Model

Tiki has a development model which is different than that of most of the other comparable projects. In a nutshell: Tiki is **software made the wiki way** and the FLOSS Web Application with the most built-in features.

The "Tiki model" consists of:

- Open Wiki community (do-ocracy) (sort of like Wikipedia but for software instead of content)
- **Wiki Way** participation in the code (500+ with full write access to the complete code base cumulatively, as of 2012 — now over 600)
- Scheduled releases (major releases every eight months, with Long Term Support versions, similar to Ubuntu)
- All-in-one codebase (1,000,000 lines of code, with everything bundled. Each feature is optional.)
  - Inherent synchronized releases (all features have to be ready at the same time).
  - Half the code comes from other projects such as Zend Framework, jQuery, Smarty, etc.
- Many features, but no duplication (in a wiki, similar/related content is merged, so the same is applied to Tiki's features).
  - Tiki is the Free/Open Source Web Application with the most built-in features.
- Dogfood - Tiki is a community recursively developing a community management system.

Table of contents

Video

Marc Laporte, a long-time Tiki Administrator and Tiki Community Software Association board member is interviewed for the 146th FLOSS Weekly netcast, with Randal Schwartz and Simon Phipps, current President of the Open Source Initiative (OSI).

Model

Wiki community

It's a Do-ocracy. Everyone is a volunteer. Things get done because someone **like you** decides to take the time to make it better. Join a team!

Participate in a TikiFest!

Software made the wiki way

Reflecting one of its core features, the wiki, a catchphrase for the project's development is "Software Made the Wiki Way". What this means is that the strengths of wiki-style collaboration for building knowledge bases — relatively free access to create and edit information in an essentially flat, bottleneck-free organization — can also be applied to the project's software development process itself. Virtually anyone can contribute code additions and changes to the Tiki project. While many Open Source communities have a send-a-patch-and-someone-else-will-commit approach*

, in Tiki, we encourage everyone to commit directly to the source code. Traditionally, all that has been necessary is to register at tiki.org ("join the Tiki community"), be registered at the project files repository site, and agree to abide by the "3 Rules" — respect the environment, commit early and often, and make it optional.

- 250+ people have contributed code to Tiki core (out of 500+ which have access)
- How to get commit access (as easy as it gets!)

There isn't any pre-commit evaluation procedure; developers are given access to commit and are encouraged to simply submit bug fixes or feature enhancements directly. Feature additions and other extensive changes can be done in an experimental branch and then merged into the main development branch. The more active members of the Tiki community informally monitor the commits of new participants. Code contributions such as new features are marked as "Experimental" in the Tiki site administration panel. As a feature gains traction as a useful one to continue, then it is maintained along with all of the Tiki code. Features that don't gain that traction or ones that are made redundant by other, better features and so on are given an "Endangered" label, and a path to migrate away from them is provided as they are removed from the code. In this manner, the wiki way of software development makes participation easy and protects the quality of the code over time.

This method reflects the general approach of Agile software development model; specifically, being a mature software project in which development is mainly feature addition and code improvement, Tiki has much in common with the Feature-Driven Development method. The 12 principles of Agile software development generally apply to Tiki software development process, although regarding "Face-to-face conversation is the best form of communication (co-location)," the Tiki project has always relied on the remote participation of its international community of developers, with only occasional in-person meetings (our "TikiFests"). The developers use the Tiki software itself to communicate, collaborate, track issues and so forth, so the software is tested and improved as it is used (sometimes referred to as "eating our

dogfood", as described below).

Scheduled releases

Scheduled releases are a fundamental part of the Tiki model. As a large community, it makes it clear for everyone what to expect.

There is a major release every eight months (adjusted from the previous two major releases per year), with several minor releases in between. Please see: Roadmap and Version lifecycle and When to release - think popcorn.

Whatever is not ready in time for one release is pushed back to the next. If you are late for a train, you may miss it, but you'll be early for the next one .

> *Why every eight months? Why not four, six, or twelve?*"

Six was too short (although this was the pace for several years). Twelve was too long.

Six to eight months was just about right. It's long enough to make big changes, yet it's short enough that you don't have to wait *forever*. For people in a business setting, it's essential for them to be able to a commit to a date with their customer. By having a set date, there is an incentive to commit to trunk, and know when it will become stable. People who need a slower cycle can use LTS.

> *How can we know there will be enough new features to justify a major release?*"

During an eight-month period, there are several thousand commits, which is more than enough to justify a major release. Just check Tiki2, Tiki3, Tiki4, Tiki5, Tiki6, Tiki7, Tiki8, Tiki9, Tiki10, Tiki11, Tiki12, Tiki13, Tiki14, Tiki15, Tiki16, Tiki17, Tiki18, Tiki19, Tiki20, Tiki21 and the RoadMap.

All-in-one codebase

All functionality is in the core. Developers are **collaborating** throughout the whole codebase (over 1 million lines of code) vs third-party module developers *cooperating*, or *coordinating* around a common "core." As far as we know, Tiki is the Open Source web application with the most built-in features. Read more about cooperation vs collaboration.

Inherent synchronized releases

Releases take more planning and coordination because all the features are released at once, but since it's de facto Synchronized Releases, the whole community can collaborate right away on the next release and this permits rapid major releases (even with such a vast code base). See lifecycle.

;Compare Tiki's all-in-one model, which makes it resistant to the typical issues faced by other applications, as explained by CMS Wire (Jan. 5, 2011):

> *Yes, Drupal 7 is officially available for download. But it's important to recognize that this is just the Drupal 7 core — the release status obviously does not apply to the thousands of modules (add-ons) contributed by the community. Although there has been much support from the community that supplies contributed modules, and an entire movement that pledged to have their modules D7 ready by release time, it hasn't exactly happened.*"

Lots of features, but no duplication

Tiki is a "one-stop shop" with tons of features. In fact, as mentioned, Tiki is the FLOSS web application

with the most built-in features. However, there is just one forum, one blog, etc. Having duplicate features would lead to all kinds of problems and would splinter our user and developer base. If several people want to improve a feature, they collaborate and extend, in the wiki spirit. This is why we can give specifically descriptive names like Tiki blog, Tiki forum, etc All features are optional.

On Wikipedia, when two pages are about the same topics, they are merged:
http://en.wikipedia.org/wiki/Wikipedia:Merging
In Tiki, it's the same idea but for features.

Dogfood

We eat our own Dogfood and we hope you will as well All *.tiki.org sites such as doc.tiki.org and dev.tiki.org are powered by **unmodified** versions of Tiki. And new versions of Tiki power those sites **before** the official versions are released, for more real-world testing.

FAQ

Decision-making & Governance

Decisions

Tiki is a do-ocracy. Join a team!

> *Ok, but who decides?*"

The vast majority of decisions go to the ones actually doing the work (be it code, translations, documentation, etc). There is a very strong collaboration culture in Tiki. So contributors discuss and find better ways to do things. You may have the impression that this is a big and complicated thing, but it's not. For the vast majority of things to decide, everyone agrees (it's easy to agree on the ideal; the real challenge is deciding on what you can do with limited resources, but since it's a do-ocracy, this solves itself).

> *No offense but sounds like a 'hippie mumbo jumbo'. If the shit hits the fan, who decides?*"

The project admins as listed on WhoWhat. For example, they act as a safeguard, as a last resort to deal with Difficult People.

> *Ok, but what if these people make a really bad decision?*"

Anyone can take the source code and start a fork.

> *Ok, ok, but what are the general priorities of the community?*"

Please see SWOT.

> *Is it backed by a big business?*"

While it's true that a number of Tiki developers run consultancy businesses around Tiki, decisions are made in the best interest of the community, not of one or many businesses. For example, a business wants

to find a way to lock you in. Some could be tempted to make hosting more tricky so they can promote their own hosting offer. We want Tiki to be as easy as possible to install, via Fantastico panels for example. We are not just some geeks who are running Tiki on dedicated servers as root. We do care about the little guy on a budget shared hosting account.

Is there an association?

Tiki is community-managed and open development is pursued by an official Tiki Software Community Association.

What is the business model?

Tiki as a project, has modest expenses, which are covered by donations and affiliate partner relationships. It doesn't need a "business model" like a company that has expenses or employees to pay. While consultants are available for hire, Tiki is "Community Open Source" and not "Commercial open source". If you would like to hire or offer Tiki services, please see business models.

Is it sustainable?

Absolutely! Various Tiki contributors can come and go, but Tiki, as a project and community will continue no matter what. No one can lock down the code and require people to pay for it. Tiki evolution and development are in no way, shape or fashion the result of VC funding.

Who contributes to Tiki?

Users, just like you!

Typically,

1. Consultants, which offer Tiki services
2. IT departments in organizations
3. Volunteers who are using Tiki for personal projects

License

Tiki is LGPL 2.1

> *Ok, but is it really free? I want to develop websites and I don't want my customers to know I am using Tiki.*"

Some systems have restrictions on removing promo link or logo. Some software is GPL but they ask money to remove the Copyright messages. With Tiki, you can do whatever you want. There is no commercial license. Since it's LGPL, you can include in commercial apps, sell, etc. Of course, we hope that you will want to contribute code, documentation, bug fixes, bug reports, feature requests, translations, or even money, etc. and/or hire some of the consultants, but you are not required to do so.

> *Will it always be free?*"

Yes, because Tiki is a community project.

Core and module developers

> *In CMS [add your CMS name here], there is the core and there are third-party modules. There are core developers and modules developers.*"

In Tiki, that distinction doesn't exist. Or if you wish, **all developers are core developers**. This is pretty much the opposite of most other systems.

Why don't you fix/improve the existing features instead of adding new ones?

This is a common concern/frustration of users. Especially when struggling with a bug.

Who would decide the priorities? Person A may think the priority is to improve feature X, but for person B, feature X is good enough, and adding Feature Y is more important. Of course, each person will tend to think their thing is the most important.

Tiki is a do-ocracy. Tiki is a diverse community. While Tiki is central to the livelihood of many people, they don't work *for* Tiki. They work for various entities that use, love, depend on, contribute to and want to see Tiki improve as a project. But each has his or her own priorities. This diversity is part of the strength of the project. The answer is that different people that work on different things. The WishList has that name because lists things that people wish for. These can be bug fixes, feature requests, etc. The line between a bug fix and a feature request is often muddled.

Even if we somehow did stop people from adding new features (imagine the discussions!), would they really re-allocate their energy to fixing bugs? Since they may need a specific feature for a project, so if they are not allowed to contribute, perhaps they would use another CMS/framework or they would do it anyway but not contribute. None of which is in Tiki's interest.

Drawing the line

Why don't you draw the line and say, "this is what we do and that's it"?

- Because of intertwingularity, it's very difficult to draw any lines. This is true for knowledge, it is also true for software functionality.
- And also, see answers previous question.

I reported a bug several months ago and it has not been fixed

Pretty much the same answer as "Why don't you fix/improve the existing features instead of adding new ones?" above.

Are features ever removed?

Yes, but with care and consensus. We want to keep upgrades easy.

The goal is to have a useful application. Typically, features are removed because of the evolution of browsers or the web makes them obsolete. Or it could be because another feature in Tiki becomes so powerful that it makes this one useless.

In a traditional small-core-and-tons-of-extensions model, code will die off on its own by a lack of interest. In the wiki way, the community must proactively do housecleaning. This is much like wiki pages, which should be re-factored, deleted or merged. Please see: Endangered Features.

Will features continue to be added forever?

Well, that depends on the people that participate in the community. So far, the number of features has been an asset, not a liability. Should this change, we'll adapt at that time. It doesn't matter how many features you offer, people still ask for more.

Of all the innovations, the rate of the totally new features has been slowing and what we see a lot is how various features are each improving and blending more tightly with other features, for example, Pretty Tracker.

You can shape the future of Tiki .

Continuity

> *How can I know Tiki will be there in 5 or 10 years? What about stability/long-term support?*"

Predicting the future is very hazardous. However, as the past can often be an indicator of the future:

- Tiki has continued to evolve since 2002. Please see Continuity.
- The source code is free software and will always be.
- All the data is stored in simple database tables. If need be, it can be exported to be re-imported in another application. Of course, this is not trivial, and migration tools from one app to another tend to be fragile. However, the data is there.
- There is a large install base.

So, Tiki should be around as long as the technology is around. If you know what will replace the web browser, please tell us .

So Tiki will be around. But how good will it be? Well, the rate of evolution, innovation, etc. depends on the participation of people just like **you** .

Thus, **think long-term**. Overall, we should be mindful about how our immediate actions fit in the long-term. Tiki will never be "finished" so any shortcuts now can come to bite us down the road (especially if someone else is stuck cleaning up the mess). See: Maintainability.

All-in-one model

Tiki has an all-in-one approach to features. Think of it as a suite of applications like Open Office. The features are built in and all optional instead of offered via third-party modules/extensions/plugins. Because of this model (and the open participation), Tiki has more built-in features than any other Open Source web app. (If you know of one, please let us know!)

Best-of-breed applications

> *I prefer a collection of best-of-breed applications that I assemble.*"

This can work only if you have the skills to tweak these applications to make them work together. And when each component is upgraded, you have to make sure the glueware is still working. And you need the programming skills to make this happen. If you don't complete the job, you will have a "Frankensite" patchwork of three to four apps, each with its own username/group and permission system. Each has its own template and menu system (if you care to unify the look and feel + navigation). Each will have its own search engine.

Lean apps / complexity

> *I like simple apps because I can easily tweak the code to my liking.*"

Tiki is a large application and there is a learning curve. However, it is easy to tweak. It was designed with simplicity and community development in mind. Please see the introductory documentation for developers and Coping with complexity.

Survival of the fittest

> *Most other CMS systems have several "modules" to choose from. Since there is competition among the "modules", it is the survival of the fittest.*"

That sounds like it could work and of course it can. However, this is Open Source software, not biology . . . so is it the most efficient? There are several challenges with this approach.

1. The site admin has to find, evaluate, test, install and maintain each one of these modules. If there are 3 wikis for Xoops for example, which one to choose? In Tiki, we believe if several people want to work on a specific feature, it is a lot more efficient, (and fun!) to work together. Please see this interesting Drupal note and this one, too.
2. The site admin must manage the versioning of the application vs the "modules". Some "modules" can be abandoned by their authors. So in the end, it is very important to choose your modules well. And again, choose the tool according to your needs.

> *Yes, but competition is good.*"

Competition is not the only path to innovation. Say you wanted to have the best possible definition for a word on Wikipedia and you have nine volunteers to help. What would you do? Would you create three teams of three and pick the best definition? Or would you have all nine people collaborate? If you split into three teams, chances are, when you evaluate what the three teams came up with, each team will have something you like. Yet, by picking one team and discarding the two others, you are wasting talent, motivation and energy. Now, it makes perfect sense to split the nine people in smaller teams according to **specific responsibilities towards a common goal**. If this model makes perfect sense to generate content for Wikipedia, why is it not a good model for software development? Imagine two communities of the same size. One with the competition model and the other, with the collaboration model. Which one will have the most code review? which one will have more code/functionality duplication?

Feature interaction problem

> *Features in the core generally tend to interact better together than external code. So how is this in Tiki?*"

Feature interaction is a huge challenge with all software. In Tiki, it's not perfect but it's quite good.

- Bugs and wishlist stats indicates less than 2% of reported bugs as "Bug: conflict of two features (each works well independently)"
  - The actual number is surely higher than 2% because not all people reporting bugs may have detected this as the cause of the bug. However, it's massively lower than the case in a core/third-party extension system.
- All the features are in the core so they share the same eyeballs
- There is very little feature duplication so this limits the Combinatorial explosion you have with hundreds or thousands of third-party extensions.

Lack of feature development

> *If there is only one of each feature, what happens when it doesn't do what I want/need?*"

Avoiding feature duplication doesn't prevent people from improving since participation is open. If there are several authors that want to improve a feature but have different needs/goals, they can just make them optional, and let end-users choose. It's free software. If you can code, just add the feature you need. If you can't code, try to find someone to help you.

> *Yes, but System Z has hundreds of modules. Here are extensions and third-party modules listings of popular PHP applications:*
>
> - *Drupal modules*
> - *Xoops modules*
> - *Typo3 extensions*
> - *Mediawiki extensions*
> - *WordPress plugins*
> - *Joomla Extensions*"

In the early days, search engines would boast about how many million web pages they indexed. And we had to navigate through pages and pages of results to find relevant information.

The important question is: "does the system do what you need now and for the foreseeable future". (This is just as true if you are picking Tiki vs another solution)

Tiki has over 1500 preferences/features/settings built-in, vs the sometimes 10,000+ external extensions/mods/plugins of some fine applications. But let's not forget that some of those external packages are features duplication.

The important thing is, do you see a feature that you want / need that is not offered by Tiki (please see feature list, modules list, mods list and all plugins).

Feature-creep

> *What about feature-creep? If you let just anyone add anything, it'll be chaos.*"

One person's "killer app" feature may be useless bloatware for another.

Tiki is a shared resource. We want it to be useful for various people, to solve various needs, in various contexts. If it's not solving your problem, chances are, an enhancement would be useful to other people. The trick is to try to use and extend the existing feature set. If there is nothing there that can help, it's to translate your need into something as generic as possible, so many problems can be solved with the same code, by changing labels or flipping a few switches. Of course, it's tricky to know this if you are new to the application, so you can work with the community to find solutions. A good example (of a generic feature) is trackers. Trackers are a database, forms and report builder, so you can create a bug trackers, a to-do list, a mini-CRM, etc, whatever you want. You can build unlimited custom applications specifically tailored to your needs.

The real challenge is to find a technical and organizational way to deal with the Law of Software Envelopment.

Tiki8 has filters in the admin panels which will lower the learning curve.

> *But too many features is confusing!*"

Yes, it can be. If you don't use a feature, don't activate it! And Tiki's Profiles (http://profiles.tiki.org) are there to help.

> *I just need a [add your feature name here]. This site is just a hobby, won't be big and professional like. . . .*"

Even if you think you just need a wiki or a forum now, isn't it nice that if you need a new feature later on, it will be there waiting for you? Just activate it, assign permissions and add links to your navigation.

Dead-end features / Dependency hell

> *I was using system X. My problem: I was using a third-party module. It is no longer supported. The developer has given up. I can't upgrade my site without breaking the module. Could this happen to me with Tiki?*"

This is called Dependency hell. The way Tiki is designed and organized, this has much less chance of happening. Yes, some Features can become unmaintained. However, they continue to work. If a developer enhances something in Tiki which breaks something else . . .

1. The developer has access to all the Tiki code so (s)he can test and fix anything that is risky.
2. We will know quickly via bug reports and IRC
3. The developer can fix whatever he broke because everything is in Git/SVN. The code is not scattered across dozens of websites. Tiki is very centralized, with all info on *.tiki.org sites and in Git/SVN. For better or for worse, backward compatibility (we call it "Respect the Environment") is a rule of Tiki.
4. While for Drupal, "Any feature that we add to Drupal core needs to be good enough to last us at least 3 years in a relatively unchanged state.", if you like biology, think of it as Tiki being more adaptable to changes **because** we don't have to worry about breaking hundreds of third party modules/plugins. We can undertake **massive** changes with each major version, like Workspaces in Tiki4.

How many times in the 10+ year's history of Tiki have users ended up with data in dead-end abandoned code? **This is very, very rare.** (This could have happened with daily snapshots of Git/SVN versions, but not when using stable versions.)

Reinventing the wheel

> *Tiki is like its own world, it is always reinventing the wheel.*"

While Tiki is self-sufficient, it offers many interactions with the outside world (services, technologies, standards, code re-use, data exchange, etc). Also, Tiki eats its own dog food.

> *Yes, but what about phpBB?*"

Sometimes, it's less work and more future-proof to develop your own features. Just because Tiki doesn't integrate phpBB like many other projects out there, it doesn't mean we are reinventing the wheel. Tiki forums are very powerful and support threading, wiki syntax, attachments, etc.

In reality, Tiki relies heavily on External Libraries, to have less code to maintain and to leverage other projects' experience. We try to contribute back (upstream) as much as possible. More than half of the code included in Tiki is from other FOSS projects.

Waste of disk space

> *Yes, but Tiki take a lot of disk space, especially for a PHP application.*"

Tiki 20 weighs 275 megs unzipped (93 megs zipped), but hard disk is really cheap these days. You can also use MultiTiki to use one set of files for several Tiki-powered sites/domains. Many people install via Control Panels and thus aren't even aware of the size of the package files and don't necessarily use, know or even have access to FTP.

Performance

> *Tiki is really slow, consumes a lot of RAM and CPU cycles.*"

In Tiki, we have a concept called WYSIWYCA. What You See is What You Can Access. The permissions system is very elaborate. Each user has a different experience, showing him/her what has changed since their last login (and only showing links for items which they are allowed to see). This takes more queries.

> *Yes, but even for anonymous users, Tiki uses way more resources (SQL queries, RAM, etc.) than other CMSs.*"

It would be interesting to have a real analysis to see if this is true. You would have to compare several apps with relatively similar content, features, etc. It's not that easy because apps are different. But let's say it's true Short answer for 98% of websites out there: **who cares**? Computers are getting faster and cheaper every month. People power is more expensive than machines. Yes, the code could and should be optimized in certain places. Yes, we should put some indexes on certain tables. However, we must always keep in mind that keeping the code simple makes it easier to understand, to customize and to fix. For the 2% of high-volume read-only sites: Please collaborate within the Tiki community to find ways to make Tiki faster. Ex.: http://varnish-cache.org/ Please also see performance for advice.

Security

Bottom line: Systems that are heavily reliant on plug-ins are more apt to have security vulnerabilities for which there is no known patch. Why is this? Even frequently-used plug-ins are not always actively maintained and enhanced by their original developers. Source

Another risk:
http://wpmu.org/why-you-should-never-search-for-free-wordpress-themes-in-google-or-anywhere-else/ Tiki doesn't permit PHP code in templates (by default but can be changed by the admin), which is a nice benefit of Smarty.

On the other hand, Tiki has more code. And more code is more potential issues. Our solution to this is simple but effective: Each file in Tiki has a feature-check, which essentially renders the file inactive. If a security vulnerability is disclosed, admins can turn off that feature.

Both models have risks. We feel it's a good trade-off.

API

> *Yes, but system Y has an API.*"

Every time people ask for an API, it's important to ask the question: "What are you trying to do?"

Tiki has no formal "API" but has several ways to connect/enhance/interact. It is designed to be easy to add functionality. Do you think over 300 people would have contributed via SVN if it was difficult? Hello

World.

Now, depending on what people are trying to do, they should look at Webservices, Plugins, Mods, Trackers, Interaction, External Authentication, etc.

Extensibility

- See Extension points

Examples of success with plugin model

> *Yes, but Firefox uses the plugin model and it's very successful.*"

Please read:
http://www.howtogeek.com/179213/why-browser-plug-ins-are-going-away-and-whats-replacing-them/

There is no question that the plugin model can and does work. This is to illustrate and explain how/why the all-in-one model works for Tiki.

1. We have to be careful about comparing a web application to a desktop app.
    - File permissions is tricky for web apps.
2. Firefox has a nice installer to upgrade the extensions, which makes it very easy for the end user.
3. If you have many extensions, you will have noticed that they are not always compatible with the latest release.
4. Do Firefox plugins have any kind of interaction with other plugins? If this is what you have in mind, please see mods and plugins).
5. Add-ons behaving badly: the challenges of policing the Firefox ecosystem
6. Open Source Browsers Change Their Tunes on Extensions

Experimental

> *But what if a feature is not good (yet)?*"

During the development phase, developers can use Experimental branches.

Once a feature works (even if basic), it can be added to the main code base and tagged as Experimental.

Is the Tiki model better?

It is very risky to say something is better. The answer is **it depends**.

The Tiki model works for Tiki. And the community will adapt as needed. The Tiki model has benefits and drawbacks, therefore, the Tiki project has different challenges to tackle than other comparable projects.

In the small-core-and-thousands-of-extensions model, an important core competency is the package management and the big challenge is dependency hell, which has prompted calls for Synchronized Releases.

Tiki has always, by its all-in-one model, had synchronized releases. However, this brings another set of challenges. (There is no free lunch!)

Benefits

- Tighter integration
- Less feature overlap/duplication
- Less code duplication and more code review

- Higher collaboration
- Lower total cost of ownership for users (easy upgrades, less chance of being in a dead-end feature)
- Users and developers can count on the fact that code or features are present.
  - This reduces code duplication because developers can call that code (instead of duplicating code to avoid adding a dependency).
  - Something like http://profiles.tiki.org which helps to configure your site is possible. Otherwise, it would involve downloading additional code and it would be complicated.

Drawbacks

- Huge codebase (Over 1 million lines of code)
  - Most users only use a small (but different!) part of it.
- Huge admin panels with over 1500 settings/preferences/options
- Big concern for backward-compatibility, which can slow progress
- Inconsistency in codebase
- Learning curve
- Dispersed focus
- Code_rot

Some things are good in Tiki and can be attributed to the model, but that doesn't mean other projects don't or can't have them.

For example, Tiki has one full documentation set for the whole application (over 1000 pages). It would be more difficult, but not impossible, to make a similar resource with the small core/thousands of extensions model.

Open participation

If you let everyone contribute, it'll be chaos. How can it possibly work?

That is what some said about http://wikipedia.org/

Conflicts

> *If I add my features to the core, then, any of hundreds of people can break my code!*"

Why would they possibly want to do that?

> *If you let "just anyone" participate, won't you get in trouble with malicious or incompetent people?*"

Answer: eyeballs.

Documentation

> *The documentation is missing, incorrect, out of date or just plain bad.*"

The quality of the documentation is variable. Some parts excellent, some are bad. There can be a correlation with the quality or activity level of the feature itself.

The good news is that it's a wiki, so please log in and fix it. If you know it's wrong but don't know what is right, ask in the Mailing Lists, forums, irc.tiki.org or wikisuite.chat.

> *The project should disallow code submissions without documentation updates.*"

This is certainly a possible model and likely used by some other projects. However, this is not the case in the Tiki community.

1. Some developers are fluent in PHP/MySQL but are not good in written English.
2. Some developers are not interested or good at writing documentation.

If you force developers to make unit tests, add documentation, or whatever, you will end up with fewer contributions. There is a valid debate to say "I prefer fewer contributions of higher quality". We are in the "commit early, commit often" mood and "things will improve with each version".

Some could argue that it's preferred to focus developers on adding/fixing features and it's up to the documentation team to handle documentation.

Whose job would it be to refuse (roll back!) a wanted feature because the developer didn't document? Can you imagine the debates there would be?

What often happens: Developers put up basic documentation, enough so that end-users get an idea of what the code does, and then these people can write proper documentation (hopefully with screenshots) from an end-users' perspective. Power users can also "interview" developers and capture this information for the documentation.

Code quality

> *The code sucks and is inconsistent.*"

Tiki is the FLOSS web application with the most built-in features, is 17 years old and 250 people have contributed to the code base. It's not surprising that code quality and style varies.

Take any open source application and ask around if the code is good, is modular, scalable, etc. Someone will always find something to complain about. A "critique" is very easy. I am sure if some of your work is available to public scrutiny, you will know what I mean . The question should be: Does it work or not? Does it solve my needs for now and the foreseeable future? Can I improve it? Most people use Tiki without changing any code. So code structure/philosophy/organization is of no importance to them. They just want something that works and something that they can upgrade easily over time.

Also, Tiki was started with the technologies available in: PHP 4, MySQL 3, Smarty 2, and no jQuery!

The good news is that various parts of Tiki undergo massive improvements. For example:

- Change to new profiles (Tiki3)
- Moving to Zend Framework 1.10 (Tiki3)
- Themes (Tiki3)
- Moving to PDO (Tiki4)
- Moving to jQuery (Experimental in Tiki3, official in Tiki4)
- Permission Cleanup to permit better category permissions (Tiki4)
- Modules (Tiki4)
- Toolbars refactor (Tiki4)
- Preferences in admin panels moved from static to dynamic, to permit searchable admin panels (Tiki5)
- Slideshows moved from native code to S5 (Tiki5)
- Logs revamp (Tiki5)
- Javascript refactor (Tiki5)
- UTF-8 support revamp (Tiki5)

- WYSIWYG from FCKEditor2 to CKEditor3 (Tiki6)
- Spreadsheet from native JavaScript front-end to jQuery.sheet (experimental in Tiki5 and default in Tiki6)
- Blog revamp (Tiki6)
- HTML5 (Drop support for IE6) (Tiki7)
- Mobile from HAWHAW to jQuery Mobile (Tiki7)
- Theme revamp for theme generator (Tiki7)
- Modules revamp for drag and drop (Tiki7)
- Search with Zend_Search_Lucene (Tiki7)
- Use PHP autoload (Tiki7)
- Parser revamp for plugins (Tiki7)
- Zend Framework 1.11 (Tiki7)
- Trackers revamp in Tiki7 and Tiki8
- Language handling (getstrings) (Tiki8)
- IPv6 support (Tiki8)
- Smarty 3.0 (Tiki8)
- Category refactor (Tiki8)
- Comments refactor (Tiki8)
- internal events (Tiki8)
- Smarty 3.1 (Tiki9)
- InnoDB support (experimental in Tiki8 and optional in Tiki9)
- Moved to Composer (Tiki11)
- URL Rewriting Revamp (Tiki11)
- Integration of Bootstrap CSS framework (Tiki13)
- etc.

The all-in-one model actually makes it easier to do massive changes from one version to the next.

> *The code is not modular enough.*"

Please see: Modularity.

> *I am a professional. Really, Tiki's design sucks. It should be more object-oriented.*"

Often, people with strong programming skills (or even not so strong) do not understand the unique challenges and dynamics of an Open Source project, especially a community-developed one like Tiki. Some will design something which is meant to be "superior", but it is too complex for people to use and participate in. Over 250 people have contributed to Tiki via CVS/SVN/Git. That is a lot of eyeballs! The code is simple and has **community coding** in mind. Please see the goals of Luis Argerich, Tiki founder.

Please also see: I know this is all wrong, but fear it might be right.

> *Tiki's design sucks. It's too big. It won't scale. It will implode. It's impossible to have so many features.*"

Tiki has been going on strong for 17+ years now. Tiki has more built-in features than any other open source web application (which is good or bad depending on your point of view).

Not only does it have a ton of features, it also has a very fast release cycle. Many powerful web apps have a release cycle of one and a half or two years (for the core; modules evolves faster), whereas it is every eight months for Tiki.

The Tiki model can't work

> *If it's not run like a professional project, it will fail.*"

There are many points of view over what is "professional" and "failure" vs "success". Some people could view that Wikipedia is not "professional" but it's hard to argue that it's a failure .

For many years, some people have expressed these concerns. There was a lot of FUD being spread around. Perhaps Tiki is like Wikipedia and is "is one of those things impossible in theory, but possible in practice.". However, here we are. Tiki is over 17 years old. It's a mature, very powerful application, with a large community of contributors, with tons of features and which is tons of websites & Intranets. Of course, there are a lot of things to improve and there will always be. Remember: if you don't need a feature, don't activate it. However, if you need it in the future, it's only a few clicks away. And remember also, Tiki is open source and open development. **You** can participate in its future.

A glance back - Tiki wins a Bossie!
http://www.infoworld.com/d/open-source/bossie-awards-2010-the-best-open-source-applications-150&current=9&last=10

Things you can pretty much only do in an all-in-one model

- https://profiles.tiki.org
  - You can make profiles because you know the features (code) is always there
- https://i18n.tiki.org
  - You can provide a site where all strings can be translated, in-context and web-based
- If you have duplicate extensions for something, you can't dogfood them all.
- etc.

Some important links

- A Process That Is Not, IEEE Software, November/December 2009 (vol. 26 no. 6), ISSN: 0740-7459 by Hakan Erdogmus
- http://sourceforge.net/blog/podcast-tikiwiki/
- How CMS architecture affects dev communities - A case study of Drupal, Tiki Wiki and XOOPS
- Hello World
- SWOT
- Teams
- http://rocococamp.info/WikiWay
- http://montrealtechwatch.com/2008/10/25/software-development-the-wiki-way/
- The Wiki Way
- The Cathedral and the Bazaar
- http://c2.com/doc/wikisym/WikiSym2006.pdf
- Books
- Wikipedia content guidelines and how they relate to Tiki software features
- Modular Application Architecture, presentation by Kore Nordmann and Tobias Schlitt
  - Tiki is somewhat a counter example to this presentation but it's important to think how Tiki has solved the problems in a different way.
- http://www.reddit.com/r/programming/comments/1a7gns/whenever_somebody_sends_you_a_pull_request_give/
- http://fedoraproject.org/wiki/Staying_close_to_upstream_projects#Why_push_changes_upstream.3F
- http://blog.versioneye.com/2014/02/18/why-your-software-project-will-slowly-die-without-continuous-u

pdating/
- http://blog.open-xchange.com/2014/04/15/what-makes-swiss-army-knives-so-successful/
- http://nic.wiki/about/wiki-values/ -> Assume Good Faith, Be a Builder, Change is Cheap, Radical Trust, Real People, Refactoring, Transparency, Work on What Interests You

Page content todo

- Move all related content to this page or at least link to it.
- Move some stuff out of Hello World
- address objections/questions
  - You can't be good at everything
    - Wikipedia has great knowledge in most domains because it leverages a community. The same can be done for code.

Alias

- philosophy
- development model
- organizational model
- Governance
- doocracy
- do-ocracy
- Tiki model
- The Tiki model
- Tiki way
- The Tiki Way
- all-in-one model
- all-in-one
- Wiki Way model
- The Wiki Way applied to software development
- The Wiki Way applied to software
- The Wiki Way for software
- The Wiki Way in software