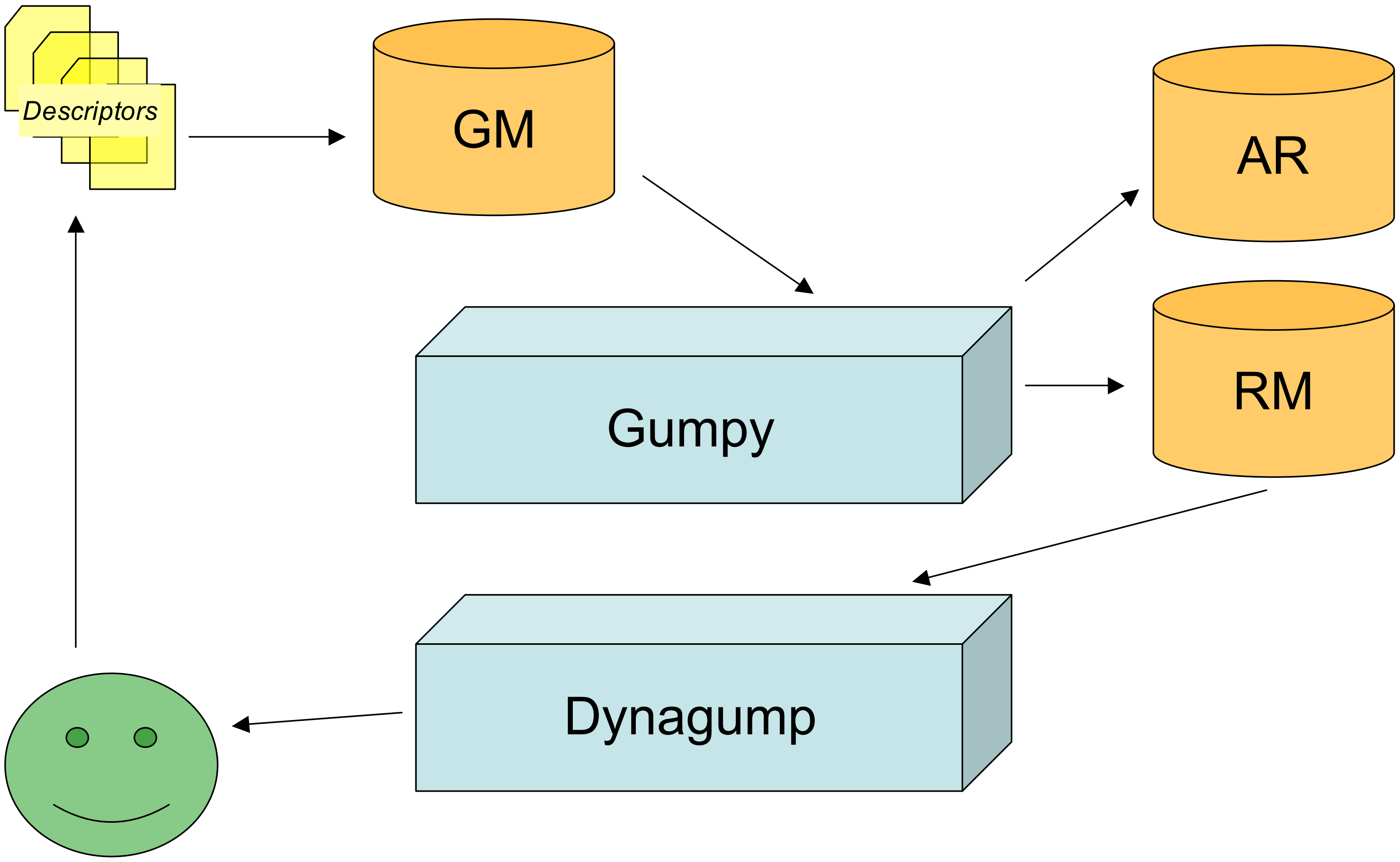
Workflow model



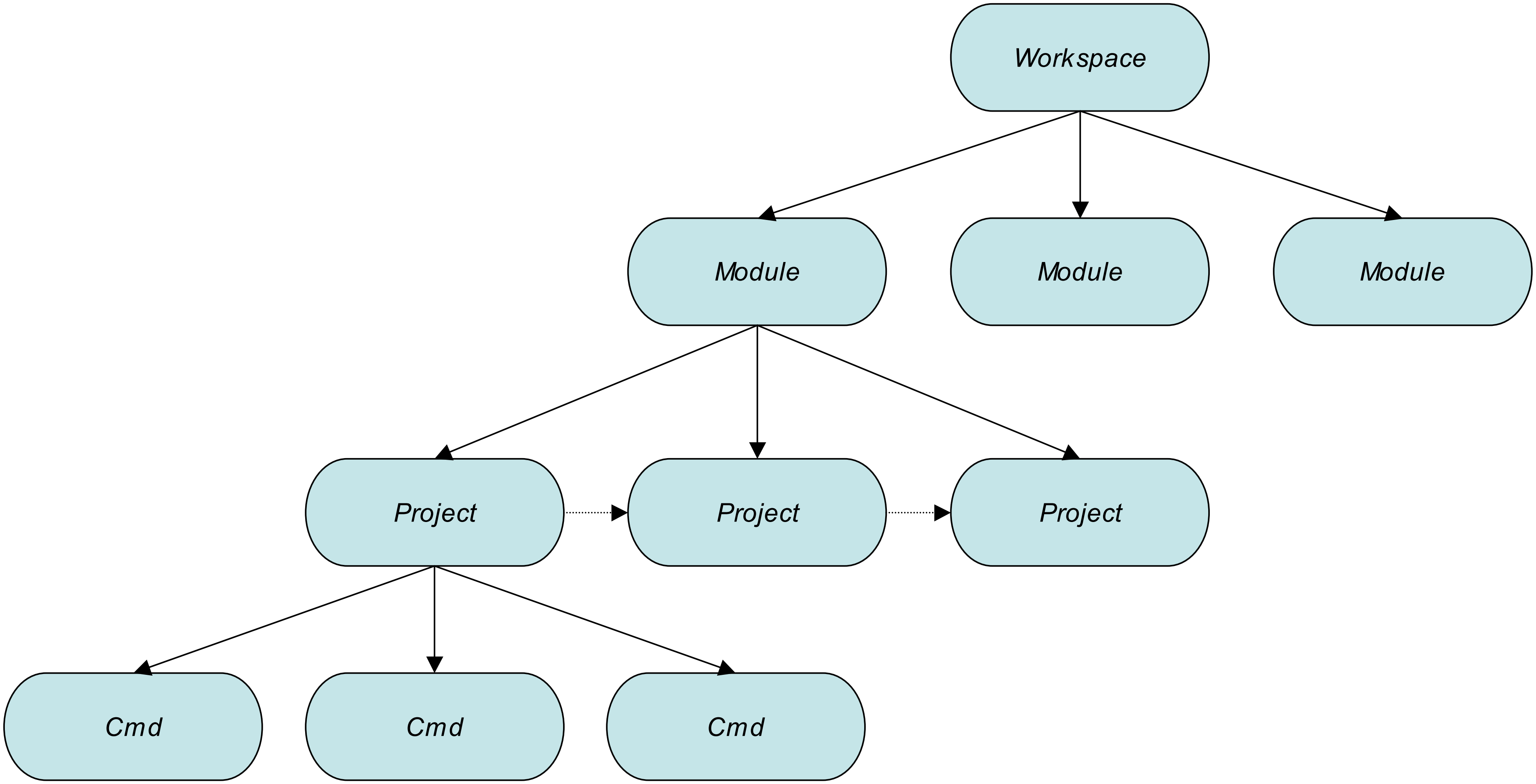
Descriptors

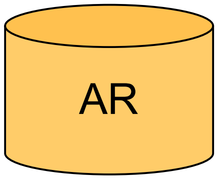
```
<workspace>
  <module>
      <svn …/>

      <project name=“myproject”>
            <ant target=“jar”/>
            <jar name=“build/myproject.jar”/>
            <depend name=“otherproject”/>
            …
      </project>
  </module>
  <module href=“project/other.xml”>
</workspace>
```

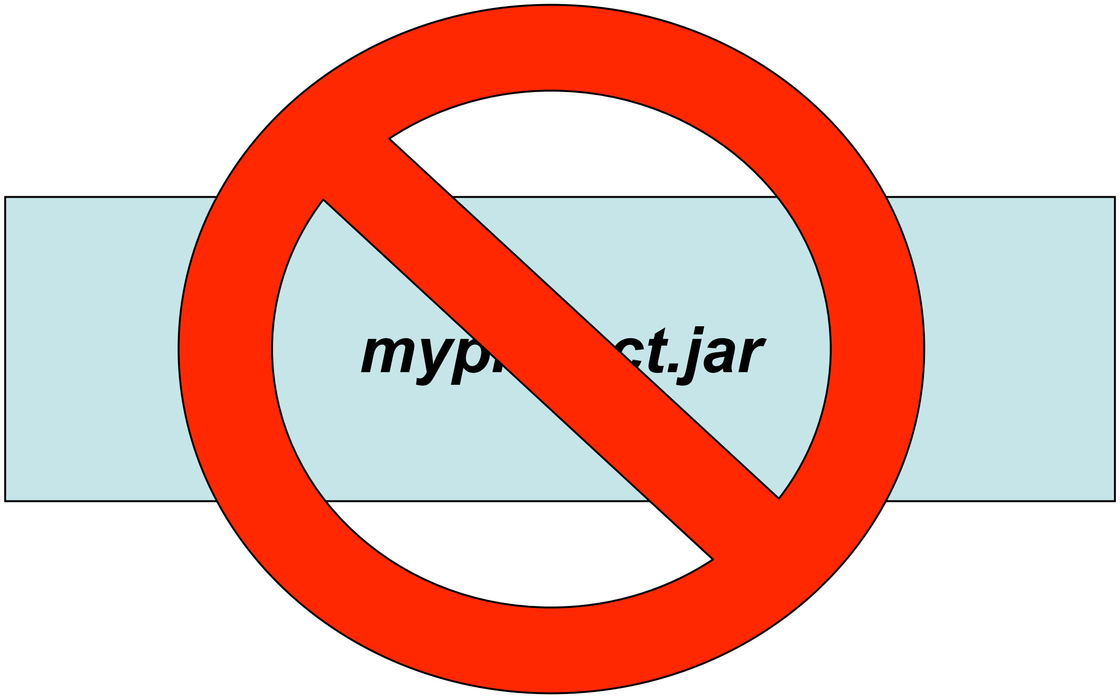
GM

Metadata


Workspace
Module
Module
Module
Project
Project
Project
Cmd
Cmd
Cmd

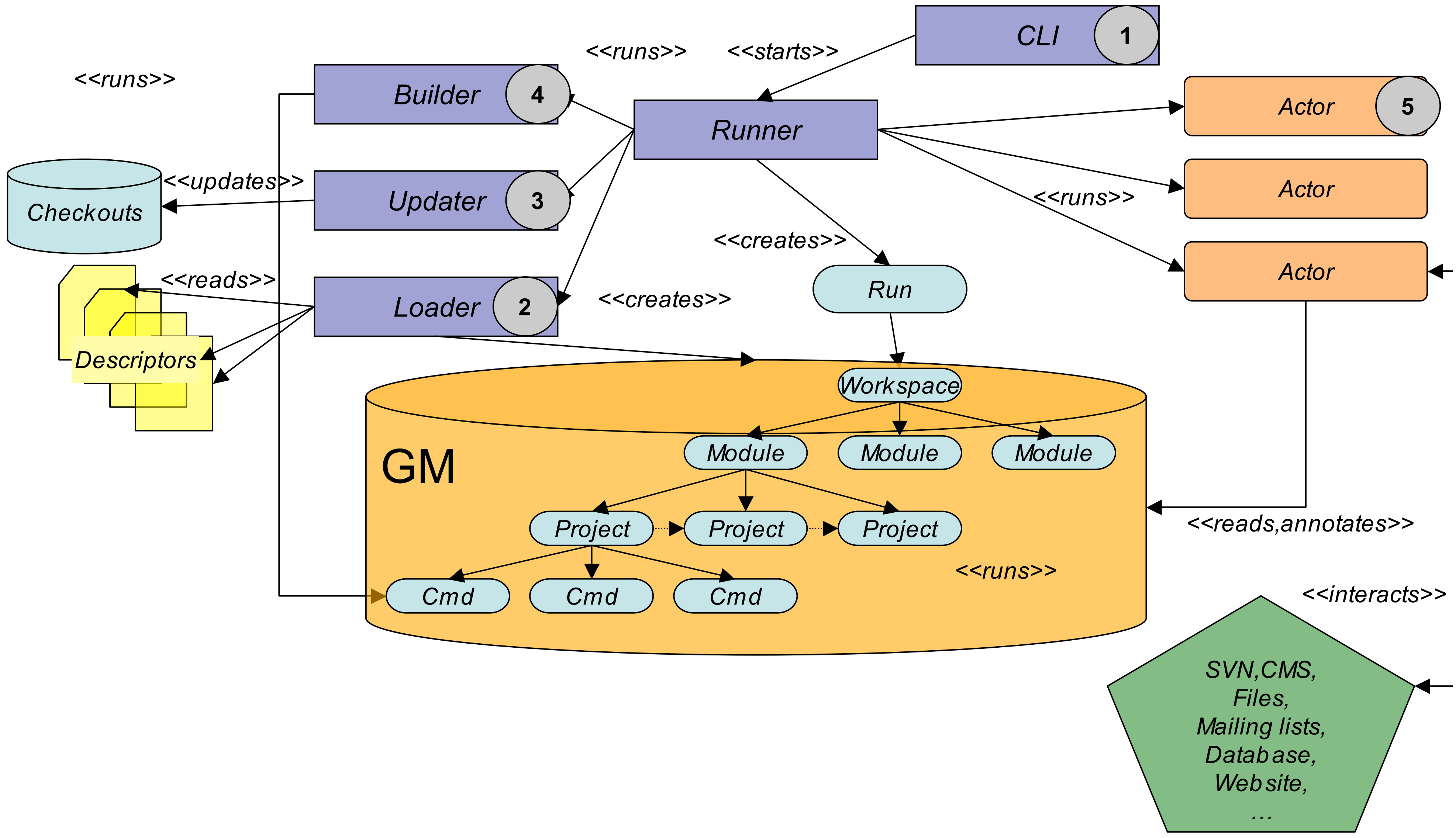


Artifact Repository



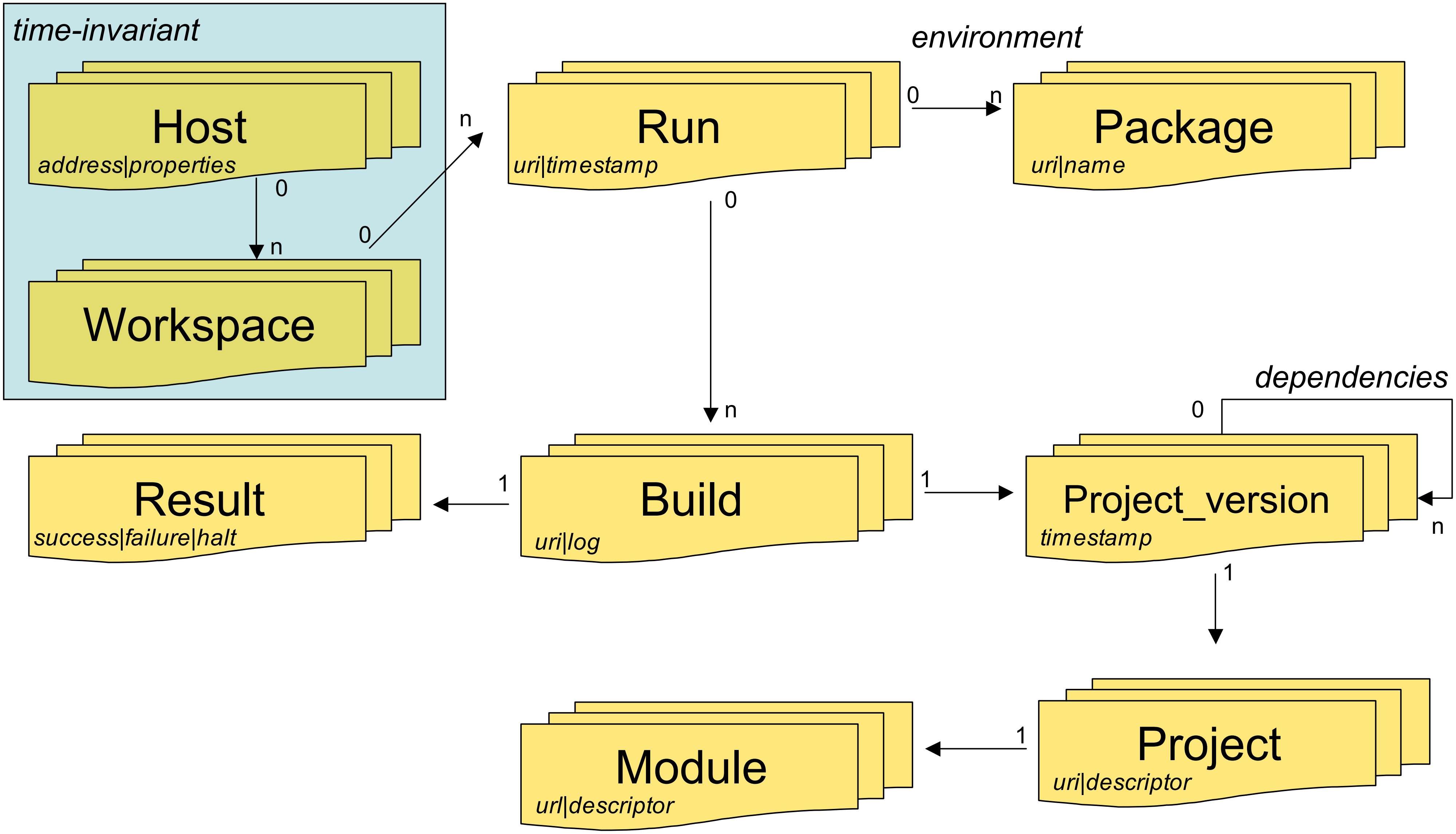
Gumpy

Gumpy


CLI
1
<<starts>>
<<runs>>
Runner
Builder
4
Updater
3
Loader
2
<<runs>>
<<updates>>
Checkouts
<<reads>>
Descriptors
<<creates>>
<<creates>>
Run
Actor
5
Actor
Actor
<<runs>>
GM
Workspace
Module
Module
Module
Project
Project
Project
Cmd
Cmd
Cmd
<<runs>>
<<reads,annotates>>
<<interacts>>
SVN,CMS,
Files,
Mailing lists,
Database,
Website,
…

RM

Runtime Metadata (RM)



time-invariant

Host
address|properties

Workspace

Run
uri|timestamp

environment

Package
uri|name

Result
success|failure|halt

Build
uri|log

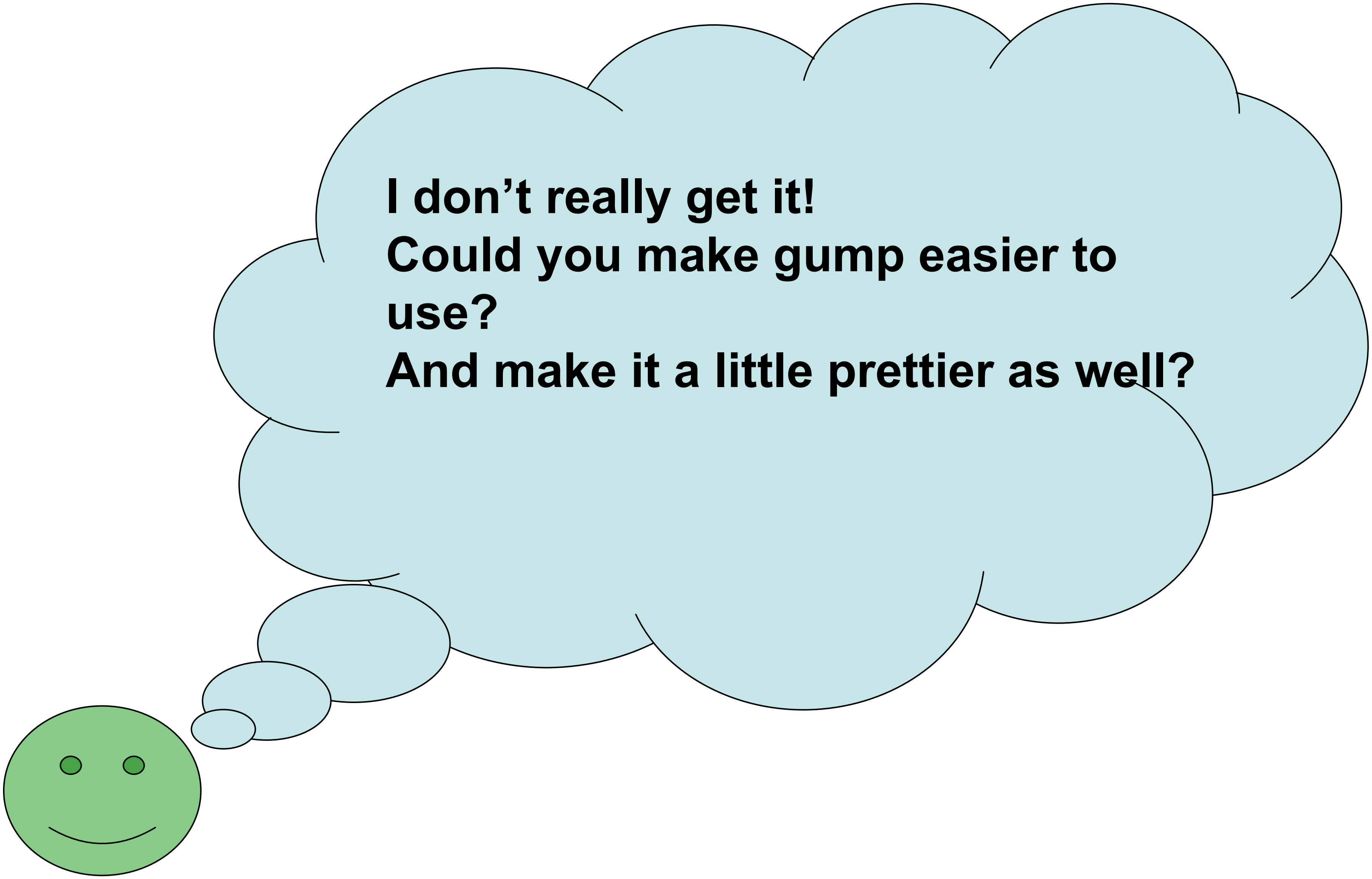
dependencies

Project_version
timestamp

Project
uri|descriptor

Module
url|descriptor

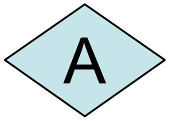
Dynagump



I don’t really get it!
Could you make gump easier to use?
And make it a little prettier as well?

Gumpy Runner algorithm (1)

Building blocks:

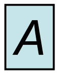


Building project "A"

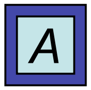
A+ *Project "A" built successfully*

A- *Project "A" failed to build successfully*

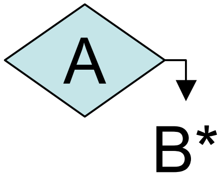
A* *Last successful build of proejct "A"*



Send e-mail to project "A"

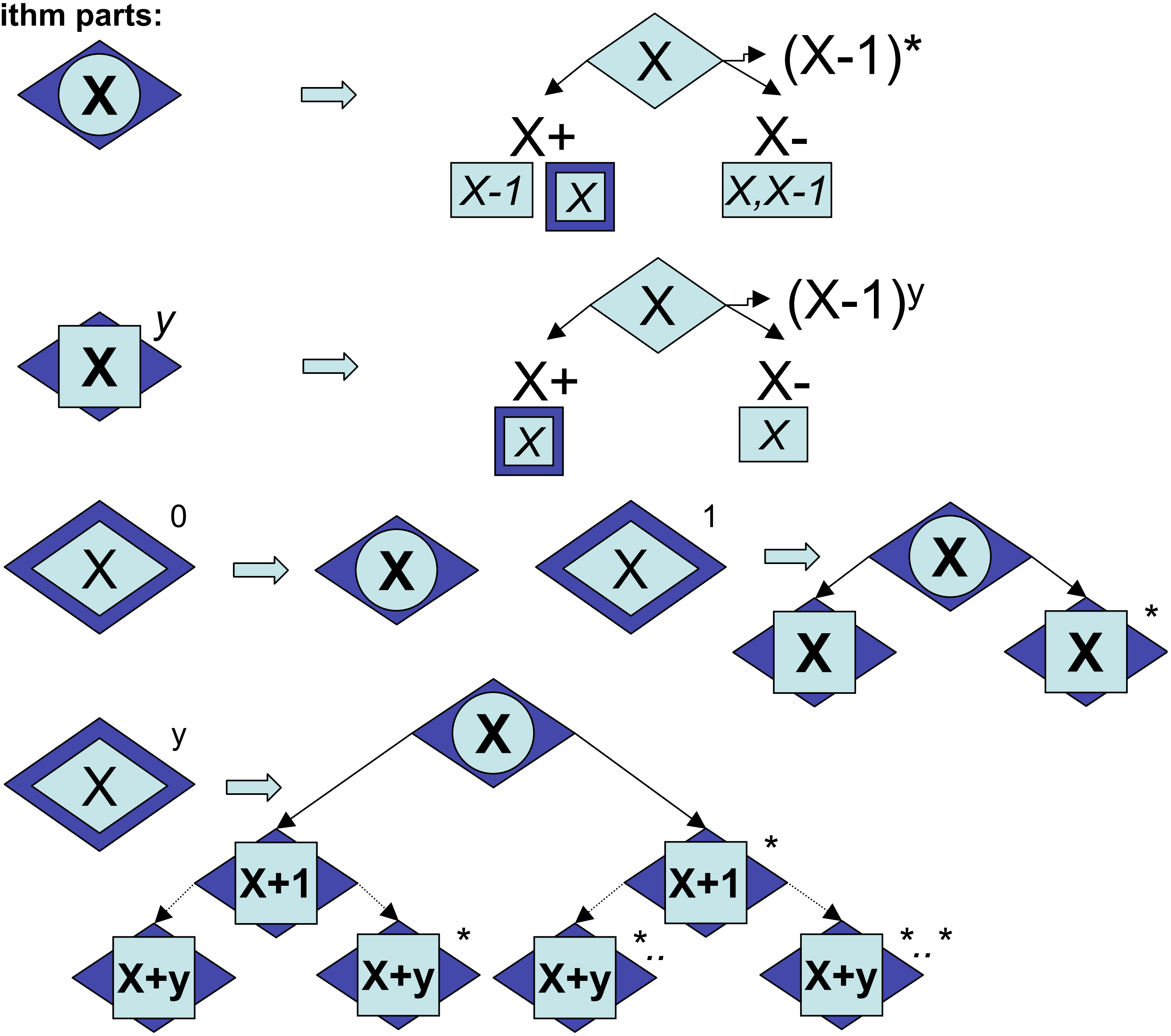


Send e-mail to project "A" if this state has persisted for "a while"



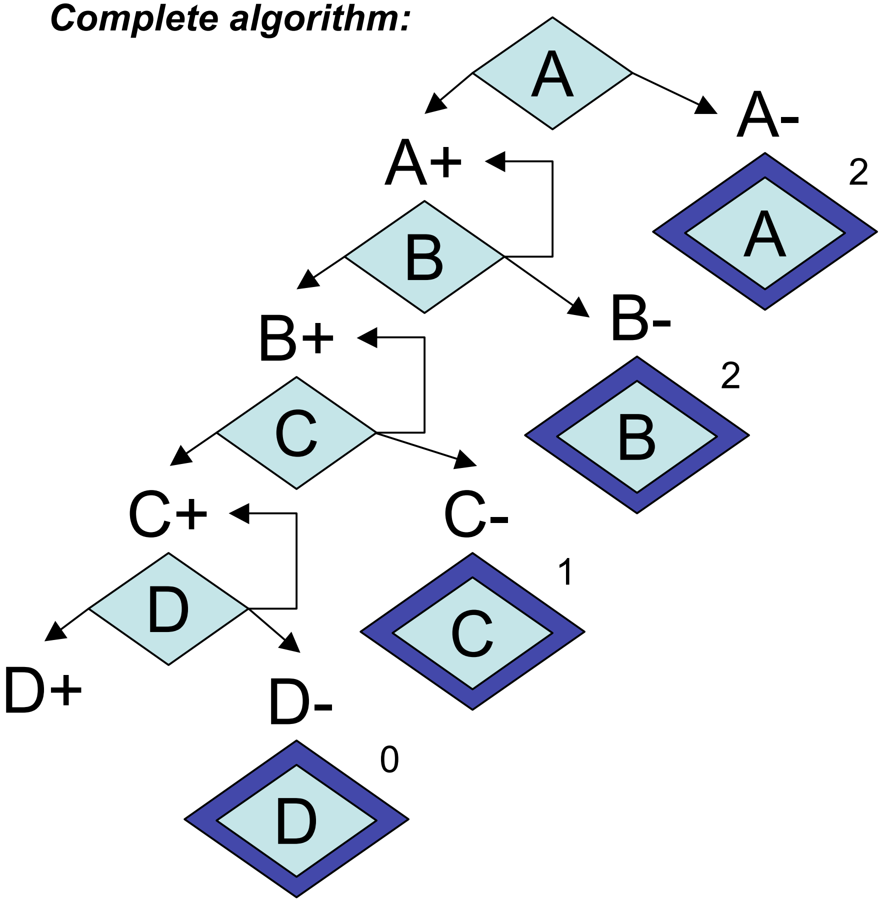
Build project "A" against "B"*

Algorithm parts:



Gumpy Runner algorithm (2)

Complete algorithm:



This tree shows the basic control flow for gumpy. "Build all projects starting from the top of the tree ("A"), if successful, use that build further on, otherwise, use one from a previous run ("A").Send e-mail to "A" or "B" based on the result of those attempts.*

There are several things not clearly shown here (like the fact that most projects have multiple dependencies), but the "basics" of detecting "cause" are in there.