Team Apollo
Project Package-Bird
Description

CPTS 421 – Software Design Project I

By: Denish Oleke, Elisha Aguilera, Setenay Guner
Mentor: Brandon Busby
Instructor: Bolong Zeng

September 24, 2021

Project Description

Package-bird is a dependency management solution consisting of a client installed and operated by a user, which operates on a manifest file in a project directory, request the packages from a specified remote registry, which in turn generates a dependency graph by crawling packages contained in the registry. If all packages are valid, the registry transmits the compressed packages back to the client, which then extracts and installs them in the project directory.

The utility of this solution is two-fold: the end user can quickly configure a project directory, needing only a manifest file, the client, and connection to the remote registry. Organizations working with the user can quickly configure and deploy the registry as an internal solution to package management, as opposed to relying on public, externally controlled registries.

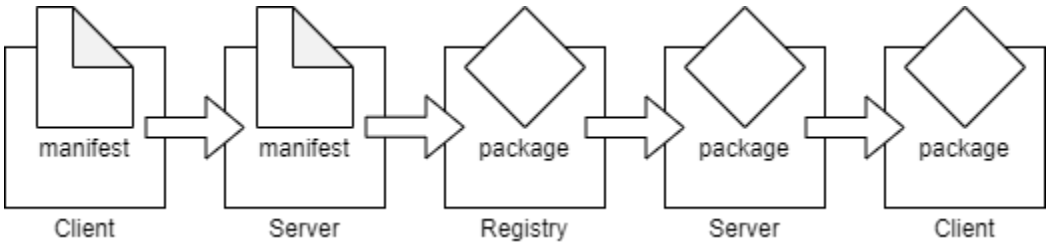

Figure 1 – General Project Workflow

Features

Functionality provided by this project is multi-tiered and oriented around project configuration, but primarily:

- Create and parse manifest files to find and track project dependencies.
- Request dependencies from a centrally managed internal registry.
- Generate dependency graph and validity integrity of co-dependent packages.
- Transmit requested compressed packages to the client machine.
- Configure and install packages in a standardized format for the project.

Limitations

The uncertainty and management of the project will likely be the greatest limitations facing a complete and successful limitation. Many examples of similar working implementations can be found in the literature section of this document, but in comparison, these examples were implemented by teams of skilled professionals with expanded resources.

Our team faces multiple different items that compete for time to be spent on this project, forecasting that the greatest risk to the project will likely be poor time management. Beyond that, there are components of the project that require some networked technologies and concepts which members are somewhat inexperienced with.

Gantt Chart

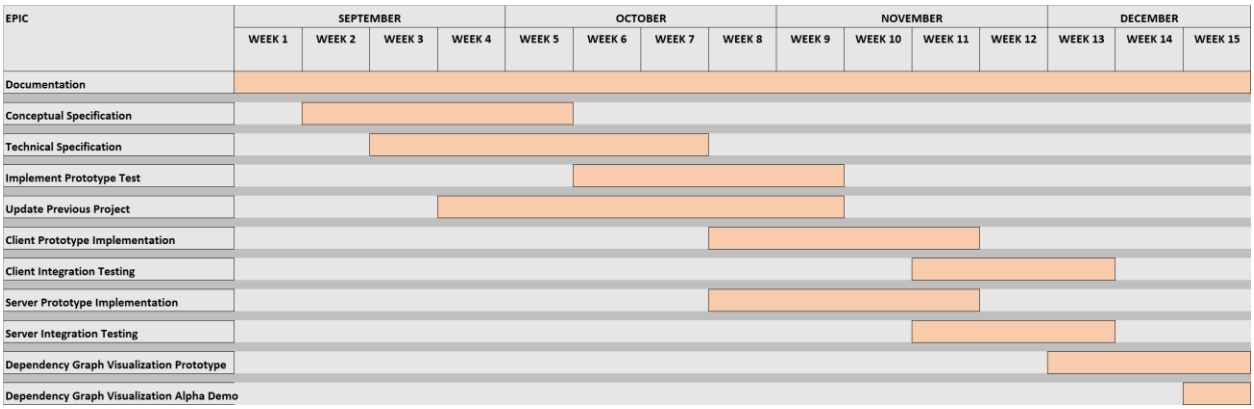

Figure 2 – Gantt Chart for first semester development outline and deliverables.

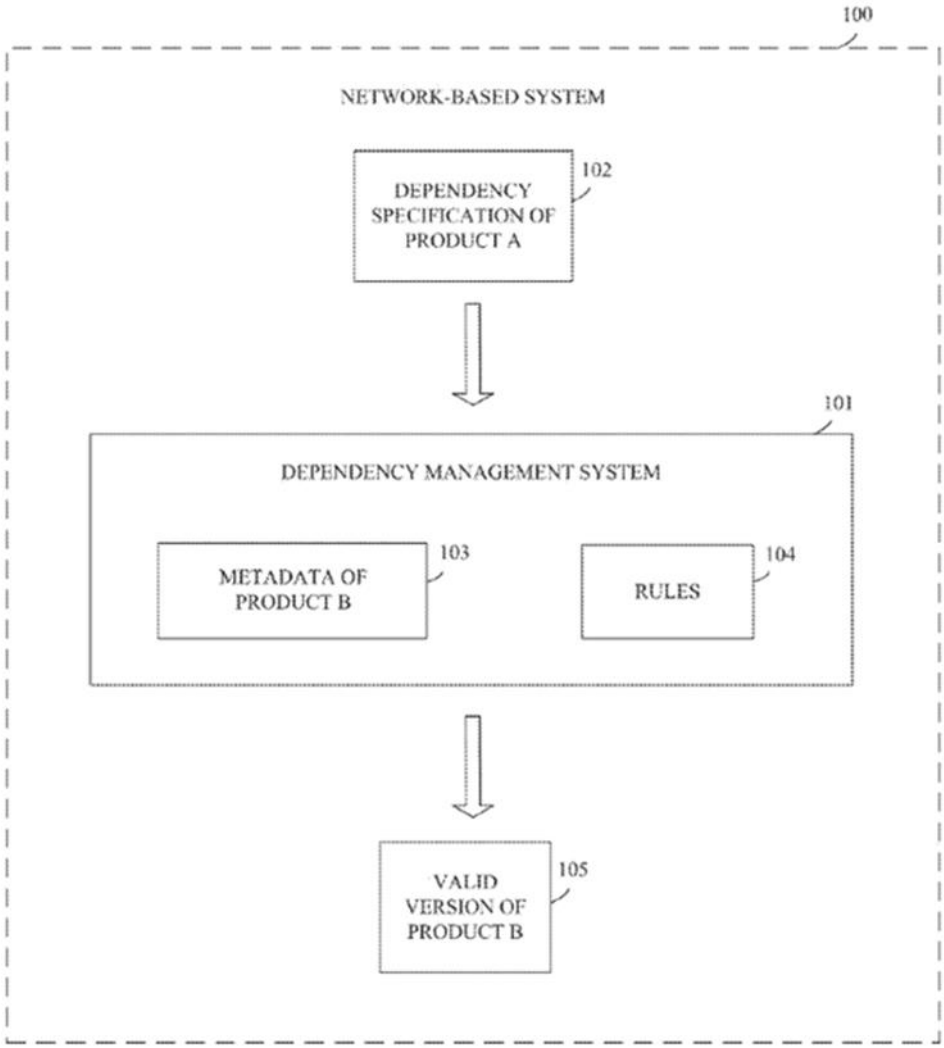

Figure 3: example of a dependency management system from the Software Dependency Patent [1].

Literature Review

Modern software applications are designed to use other, existing software components. Dependency on other software components can result in numerous dependency issues, such as conflicts and a circular or long chain of dependencies.

After we researched the current dependency management tools, we have made this table to show the different uses of each tool.

Tools	Description
NuGet	A package manager for .NET. When used, it installs a package and copies the library to the local solution and updates your project. When a package is removed, it reverses the changes and clears the clutter left behind.[2]
Bower	Developed for web package management for CSS and JavaScript. It can manage components containing CSS, JS, HTML, and images. Bower installs the right version of packages and their dependencies. [3]
pip	Written for Python to install and keep track of packages. Offers local or remote configuration and a CLI. [4]
RubyGems	A package manager used for web developing. It allows extending or modifying functionality in Ruby applications. CLI is available to automate tasks.[5]
Jam	This is a browser-based package management for JS. It pulls all the dependencies into a JS file for easy manipulation. Still offers a CLI for automation. [6]
NPM	Another JS CLI tool for updating and optimizing dependencies is through the terminal. Uses a package.json file to keep track and update automatically when building projects. [7]

Table 1- Literature and Comparison

Stakeholder Identification and Considerations

Most dependency and package management applications are limited to the scope of technical developers and hobbyist contributing to the software projects, because of the potential increases towards developer productivity, the secondary stakeholders may often be the end users of whatever product the company is developing. Below is a table describing the various stakeholders, both primary and secondary, that may benefit from this project.

Stakeholder	Description
Team Apollo	Team developing the dependency management tool.
Brandon Busby	Team mentor guiding through the development.
Developers	Consumers of the results of the project.
Company?	General population within the company.
Company's Product Users	Users of the software developed by the company.

Table 2- Stakeholders table

References (MLA)

1. Pillgram-Larsen, Jens, et al. *Systems and Methods for Software Dependency Management* . 24 May 2016, patft.uspto.gov/netacgi/nph-Parser?Sect1=PTO1&Sect2=HITOFF&d=PALL&p=1&u=%2Fnetahtml%2FPTO%2Fsrchnum.htm&r=1&f=G&l=50&s1=9348582.PN.&OS=PN/9348582&RS=PN/9348582.
2. Jon Douglas, et al. "What Is NuGet and What Does It Do?" *What Is NuGet and What Does It Do? | Microsoft Docs*, 24 Apr. 2019, docs.microsoft.com/en-us/nuget/what-is-nuget.
3. MacCaw, Alex, and Jacob Thornton. "About · Bower." *Bower*, 2012, bower.io/docs/about/.
4. "User Guide." *User Guide - Pip Documentation v21.2.4*, 2011, pip.pypa.io/en/stable/user_guide/.
5. "Rubygems Guides." *Guides*, 2020, guides.rubygems.org/.
6. JonDouglas. "What Is NuGet and What Does It Do?" *What Is NuGet and What Does It Do? | Microsoft Docs*, 24 Apr. 2019, docs.microsoft.com/en-us/nuget/what-is-nuget.
7. "About Npm." *Npm Docs*, 2016, docs.npmjs.com/about-npm.