Taxonomy of **Package Management** in Programming Languages and Operating Systems

Hisham Muhammad
Kong Inc. / htop / LuaRocks / GoboLinux
Lucas Correia Villa Real
IBM Research / GoboLinux
Michael Homer
Victoria University of Wellington / GoboLinux

Introduction: a real-world example

To do frontend web development on a Mac, you may need to

install client-side JavaScript components

Introduction: a real-world example

To do frontend web development on a Mac, you may need to

install client-side JavaScript components using **Bower**

Introduction:
a real-world example

To do frontend web development on a Mac, you may need to

install client-side JavaScript components using **Bower**
    install Bower using **npm**

Introduction: a real-world example

To do frontend web development on a Mac, you may need to

- install client-side JavaScript components using **Bower**
  - install Bower using **npm**
    - install npm using **Homebrew**

Introduction: a real-world example

To do frontend web development on a Mac, you may need to

- install client-side JavaScript components using **Bower**
  - install Bower using **npm**
    - install npm using **Homebrew**
      - install Homebrew using… **curl**?

Introduction: the mess we're in

The ecosystem is not getting any simpler

"Why don't we all just use RPM/APT/etc.?"

Introduction: the mess we're in

The ecosystem is not getting any simpler

- "Why don't we all just use RPM/APT/etc.?"
  - "Why don't we all just use C/Rust/Python/Haskell/etc.?"

Introduction: the mess we're in

The ecosystem is not getting any simpler

"Why don't we all just use RPM/APT/etc.?"

"Why don't we all just use C/Rust/Python/Haskell/etc.?"

"Package management is a(n un)solved problem"

Introduction: the mess we're in

The ecosystem is not getting any simpler

"Why don't we all just use RPM/APT/etc.?"

"Why don't we all just use C/Rust/Python/Haskell/etc.?"

"Package management is a(n un)solved problem"

"Programming languages are a(n un)solved problem"

Introduction: the mess we're in

The ecosystem is not getting any simpler

"Why don't we all just use RPM/APT/etc.?"

"Why don't we all just use C/Rust/Python/Haskell/etc.?"

"Package management is a(n un)solved problem"

"Programming languages are a(n un)solved problem"

We accept the multitude of languages,
why not the multitude of package managers?

Introduction: multitude of languages, multitude of managers

1. Different languages for different purposes
   Different paradigms, different trade-offs
2. We know that there is room for DSLs and general-purpose langs
3. We know how to set boundaries and make languages interact

Introduction: multitude of languages, multitude of managers

1. Different languages for different purposes
   Different paradigms, different trade-offs

2. We know that there is room for DSLs and general-purpose langs

3. We know how to set boundaries and make languages interact
   Not so much for package managers!

Introduction: multitude of languages, multitude of managers

1. Different languages for different purposes
   Different paradigms, different trade-offs

2. We know that there is room for DSLs and general-purpose langs

3. We know how to set boundaries and make languages interact
   Not so much for package managers!

Let's look at package managers
the same way we look at languages

Understanding package management

1. *Language-specific vs. Language-agnostic package managers*

Types of languages: domain-specific and general-purpose PLs

domain-specific	general area ("systems")
seen as "smaller"	seen as "complete"
defined by *inclusion* of features for the domain	defined by the *generality* of its features

Types of package managers: domain-specific and general-purpose PMs

"language-specific"	"language-agnostic"
language ecosystem	often deal with whole OS
defined by *inclusion* of features for the domain	defined by the *generality* of its features

Types of package managers: examples

language-specific	***language-agnostic***
pip (Python)	RPM (RedHat/Fedora/etc.)
RubyGems (Ruby)	dpkg/apt (Debian/Ubuntu/etc.)
npm (JavaScript)	Pacman (Arch Linux)
Cabal (Haskell)	Homebrew (macOS)
Cargo (Rust)	Nix (NixOS)
LuaRocks (Lua)	GoboLinux

Why have language-specific PMs at all?

scalability

Debian: 59,000+
Java (Maven Central): 290,000+

Ruby packages in Debian: 1,196
Ruby packages in RubyGems: 150,000+

Why have language-specific PMs at all?

portability

packaging language extensions using OS managers leads to an n-by-m explosion

Windows and Mac have no native package manager

2. *Paradigms of package management: Filesystem–oriented vs. Database–oriented*

Paradigms of programming languages: a didactic tool

It's a typical didactic device to organize PLs by "paradigms":

Imperative, functional, etc.
Procedural, object-oriented, etc.

They illustrate design choices ("how to represent computation")

and design choices bring design trade-offs

Paradigms of package managers: a central design choice

"How to map files into packages"

Using the file hierarchy itself: **filesystem-oriented**

Externally to the files managed: **database-oriented**

Paradigms of package management: examples

filesystem-oriented	***database-oriented***
Homebrew (macOS)	RPM (RedHat/Fedora/etc.)
npm (JavaScript)	pip (Python)
Nix (NixOS)	dpkg/apt (Debian/Ubuntu/etc.)
Cargo (Rust)	Cabal (Haskell)
GoboLinux	Pacman (Arch Linux)
LuaRocks 1.x	LuaRocks 2.x+

Paradigms of package management: trade-offs compared

filesystem-oriented	***database-oriented***
define the structure: designed to avoid clashes, keep mapping in sync is **trivial**	**adapt** to pre-existing structure: needs to forbid clashes, **fragile** if goes out of sync
applications need to **use** the filesystem structure defined: runtime lookup may be **complex**	applications can be **unaware** of manager: runtime lookup can be **trivial**
more often language-specific	most distro managers

	Filesystem-oriented	Database-oriented
Language-agnostic	Homebrew (macOS), GNU Stow, Nix, Encap, PBI 8 (PC-BSD), GoboLinux	RPM (RedHat/Fedora/etc.), dpkg/apt (Debian/Ubuntu/etc.) PBI 9 (PC-BSD), Pacman (ArchLinux)
Language-specific	npm (server-side JavaScript), Bower (client-side JavaScript) RubyGems (Ruby), Cargo (Rust), LuaRocks 1.x (Lua)	Cabal (Haskell), pip (Python), LuaRocks 2.x (Lua)

Figure 1. A package manager taxonomy, with representative examples

	Language-specific managers			
Package managers	**npm**	**RubyGems**	**NuGet**	**LuaRocks**
Portability	OS-independent (all Unix, Windows)			
Installs code written in	JS family, C/C++	Ruby, C/C++, JVM family	any .NET, C++	Lua family, C/C++
Files managed	JS scripts, JS modules	Ruby scripts, Ruby modules	.NET and native packages	Lua scripts, Lua modules
Supports per-user install	yes			

	Language-agnostic managers			
Package managers	**Nix**	**Homebrew**	**RPM**	**GoboLinux**
Portability	Linux/macOS	macOS/Linux	Linux/AIX	Linux/Cygwin/OSX
Installs code written in	any language			
Files managed	all kinds			
Supports per-user install	yes	no*	no	yes

* different installation prefixes are supported but `/usr/local` is strongly recommended.

Figure 2. Contrasting language-specific and language-agnostic package managers

3. *Integration between languages vs. Integration between package managers*

Integration between languages: dynamic and static integrations

Dynamic (at runtime):

calling conventions, LuaJIT FFI, Python cffi...

Static (at compile time):

linking formats, Lua C/API <lua.h>, PyObject API <Python.h>...

Integration between package managers: dynamic and static integrations

none?

Integration between package managers dynamic and static integrations

Dynamic (at runtime):

> what happens if you install a package that uses a runtime FFI and the C library is not installed?

Static (at compile time):

> what happens if you install a bindings package and the headers of the library you're binding to are not installed?

Experiences with package management

GoboLinux: fs-oriented OS package management

Linux distribution project started in 2003

Each package installed under a separate prefix:

```
    /Programs/GCC/6.2.0/bin/gcc
    /Programs/Glibc/2.24/lib/libc.so.6
```

A tree of symlinks provides compatibility and runtime resolution

Running on this computer!

Informed the design of Homebrew ("the GoboLinux way")

LuaRocks: a language-specific package manager

Package manager for the Lua programming language (2007-current)

LuaRocks 1.x: filesystem-oriented design
- informed by GoboLinux design: multiple versions, no file conflicts!
- required runtime cooperation: custom package loader for require()

LuaRocks 2.x+: database-oriented design
- lots of code to deal with file conflicts
- no runtime cooperation required: works with Lua out-of-the-box!
- maintained optional custom package loader (does way more work)

LuaRocks and GoboLinux Aliens: bridging OS and PL package managers

LuaRocks: minimal PL-to-OS management awareness

```
external_dependencies = { MYSQL = { header = "mysql.h" }
```

can be used for both FFI and C-API dependencies
to gracefully fail ahead-of-time — doesn't actually install

GoboLinux Aliens: OS-to-PL management awareness

GoboLinux packages can depend on PL packages, uses PL managers:

```
Cabal:mtl
CPAN:XML::Parser 0.4.1
```

Conclusion: multitude of languages, multitude of managers

1. Different package managers for different purposes
   Different paradigms, different trade-offs
2. There is room for PL-specific and OS-wide PMs
3. We need to set boundaries and make package managers interact

Conclusion: multitude of languages, multitude of managers

1. Different package managers for different purposes
   Different paradigms, different trade-offs

2. There is room for PL-specific and OS-wide PMs

3. We need to set boundaries and make package managers interact
   Time for a package manager protocol?

Conclusion: multitude of languages, multitude of managers

1. Different package managers for different purposes
   Different paradigms, different trade-offs
2. There is room for PL-specific and OS-wide PMs
3. We need to set boundaries and make package managers interact
   Time for a package manager protocol?

Let's look at package managers
the same way we look at languages