions. Project should be happy to add these new features ideally, in the core of the official next release and not a third-party add-on that needs to be maintained separately.

If a project has a non FLOSS component, it could be reticent to accept the inclusion of a feature which is an alternative. Please see: Open Core software debate.

Being backed or managed by a foundation/association is preferred. Vendor lock-in is to be avoided.

WikiSuite is resolutely "Upstream First", and has contributed massively to Tiki Wiki CMS Groupware and Openfire.

Chat / Forums / Mailing List / Bug Tracker / Source Repository / Wiki

Projects should have an online presence, where it's easy for new users to ask questions, get support, join the community, etc.

- Source code should be readily available via SVN, Git, etc.
  - We want all commits, not just snapshots.
- An XMPP (ideally) or IRC chat room or similar should exist, with regular presence of key people.
- Documentation and planning should be done with open processes, and tools like a bug tracker, mailing

list/forums and wiki.

Available as a SAAS

We expect that a good number of future users will enjoy the flexibility and avoidance vendor lock-in. Yet, they still want to hire someone else to host and maintain it. Encouraging SAAS is part of the project.

Availability of a services ecosystem

Similar to SAAS, future users will want to be able to hire WikiSuite expertise for support, training, customizations, feature development, etc. Encouraging this is an inherent part of the project.

Technology

Should be based on sufficiently popular technology so it has long-term viability. However, it doesn't have to be the most popular. PHP / MySQL / Smarty / jQuery / Zend Framework / Bootstrap is preferred because it's the technology Tiki uses (and pretty much all components need to interoperate with Tiki). However, it's clear that we'll need some other technologies that are better suited (ex.: to desktop apps).

Switching cost

The higher the switching cost, the more prudent we are about the selection.

P2P stuff like XMPP & Email is much more protocol & standards-based because they are inherently designed to communicate to another node. Inherent switching cost is lower. So if Thunderbird is eventually dropped, changing to another mail client is not too hard.

Anything with data or that shapes your processes can have a high switching costs. Tiki has a huge switching cost, not because it doesn't respect standards (it does), and not because it tried to lock you in (it doesn't), but simply because it manages so many different types of data and processes.

Design

As much as possible, let's encourage all components to move to Bootstrap to make it easier to have a common look and feel.

Code re-use and up-to-dateness

Does the project re-use code in a smart way? Are fixes upstreamed? Are they keeping up to date with the upstream project?

Please see
http://blog.versioneye.com/2014/02/18/why-your-software-project-will-slowly-die-without-continuous-updating/
http://blog.codinghorror.com/the-broken-window-theory/

Development practices

- The Twelve-Factor App: A methodology for building modern, scalable, maintainable software-as-a-service apps

Multilingual

Should have (or be ready to add) a process to collaborate on translations.

Willingness to participate to WikiSuite

The project should be willing to promote the WikiSuite to its community (ex.: with a page explaining the project and an announcement). Ideally, each project of the suite will Dogfood at least part of it. Realistically, migrating existing data is not easy, but for new needs/projects.... Project should be eager to increase interoperability with other components.

A Good FLOSS Citizen

Upstreaming fixes, discouraging license proliferation, etc.

Active in FLOSS events and associations

Such as

- FOSDEM
- OSCON
- FISL
- RMLL
- FSF
- OSI
- https://sfosc.org/SFOSC

etc.

The Open-By-Rule Benchmark

- http://webmink.com/essays/open-by-rule/
- https://webmink.com/essays/scorecard/

Also see the SWOT analysis for Tiki: https://tiki.org/SWOT.

QSOS

"QSOS is a method conceived to qualify, select and compare free and open source software in an objective, traceable and argued way"
http://www.qsos.org/?page_id=3

The Openness Index

- http://www.theopensourceway.org/wiki/The_Openness_Index

IndieWeb

- http://indiewebcamp.com/principles
- http://indiewebcamp.com/why
- http://dev.tiki.org/IndieWeb
- http://indiewebcamp.com/site-deaths (don't let this happen to you!)
- http://indiewebify.me/

Coolness

Some projects just have a little something special which makes you want to be part of them .

Non criteria

- Multitenancy is really awesome when you are offering SaaS, and some (most?) of the apps in the suite do offer it. However, one of our main objectives is to offer self-hosting, and thus, make it easy to move data and apps anywhere.
  - Kaltura offers multitenancy, and it makes sense to use it this way. However, if one project decides to move to his own servers, it's a pain to extract the data.

Related links

- https://dev.tiki.org/How-to-pick-a-software-library
- http://osswatch.jiscinvolve.org/wp/2013/05/08/4-tips-for-keeping-on-top-of-project-dependencies/
- http://oss-watch.ac.uk/resources/evaluatingsustainability
- http://oss-watch.ac.uk/apps/openness/
- http://sosopensource.com/395.html
- http://www.infoworld.com/article/2872094/open-source-software/seven-questions-to-ask-any-open-source-project.html
- « OFFRE LIBRE » permits to distinguish commercial offerings that respect the letter and the spirit of Free / Libre / Open Source software. (Covering France only for now)

- https://jgbarah.gitbooks.io/evaluating-foss-projects/content/introduction.html
- https://github.com/linuxfoundation/cii-best-practices-badge/blob/master/README.md
- https://12factor.net/
- https://sourceforge.net/blog/common-causes-of-open-source-project-failure-and-how-to-avoid-them/
- https://www.linuxfoundation.org/publications/2019/06/determining-true-openness-of-os-projects/
- https://github.com/massonpj/OpenMaturityModel/blob/master/README.md