UserModel

The current user activity

How do people use gump right now?

Some people's activity is triggered from receiving a failure notification. On receiving it, they'll click-through to the website, spend some minutes looking at the pages (perhaps compare with the previous days' result), figure out the problem, remember the troubling project, open up their IDE, commit a change to some project, and wait for the next notification (there's no success notification).

Other people track gump actively. They take a look at the state of the tree or they'll run a local build or partial test build when they change something to make sure it doesn't infect the project its dependencies. But there's only a handful of people doing this, because it requires a lot of understand of gump internals.

Let someone else do it

For many gump-built projects, a small subset of the people working on it do the gump maintainance. A change is only needed every now and then anyway, someone is happy maintaing the thing, so people don't bother to learn. They know that a failure notification is a sign of trouble, but usually the problem goes away quickly enough that they're not too worried. If a problem pops up for a long time, they'll send an e-mail to the person working with gump and ask them what is wrong.

Update descriptor, one iteration per night

Those people doing that maintainance for that project usually have a hunch after reading the failure notification of what is wrong. Nine times out of ten, either the buildfile or the gump descriptor needs a simple change (someone introduced a new dependency and didn't update the gump descriptor, for example). They make the change (visit the "all projects" gump page and find the name for the new dependency, add it to the project definition using their favorite editor, and commit the new definition), then wait for the next run to find out if it was sufficient.

Use the commandline tools

These people have shell access to a gump machine or have a gump installation on their own machine. They're familiar with the [GumpCommandLineOptions command line], have a reasonable grasp of the gump internals (having read some of the docs), and usually they spend quite a bit of time helping "lesser gump meisters" debug their build issues by running builds locally from the commandline, often tweaking various parameters and setting things like -debug, then relaying the results via e-mail.

Keep track of the state of the gump tree

These people have an active interest in the quality of a large range of projects. They might be subscribed to the gump RSS feeds. They'll notice when things break and stay broken by looking at those feeds, and will send e-mail to the people who broke things asking them to fix them.

Develop gump itself (gump as gump's integration test)

The gump developers will usually implement a feature and push it live (via a cvs commit + cvs up) immediately, run a build, and if things don't seem to break, the feature is left in. They have working knowledge on the state of the gump tree and if it is invariant under the new feature, they're not too worried.

The current user model

The dominant user model seems to be that gump is some complex beast written in a language they don't understand, working in a way they don't exactly understand, maintained by some people to do its thing on some server somewhere. This beast is controlled through dozens of XML files which live in its CVS. It detects if something breaks, and if something breaks, you need to figure out which XML file to update. Sometimes, updating an xml file won't help and some gump master will send e-mail to a project you depend on because they broke backward compatibility.

In short, gump is something complex not easily understood, and there's people who help you when there's a problem. A bit like how granny uses a computer and calls her son or grandson everytime there's a "problem" (like when it says there's a virus definition that's wrong).

The desired user model

Something like this?

Gump emulates a fellow developer (though he's not office-assistant-style-patronizing). You can tell gump to take a look at your changes. He'll find some time for that, try and compile and test your stuff, then send you a bug report if he finds a problem. Gump is frantic about testing, so he'll usually run a lot of diagnostics while he works on the stuff you feed him, and he has an uncanny way of figuring out if your change breaks your friend's project.

There's no such thing as "friend of gump". Gump is a friend of everyone. He's a skilled, automated, intelligent, reproduceable beta (nay, alpha) tester.

In other words, gump is a unique indespensable tool (like CVS or unit tests or a bug database) that you integrate to maximum extent into your build experience, because it catches problems for you before they occur. Every time you type "ant" or "maven" or hit the play button in your IDE, your project build system interacts with a remote server to verify the changes you've just committed are backward and forward compatible with your own code and that of others. The integration of the tool into your environment is easy: you install a custom ant task, a maven plugin or an IDE plugin, and you configure those via the usual means (xml or properties config file kept in CVS or SVN with your own code).

That server provides a web interface with insight into some of its internals. It shows you your project's dependencies and dependees through the gump web application, including things like build statistics. The server can be configured to autopublish build results (jars, distributions, javadocs, clover reports). You can subscribe to various events that happen on the server (new projects being added, projects failing to build, your project failing to build), either using an RSS feed or by having the server send e-mail.