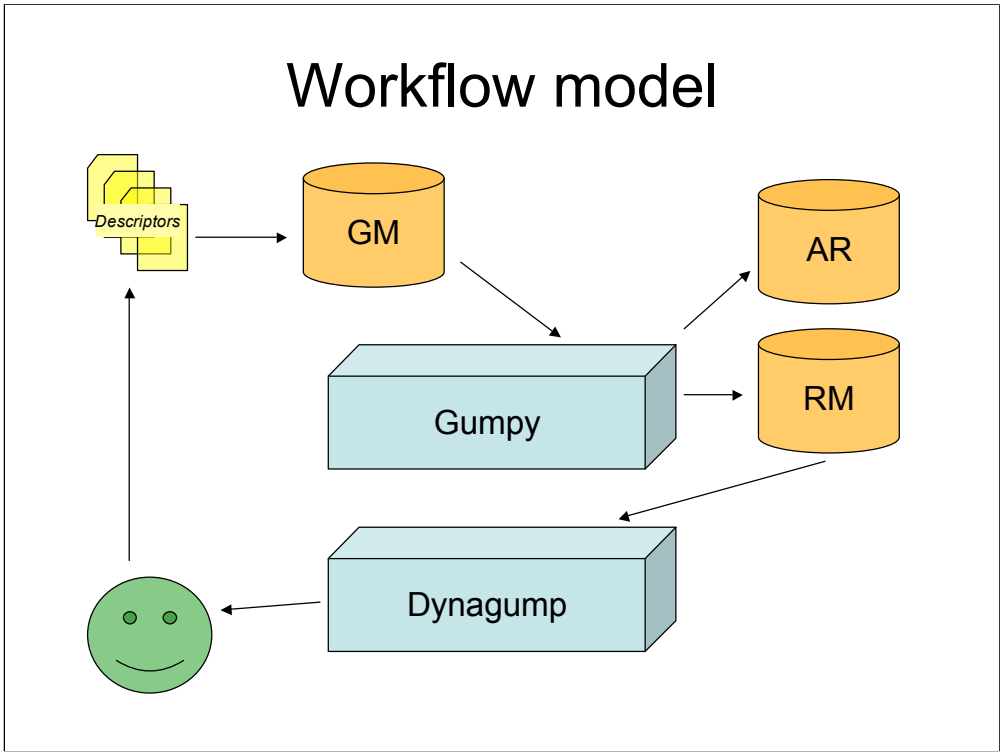


At a very high level, this is what gump does. We'll be looking at each of the items described here seperately.

•User edits project descriptor and commits

•Projects maintain their own descriptors (based on the GOM or mavens POM)

•The Gump Metadata (GM) is built from those GOMs and POMS (as a RDBMS, potentially)

•Gump checks out or updates stuff and runs build scripts

•Gump sends certain output into the artifact repository

•Gump sends build results and information into the Runtime Metadata (RDBMS with history information)

•Dynagump sends e-mails and the like based on that

•Dynagump generates user reports on demand

•XML files. Most live in Gump CVS (will live in Gump SVN).

•Similar to maven project.xml files

•Can split up into multiple files (across repositories) using hrefs

•<module/> for each CVS or SVN module

•<project/>, usually for each "unit" that produces something (ie a jar file), may also run tests or generate docs or …

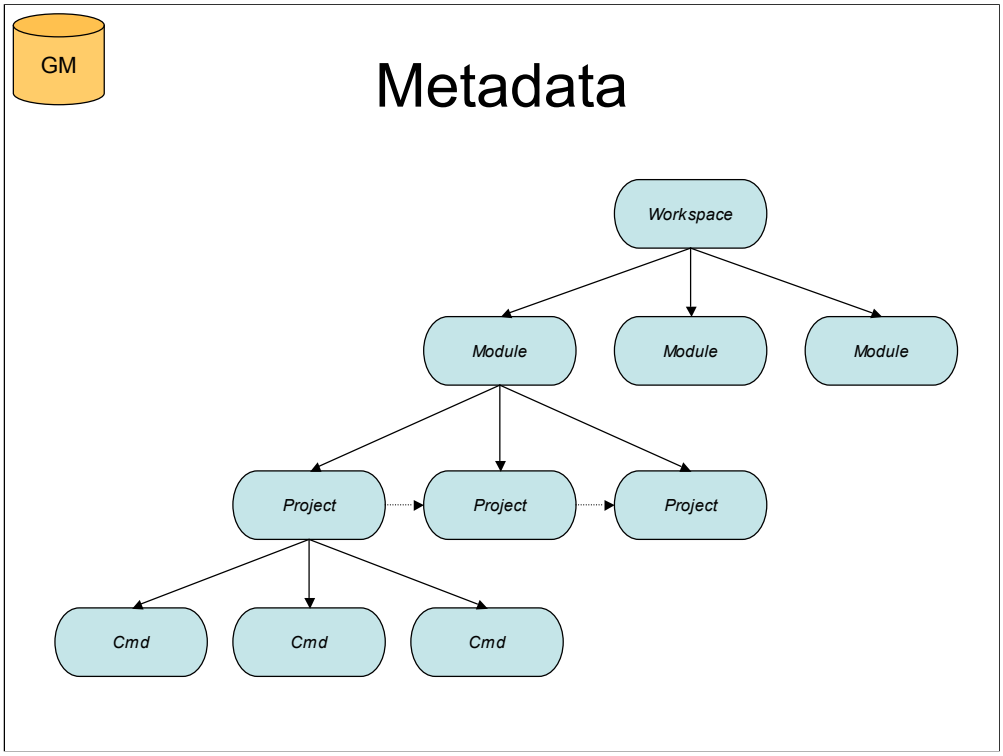
•Projects declare inputs (dependencies), outputs, and build commands (<ant/>, <maven/>, <mkdir/>…)

•See http://gump.apache.org/metadata/, actually a lot of docs on these

•Learn by example…look at setups for other projects…

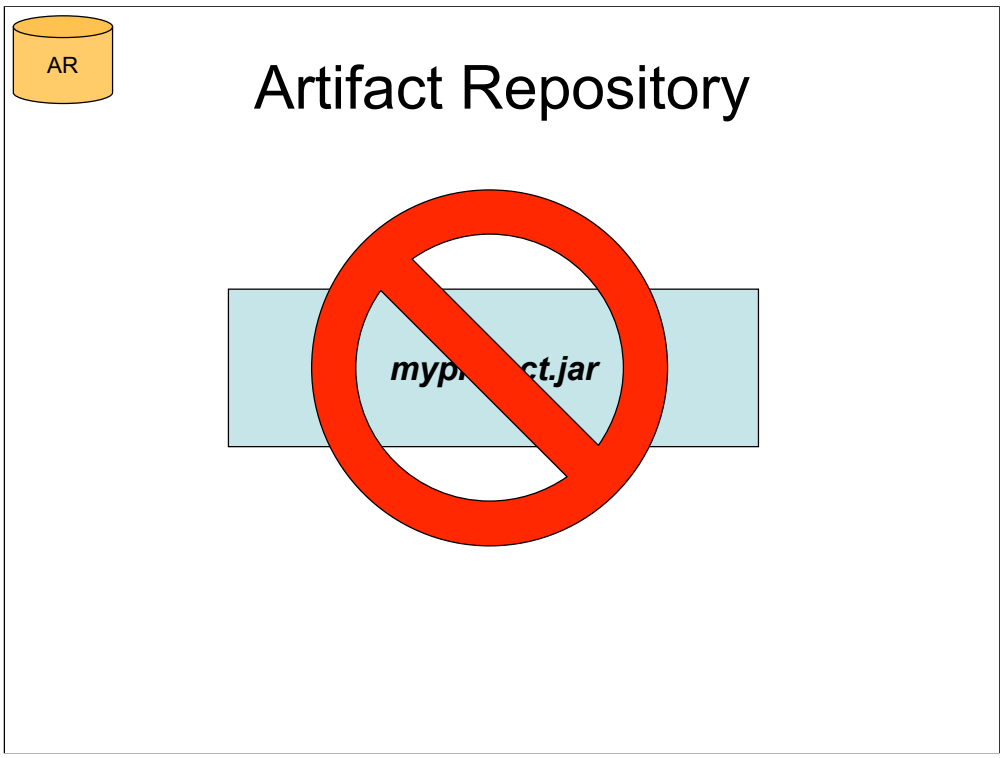
TODO:

•We want to make gump read maven project.xml files…

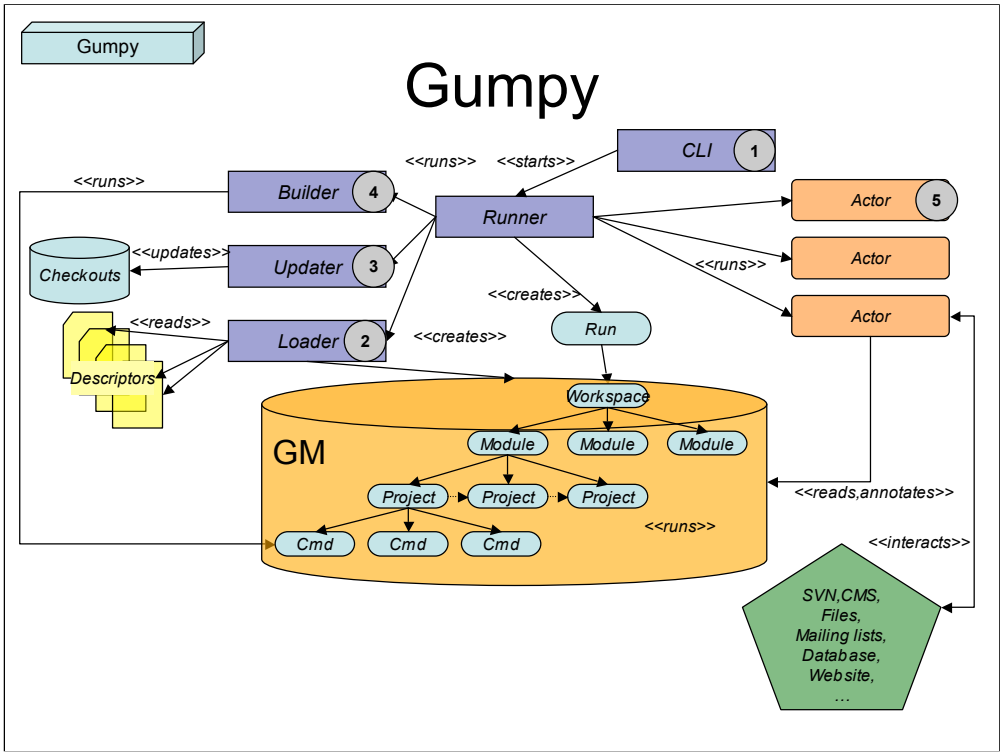


Gumpy builds an object graph in memory (gump.model package) which contains all the gump “intelligence” regarding the model, for example, it will fill out the model with defaults if some elements are omitted, and transform the way dependencies are specified into a standard way.

It is a bit of a shame that most of what goes on here is not documented. Rather, the assumptions made here can only be retrieved from seeing how the actual XML descriptors we use actually lead up to something. Fortunately for end users, you never need to see most of this.



Gumpy maintains a repository of all the stuff that it has built. But we consider it internal-use only: because Gump is inherently unsecure (we download arbitrary code from arbitrary CVS that can execute arbitrary commands), these files could contains bugs or even viruses!



Gumpy is our “buildmeister”, an object-oriented python engine which handles all different kinds of tasks, which we split up into bits called “Work”. These Work items are handled by either a core component (Runner, Loader, Updater or Builder) or by an Actor (documenter, notifier, and others).

The workflow is as follows:

•A command is started from a cronjob

•This command does a self-update, initialization, parses options, sets up logging, etc

•Now, a Runner is created, which contains most of the core build logic

•It delegates to the Loader to build a Gump Object Model in memory from the xml descriptors

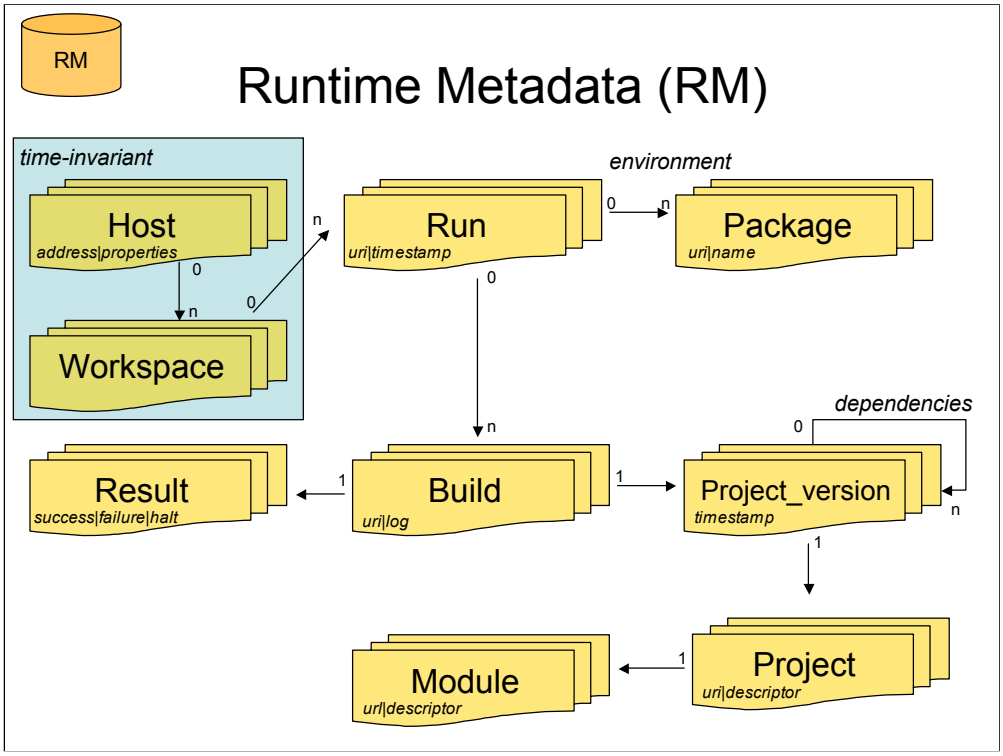
•Next it fires up the Update which checks out or updates materials from version control

•It is now time to start doing “real” things.

•The main task is building all the different projects, which the Builder handles.

•The runner also references several Actors, which can cause “side effects”, like sending out e-mail.

TODOs:



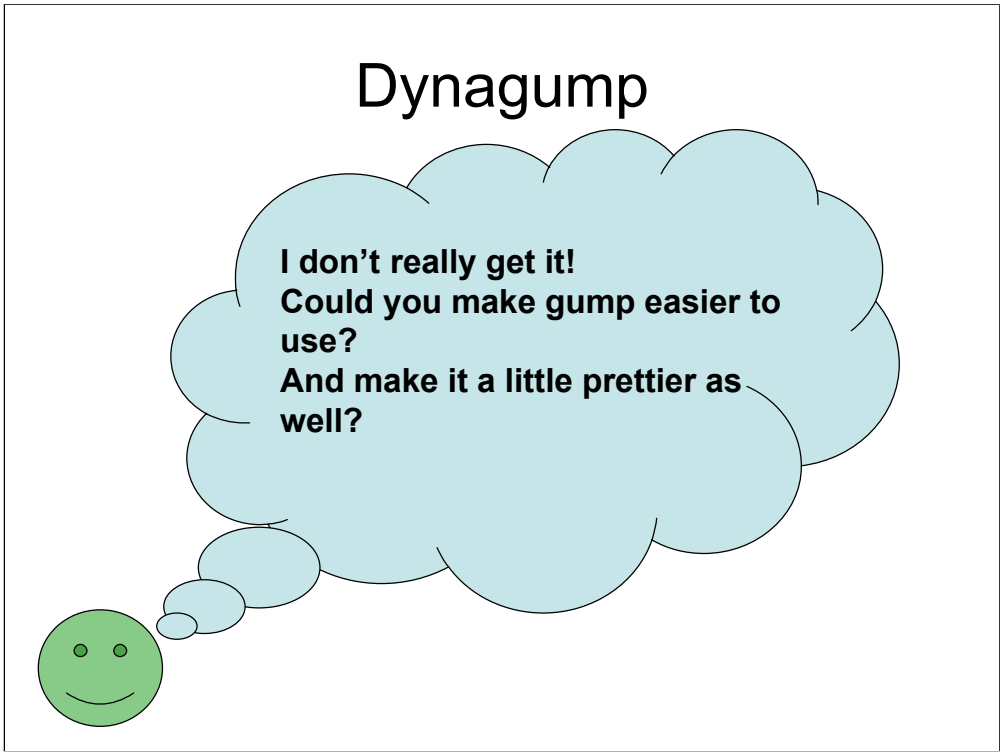
The RM database is the basis for most of gump's "intelligent" behaviour. By careful analysis of the data in this database, we can figure out who caused what to break when, which projects are long-term problematic, etc etc.

The RM database is different from what happens before it (ie the stuff that Gumpy does) in that it models time. Gumpy simply runs several times a day and publishes the result of each run, but it doesn't try and compare those runs to each other. Instead, Gumpy pushes data into the RM so that Dynagump can do that analysis.

•Hosts are identifiers for physical machines; besides an address (ie brutus.apache.org) they have certain info about it, like the # and type of cpu.

•Workspaces are gump runs configured with some particular set of parameters that might influence the way it builds (like the "live" build or the "kaffe" build on a machine). Decisions which change over time (by a gump admin changing something) are in the workspace rather than directly with the host.

•Runs are the complete sets of results for any particular gump run. They specify what installed packages (for example java version or kernel version) was built against, ie the entire environment, as well as all the builds that were run. Things which change over time (out of the gump admin's direct concious control) are part of a run rather



DynaGump is currently under development. We don't know exactly what it will look like (Shiny! Sweet! Cool!) and its featureset is still being thought about. The key thing we want to do here is make gump easier to use. Better reporting, friendlier e-mails, etc etc.

Watch this space!