Distributed Decision-Making

The Mozilla project is far too big for any one person -- or even a small set of people-- to make the ongoing decisions regarding code appropriateness, quality or readiness to be checked into the CVS source repository. The project includes both a set of core technologies (layout engine, networking libraries, cross-platform component model, etc.) and a set of applications built with those technologies (browsers, mail/news readers, calendar, Internet Relay Chat client).   The code is large and complex; the number of daily decisions to be made is enormous. The project would slow to a crawl if a small set of people tried to make the majority of decisions regarding particular pieces of code.

Instead, decision-making is distributed to a range of participants through its "modules" and module ownership. A module is a set of files that implement a piece of functionality that has reasonably defined boundaries. A module may be the set of files in a directory, such as "accessible" for Accessibility. Or a module may be more conceptual, such as "Stylesheets". In such a case the module would include a number of files in different areas of the source tree. A module could contain just two files (if written in C or C++): a .h file and a .c or .cpp file.

Tiffney Mortensen 5/23/08 10:06 AM
Comment: Frank's comment: "A module may be the set of files in a directory, such as the directory "accessible" for the "Accessibility" module, or a module may be more conceptual, such as "Stylesheets"; in the latter case the module might include a number of files in different areas of the source tree."

Of course, "reasonably defined boundaries" doesn't provide absolute clarity. We use this definition because it reflects the code itself, where there may be overlap and ambiguity as to where a particular functionality ends and a different one begins. Trying to create absolute definitions would require elaborate rules and exceptions which aren't needed in most cases. Rather than spend energy devising such definitions, we prefer to let module owners manage any overlap or shared ownership if possible, with mediation when necessary.

Oversight of the Module Ownership System.

The module owners and the health of the system are tended by an identified group. That group is itself a module -- the "Module Ownership" module -- subject to the same process and policies as all modules.

Tiffney Mortensen 5/23/08 10:07 AM
Comment: Frank's Comment: I don't think it's that the "group is itself a module", it's the set of tasks and responsibilities that's a module. Also, I think "tended" is the wrong word, at least as applied to module owners. Finally, this introduces the idea of a "non code module" in passing without any explanation. I suggest rewriting as

"The work of the module owners and the health of the system are overseen by an identified group. Although it doesn't necessarily involve writing code, that task of module ownership oversight can itself be thought of as a module -- the "Module Ownership" module -- subject to the same processes and policies as all other modules.

The Module Owner Role

A "module owner" is the person to whom leadership of a module's development has been delegated. Historically the delegation was done by a group known as "mozilla.org staff," going forward will be done by the Owner and Peers of the Module Ownership module. Module ownership includes a range of responsibilities, such as: improving code quality, implementing revisions and innovations as appropriate, coordinating development with that of the rest of the codebase, developing and maintaining a shared understanding of where the module is headed, developing APIs where appropriate, documenting as much as possible, responding appropriately to code contributions, design suggestions and stated needs of the community; and creating an environment where competent newcomers are welcomed and included.

Tiffney Mortensen 5/23/08 10:09 AM
Comment: Frank: going forward that task will be done

Tiffney Mortensen 5/23/08 10:09 AM
Comment: Mitchell: This gives us a clearly understood framework for addressing questions about modules and module ownership.

A module owner's OK is required to check code into that module. In exchange, we expect that the module owner care about what goes in, respond to patches submitted by others, and be able to appreciate code developed by other people. Module owners have a fair amount of flexibility in how they do this. We do not have an elaborate set of rules or procedures for how module owners manage their modules. If it works and the community is generally happy, great. If it doesn't, let's fix it and learn.

Module Owners need not do all the work of managing the module themselves. Module owners may identify others who can also approve code for check-in into a module. These developers are known as "peers" and ought to possess many of the qualities of a good module owner. Module owners must designate to a peer the evaluation of their own code; module owners are not permitted to review their own code. If there is no module owner, the OK of a peer is sufficient to check code into that module.

Module owners are not tyrants. They are chartered to make decisions with input from the community and in the best interests of the community. Module owners are not required to write code because the community wants them to. (Like anyone else, the module owners may write code because they want to, because their employers want them to, because the community wants them to, or for some other reason.) Module owners do need to pay attention to patches submitted to that module. However "pay attention" does not mean agree to every patch. Some patches may not make sense for Mozilla; some may be poorly implemented. Module owners have the authority to decline a patch; this is a necessary part of the role. We ask the module owners to describe in the relevant bug their reasons for wanting changes to a patch, for declining it altogether, or for postponing review for some period. We don't ask or expect them to rewrite patches to make them acceptable. Similarly, module owners may need to delay review of a promising patch due to an upcoming deadline. For example, a patch may be of interest, but not for the next milestone. In such a case it may make sense for the module owner to postpone review of a patch until after matters needed for a milestone have been finalized. Again, we expect this to be described in the relevant bug. And of course, it shouldn't go on very often or for very long or escalation and review is likely.

Tiffney Mortensen 5/23/08 10:10 AM
Comment: Frank: Module owners are not required to make code changes or additions solely because the community wants them to do so.

Tiffney Mortensen 5/23/08 10:11 AM
Comment: Frank: agreeing

Escalation and Review

The owner and peers of the Module Ownership module will get involved if controversy develops and cannot be resolved otherwise. A module owner may ask for a public statement of agreement with a particular action. Sometimes other contributors suggest ways in which a module owner might improve. Sometimes there is ongoing controversy. We prefer that the community resolve these issues when possible, but acknowledge that this can't happen all the time. We try to avoid making absolute decisions like "this must happen" but will do so if required.

Criteria for Module Ownership

There are a number of elements which are important for good module ownership. First of course, is the person's expertise with the code in question. But over time we've learned that a set of additional criteria is also important, and that a great hacker can be a poor module owner. The criteria that go into the mix for a good module owner include:

1. expertise with the code in the module
2. current level of involvement with the module
3. understanding/vision of where the module ought be headed
4. appropriate understanding of Mozilla codebase as a whole and the module's relationship to it
5. ability to evaluate code for that module, including contributions of patches and new features
6. ability to evaluate impact of code on other parts of the codebase
7. ability to communicate with a diverse, geographically distributed community
8. willingness to evaluate contributions on their merits, regardless of their source (i.e., no 'not invented here' syndrome)
9. ability to consider varying perspectives and needs of different consumers of that module
10. ability to resolve different needs through factoring or other abstraction techniques when appropriate.

Tiffney Mortensen 5/23/08 10:11 AM
Comment: Frank: ought to

Designating a Module Owner

We prefer that an individual work with a module for some time and demonstrate his or her ability to fulfill most of the criteria most of the time (we're not naive enough to require perfection), and that a consensus form about designating this person as the module owner.

This way the designation is more of a confirmation than an appointment. We haven't always done this, and we haven't always done it well.

This means that there will be times when there is no module owner. Especially in cases of modules that have received little attention, have started to rot, and some brave soul steps up to figure things out and get us back on track. We'll shower these folks with thanks for tackling the job, and we can do this immediately. We may not immediately designate these people as module owners. Almost by definition, it will be difficult for this person to have demonstrated some of the criteria, particularly 1-5 until s/he has spent some time working with the module. It's possible that someone's expertise is so broad and so deep that s/he could do this, but we would expect this to be the exception rather than the rule.

Tiffney Mortensen 5/23/08 10:13 AM
Comment: Frank: This means that there will be times when there is no module owner. In particular, in some cases modules have received little attention, have started to rot, and some brave soul steps up to figure things out and get us back on track. We'll shower these folks with thanks for tackling the job, however we may not immediately designate them as module owners.

In determining a module owner, the criteria above are not necessarily accorded the same weight for each module. The importance of a particular element depends on the module. For example, criteria 4 (appropriate understanding of Mozilla codebase as a whole and the module's relationship to it) and 6 (ability to evaluate impact of code on other parts of the codebase) will be of less importance for modules that are self-contained, and of great importance for modules containing core technologies which affect other parts of the code significantly. Similarly, criteria 9 (ability to consider varying perspectives and needs of different consumers of that module) and 10 (ability to resolve different needs through factoring or other abstraction techniques when appropriate) will be less important to a module which serves a specific, clearly defined function for a small number of contributors, and critical to a module which supports a variety of uses and a broad contributor group.

Tiffney Mortensen 5/23/08 10:12 AM
Comment: Frank: the Mozilla codebase

Tracking Module Owner and Peer Data through Despot

Mozilla uses a database known as "Despot" (despot.mozilla.org) to track code modules, module owners and peers. The data can be viewed at www.mozilla.org/owners.html. This page also contains an indication of what code makes up each module. Modules, module owners and peers for modules relating to other, non-coding activities can be found at [**wiki.mozilla.org/XXXXXX**.]

Tiffney Mortensen 5/23/08 10:14 AM
Comment: Frank: Mozilla uses a access-controlled database known as "Despot" (despot.mozilla.org) to track code modules, module owners and peers. The data from that database, including an indication of what code makes up each module, can be publicly viewed at www.mozilla.org/owners.html.

Tiffney Mortensen 5/23/08 10:15 AM
Comment: Modules, module owners and peers for modules relating to other, non-coding activities, including the Module Ownership module itself, can be found at [wiki.mozilla.org/XXXXXX.]

Transfer of Module Ownership

Module ownership is transferred through the owner and peers of the Module Ownership module. A module owner should resign by sending mail to this group. The module owner should feel free to include recommendations for a new module owner. In some cases, the module will have a peer who has demonstrated the criteria for module ownership, is interested in being the module owner, and is generally accepted as being a logical successor. In such cases, the Module Ownership owner or peers will commit the change of ownership to Despot. In other cases there may not be anyone who has demonstrated the ownership criteria with respect to that module. In these cases, the module may go without an owner until an owner develops, with the peers of the module providing the review and OK necessary to check into that module.

Relationship to Bugzilla Component Owners

Occasionally there is some confusion between the role of module owner/peer and that of default owner of a Bugzilla component. The roles are quite different. A component owner is the person best suited to receive incoming bug reports for a particular component; not necessarily the person best suited to make decisions about the direction of the module and the review of its code. There are several reasons for this. First, Bugzilla components do not map exactly to modules. That's because components reflect the way bugs are perceived and experienced, not necessarily the structure of the code. Second, managing bugs is a different task than managing the code of a module. The skills required are different. Some great hackers are not so good at reviewing bug reports regularly, tracking progress,

Tiffney Mortensen 5/23/08 10:16 AM
Comment: Frank: component,

Tiffney Mortensen 5/23/08 10:17 AM
Comment: Frank: Second, managing bugs is a different task than managing the code of a module, and the skills required are different.

reassigning bugs to correct owners, making sure test cases exist, etc. Some contributors are excellent at these skills but not necessarily at directing code development. So in some cases the Bugzilla component owner and the module owner may be the same person. But in many cases they will be different.

Tiffney Mortensen 5/23/08 10:33 AM
Comment: Frank: So in some cases the Bugzilla component owner and the module owner may be the same person, but in many cases they will be different.

Poorly Maintained Modules

Periodically a module is not well maintained and no longer interacts well with the rest of the codebase. This can happen where there is no module owner, or when a designated module owner is too busy with other things to tend to the module. Conceivably it could happen when a module owner is active, but has an approach to a module that the community in general believes is inappropriate. We prefer that the development community identify such modules, propose a solution, and implement improvement. If this can't happen for some reason then the Module Ownership Peers will get involved to find the best possible resolution.