Modularity

Some other systems like Drupal, Joomla! and WordPress have a small "core" and you add functionality through some of the thousands of modules/extensions/plugins/add-ons. On the other hand, in Tiki, pretty much all the functionality is offered in the "core". Please read about the Tiki model. There is a mods system for functionality outside the core but it is the exception (mods are for rarely used things).

A recurring concern about Tiki is the perceived lack of modularity. But should you be concerned?

There are various schools of thought about modularity. It is not a silver bullet.

> In later discussions Torvalds explained the reasons for its choice: a fully modular architecture, like the one adopted for HURD, would have posed problems to a degree of complexity that it could have compromised the accomplishment of the project. To avoid such risks and keep the degree of complexity of the project as low as possible, Torvalds decided to design a monolith and he actually wrote all the architectural specs himself, avoiding all the problems related to collective projects (e.g. division of labor, coordination, communication). On the other hand, the HURD micro–kernel, a project in direct competition with the Linux kernel, has paid for the choice of pursuing a fully modular approach from the beginning in terms of the continuous delays that have plagued its development. Nowadays, it is still under active development and still lacks the stability and performance assured by the Linux kernel.

Source: Modular Design and the Development of Complex Artifacts: Lessons from Free/Open Source Software (2005) by A. Narduzzo & A. Rossi

> As a result, the act of decomposing a large software project into components is an activity that results, at its best, in a suboptimal outcome: some sources of interdependencies are well determined and taken into account in the design of components and interfaces, while others are not. In some sense, even careful decomposition of large software projects tends to be accomplished making trade–offs between sources of interdependencies, recognizing the more visible ones and disregarding the less evident or less important ones. These reasons, combined with the huge size of large software project, account for the difficulties in the subsequent integration – testing – assembly phases. Likewise, less careful decomposition results in even greater problems at the final stages of code integration.

Source: Modular Design and the Development of Complex Artifacts: Lessons from Free/Open Source Software (2003) (which has more info than the 2005 paper: 39 pages vs 10 pages) by Alessandro Narduzzo and Alessandro Rossi

"There is, by and large, only one code base at Google. This has many advantages. Most obvious is that it is really easy to look at and contribute to code in other projects without having to talk to anyone, get special permissions or fill out forms in triplicate. That is just the tip of the iceberg, though. Having one codebase means that there is a very high degree of code sharing."

Source:
http://www.manageability.org/blog/stuff/google-coding-cultures

Gregor J. Rothfuss wrote:

> "*You can still enforce good interfaces but seeing which code you are about to break with a single grep is really useful*"

❝

> *Banon comes from a distributed systems background and says you want to run the compute next to the data. All the compute too; not search in one place, BI in another, machine learning in a third. You should do it in a single system.*

Source:
http://www.theregister.co.uk/2014/12/23/elasticsearch_big_data_search_tool_fancy_an_elk_hunt/?page=2

Modularity should not be a goal in itself.

The goals are:

- Code re-use to avoid duplication and provide easier maintenance and more functionality
- Better performance (ex.: only load what you need in memory)
- Consistency (to be able to change the look & feel or behavior throughout the application)
- Make it easier for devs to learn the system, and add/extend functionality without having to understand the whole system
- Make it easier for developers to work on various parts of the system without interfering with others
- Make it easier to test
- Security
- Maintainability
- Upgradeability
- Adaptability
- Coping with complexity
- Reduce disk space
- etc

Sometimes, too much modularity can cause that there is in fact more code duplication. When you have 3000 third party add-ons, there will be duplicate functionality because developers won't know everything that is available in the other 3000 add-ons. And even if they know, they may not want to re-use the code from the other module and have it as a dependency.

For a developer, modularity is great when what you need to do so happens to be planned that way. But what about if you want to change a behavior that was not planned that way? You may end having to change a lot of the core code anyway.

http://en.wikipedia.org/wiki/Intertwingularity

So, as with many things, while putting everything in black or white would be reassuring and simple, the reality is that there are many shades of grey and it's important find the delicate balance. The answer to the question is "it depends".

Is Tiki modular from	
a coding point of view?	Pretty much (depends where) and it evolves over time as code is refactored. Tiki has **community coding** in mind. Please read Hello World (Introduction)
a Tiki admin point of view	Yes. All the features can be turned on or off. And you can use and participate to http://profiles.tiki.org, which are collections of settings.
a packaging point of view	No. This is a choice. Many web apps will make you download a "core" which offers basic functionality. And then, you must download modules/extension/plugins/add-ons/languages separately. In Tiki, it's all in one nice package and you decide what you activate. More on this topic: The Tiki model

See also: Coping with Complexity