`impact` – A Modelica® Package Manager

Michael Tiller
Xogeny Inc., USA
`michael.tiller@xogeny.com`

Dietmar Winkler
Telemark University College, Norway
`dietmar.winkler@hit.no`

To manage complexity, modern programming languages use organizational units to group code related by some common purpose. Depending on the programming language, these units might be called libraries, packages or modules. But they all attempt to encapsulate functionality to promote modular code and reusability. For the remainder of this paper, we will simply refer to these organizational units as *packages* (as they are called in Modelica).

Also common to many modern programming languages are tools to manage these packages. These tools are generally called *package managers* and they allow developers to quickly "fetch" any packages they may need for a given project. The main functions of package managers are to allow developers to search, install, update and uninstall packages with a simple command-line or graphical interface. In the Java world, the most common package manager is `maven`. For Python, tools like `easy_install`[1] and `pip`[2] are used for managing packages. For client-side web development, `bower` is used. For server-side JavaScript, the tool of choice is `npm`[3]. For compiled languages, these package managers often include some additional build functionality as well.

This paper introduces *`impact`*, a package manager for Modelica. Using `impact`, Modelica users and developers can quickly search for, install and update Modelica libraries. In this paper, we will discuss the functionality provided by `impact`. In addition, we will discuss how the functionality was implemented. As part of this we will discuss the importance of collaborative platforms, like GitHub[4] in our case, for providing a means for collecting, curating and distributing packages within a community of developers.

The `impact` package manager is provided to the Modelica community as a free, open-source tool. Furthermore, the protocols involved are all documented and we encourage tool vendors to integrate them into their own tools so they can provide the same searching, updating and installation capabilities that the command-line tool provides.

References

[1] easy_install. *Easily download, build, install, upgrade, and uninstall Python packages.* 2014. URL: `https://pypi.python.org/pypi/setuptools`.

[2] pip. *A tool for installing and managing Python packages.* 2014. URL: `http://www.pip-installer.org`.

[3] npm. *Node Packaged Modules.* 2014. URL: `https://npmjs.org/`.

[4] GitHub. *Build software better, together.* 2014. URL: `https://github.com/`.