UTSA®
The University of Texas at San Antonio™

Towards Building a CyberInfrastructure for Facilitating the Assessment, Dissemination, Discovery, & Reuse of Software and Data Products

September 23, 2021

Ritu Arora
Email: ritu.arora@utsa.edu

- Application Software (e.g., MuST and Enzo-E)
- System Management Software (e.g., PFSTRASE)
- Programming Language Environments, and Runtime Systems (e.g., Interactive Parallelization Tool, and Eclipse PTP)
- Software Libraries (e.g., Gunrock, libkrylov, and MVAPICH2)
- Measurement and Monitoring (e.g., XDMoD, XALT, and TACC stats)
- Web Portals, middleware, and web-accessible products (e.g., Open OnDemand)
- Networking (e.g., EdgeVPN and CINES)
- Simulation Platforms (e.g., Chrono)
- Fault-tolerance (e.g., DMTCP)
- Software verification and validation (e.g., C11Tester, and roundoff-error-free algorithms)
- Workflow Management (e.g., Event Workflow Management System)
- Data Management (e.g., DataSwarm and Clowder)
- Data Visualization (e.g., SAGE3 and MetPy)
- Database and Data Processing (e.g., Hustle)

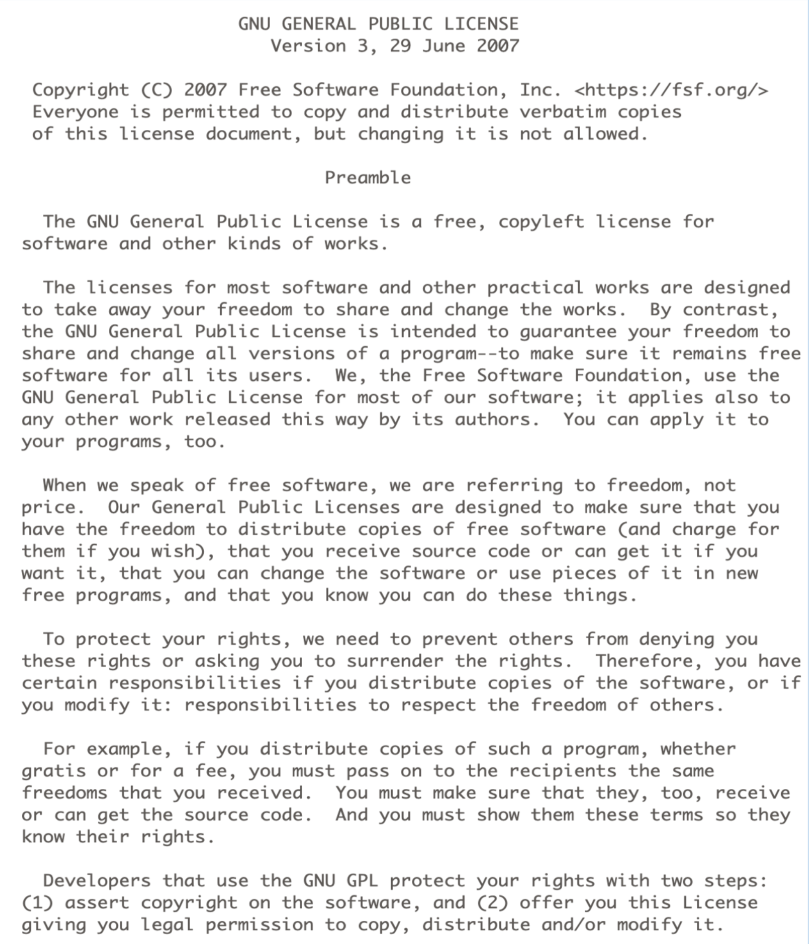
Diverse Landscape of Software and Data Products Funded by NSF

- How do we currently discover the open-source software and data products funded by NSF?
- How can we transparently obtain metrics related to product adoption while respecting the privacy of the product users?
  - Note: not all products may be running in controlled or managed environments
    - Less than 50% of all the NSF funded products directly run on HPC systems
  - Note: there are different types of environments in which a product can be used – e.g., desktop, cloud computing, supercomputing, stand-alone servers or VMs
- Do metrics such as “number of downloads” for software products present an accurate picture of their adoption in the community?
- How easy or difficult it is to discover a product and to obtain a working copy of the code that can be tested in different computing environments (think in terms of total time involved)?
- How do we currently check the compatibility of any two software products for integration and reuse?
  - Compare their licenses for permissions, constraints, and conditions regarding reuse/redistribution/modification
  - Compare their software stack for ease of integration and interoperability
- How do we currently decide which licenses are best for our products?
  - Note: think about the future potential of commercialization or reuse of the products

Lack of a Central Registry for Assessment, Dissemination, Discovery, & Reuse of the Products

Property	GPL-3.0-only Product	LGPL-3.0-only Product
Can reuse/use?	Yes	Yes
Can distribute?	Yes	Yes
Can place a warranty?	Yes	No
Cannot use trademark?	Yes	No
Is Permissive?	No	No
Cannot hold liable?	Yes	Yes
Can sublicense?	No	Yes

What would be the final license after mixing existing products with new code/components?



GNU GENERAL PUBLIC LICENSE
Version 3, 29 June 2007

Copyright (C) 2007 Free Software Foundation, Inc. <https://fsf.org/>
Everyone is permitted to copy and distribute verbatim copies
of this license document, but changing it is not allowed.

Preamble

The GNU General Public License is a free, copyleft license for software and other kinds of works.

The licenses for most software and other practical works are designed to take away your freedom to share and change the works. By contrast, the GNU General Public License is intended to guarantee your freedom to share and change all versions of a program--to make sure it remains free software for all its users. We, the Free Software Foundation, use the GNU General Public License for most of our software; it applies also to any other work released this way by its authors. You can apply it to your programs, too.

When we speak of free software, we are referring to freedom, not price. Our General Public Licenses are designed to make sure that you have the freedom to distribute copies of free software (and charge for them if you wish), that you receive source code or can get it if you want it, that you can change the software or use pieces of it in new free programs, and that you know you can do these things.

To protect your rights, we need to prevent others from denying you these rights or asking you to surrender the rights. Therefore, you have certain responsibilities if you distribute copies of the software, or if you modify it: responsibilities to respect the freedom of others.

For example, if you distribute copies of such a program, whether gratis or for a fee, you must pass on to the recipients the same freedoms that you received. You must make sure that they, too, receive or can get the source code. And you must show them these terms so they know their rights.

Developers that use the GNU GPL protect your rights with two steps: (1) assert copyright on the software, and (2) offer you this License giving you legal permission to copy, distribute and/or modify it.

Note: typically, you may need to read the license file which may look something like this

On the Topic of Checking License Compatibility

Developing automated methods for (1) collecting the evaluations metrics of the software and data products funded by NSF, (2) checking the compatibility of the products for integration and interoperability, and (3) making the products discoverable through a central registry

- **iTracker**
- **CompChecker**
- **Catalog**

This project has been generously funded by NSF through award #2037661 (for UTSA) and award # 2037656 (for SDSC).

UTSA Team

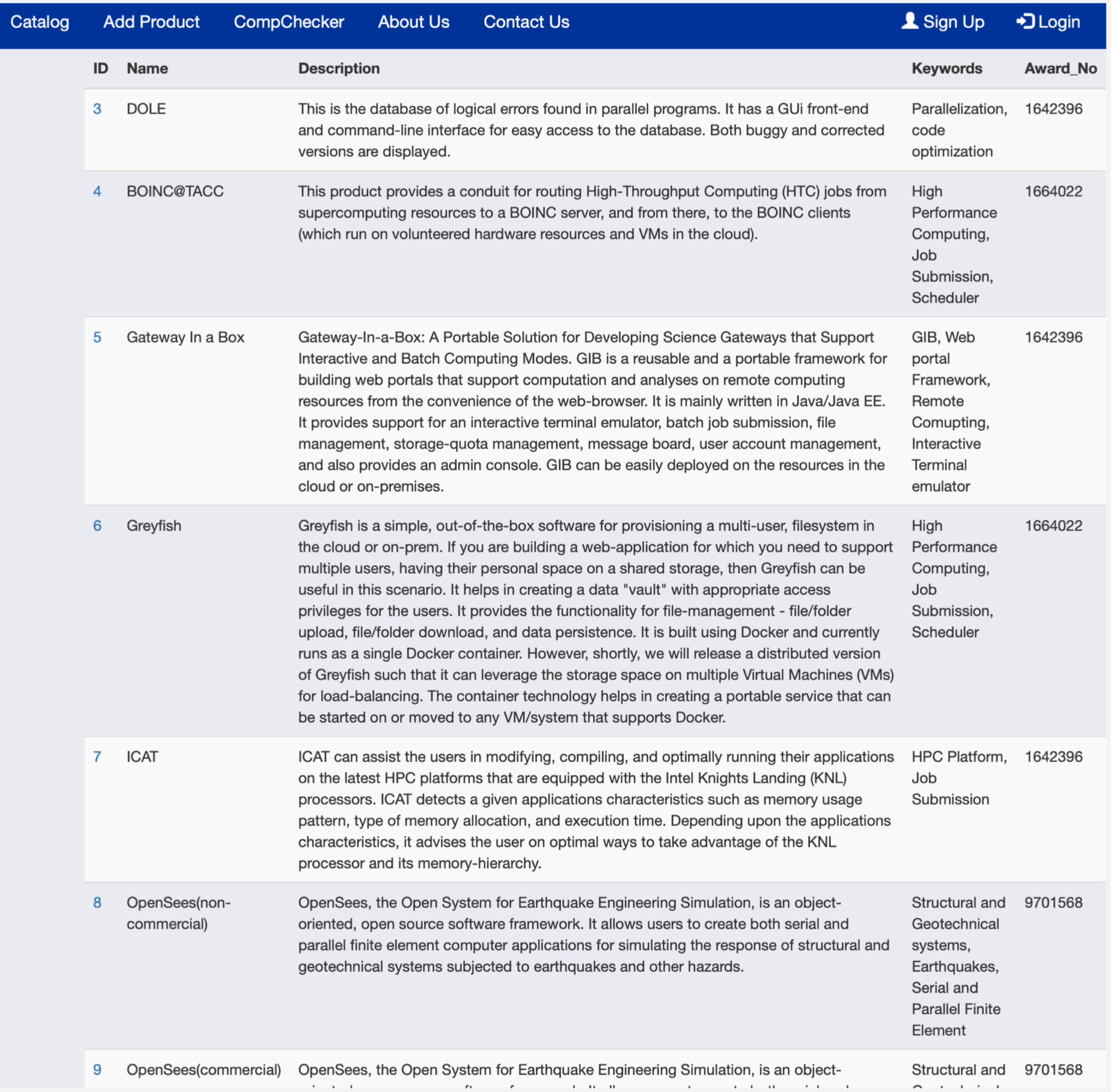
Ritu Arora

Saumya Shah

Jaidipkumar Patel

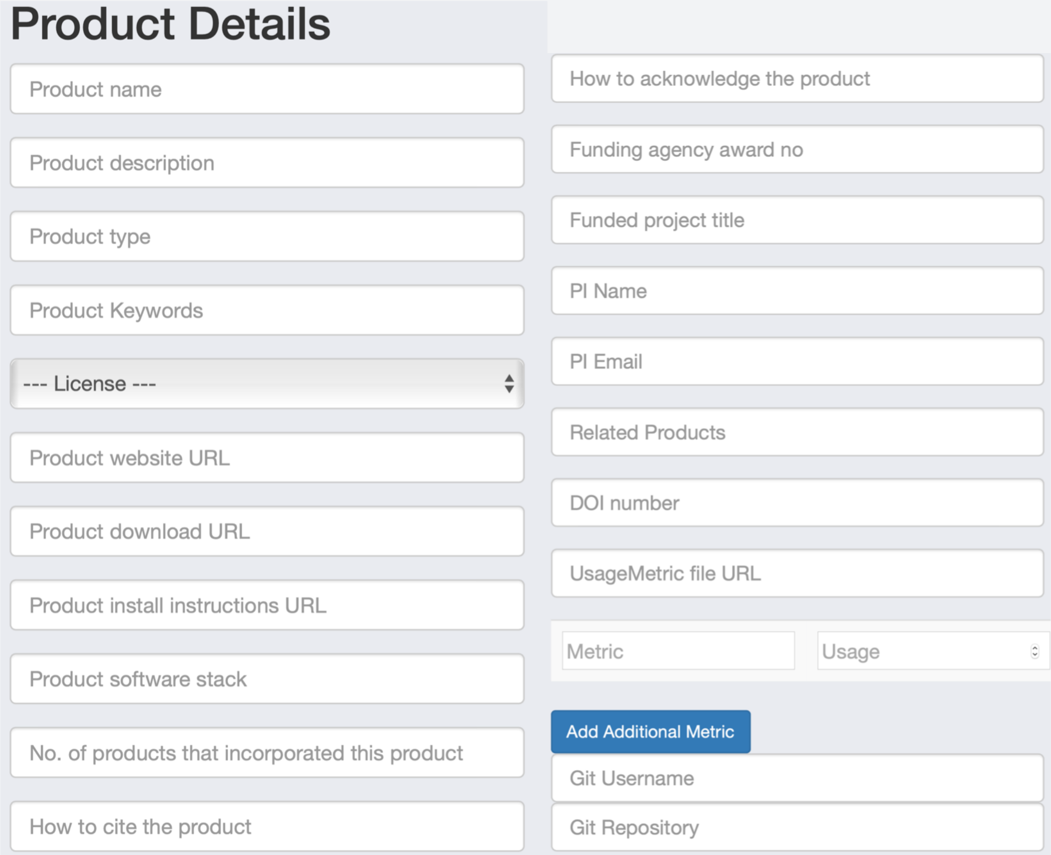
SDSC Team

- **Subhashini Sivagnanam**
- **Manu Shantharam**



ID	Name	Description	Keywords	Award_No
3	DOLE	This is the database of logical errors found in parallel programs. It has a GUi front-end and command-line interface for easy access to the database. Both buggy and corrected versions are displayed.	Parallelization, code optimization	1642396
4	BOINC@TACC	This product provides a conduit for routing High-Throughput Computing (HTC) jobs from supercomputing resources to a BOINC server, and from there, to the BOINC clients (which run on volunteered hardware resources and VMs in the cloud).	High Performance Computing, Job Submission, Scheduler	1664022
5	Gateway In a Box	Gateway-In-a-Box: A Portable Solution for Developing Science Gateways that Support Interactive and Batch Computing Modes. GIB is a reusable and a portable framework for building web portals that support computation and analyses on remote computing resources from the convenience of the web-browser. It is mainly written in Java/Java EE. It provides support for an interactive terminal emulator, batch job submission, file management, storage-quota management, message board, user account management, and also provides an admin console. GIB can be easily deployed on the resources in the cloud or on-premises.	GIB, Web portal Framework, Remote Comupting, Interactive Terminal emulator	1642396
6	Greyfish	Greyfish is a simple, out-of-the-box software for provisioning a multi-user, filesystem in the cloud or on-prem. If you are building a web-application for which you need to support multiple users, having their personal space on a shared storage, then Greyfish can be useful in this scenario. It helps in creating a data "vault" with appropriate access privileges for the users. It provides the functionality for file-management - file/folder upload, file/folder download, and data persistence. It is built using Docker and currently runs as a single Docker container. However, shortly, we will release a distributed version of Greyfish such that it can leverage the storage space on multiple Virtual Machines (VMs) for load-balancing. The container technology helps in creating a portable service that can be started on or moved to any VM/system that supports Docker.	High Performance Computing, Job Submission, Scheduler	1664022
7	ICAT	ICAT can assist the users in modifying, compiling, and optimally running their applications on the latest HPC platforms that are equipped with the Intel Knights Landing (KNL) processors. ICAT detects a given applications characteristics such as memory usage pattern, type of memory allocation, and execution time. Depending upon the applications characteristics, it advises the user on optimal ways to take advantage of the KNL processor and its memory-hierarchy.	HPC Platform, Job Submission	1642396
8	OpenSees(non-commercial)	OpenSees, the Open System for Earthquake Engineering Simulation, is an object-oriented, open source software framework. It allows users to create both serial and parallel finite element computer applications for simulating the response of structural and geotechnical systems subjected to earthquakes and other hazards.	Structural and Geotechnical systems, Earthquakes, Serial and Parallel Finite Element	9701568
9	OpenSees(commercial)	OpenSees, the Open System for Earthquake Engineering Simulation, is an object-	Structural and	9701568

UTSA side of the prototype is named as Opuntia (pronounced as up-un-chhia)



Product Details

Product name

Product description

Product type

Product Keywords

--- License ---

Product website URL

Product download URL

Product install instructions URL

Product software stack

No. of products that incorporated this product

How to cite the product

How to acknowledge the product

Funding agency award no

Funded project title

PI Name

PI Email

Related Products

DOI number

UsageMetric file URL

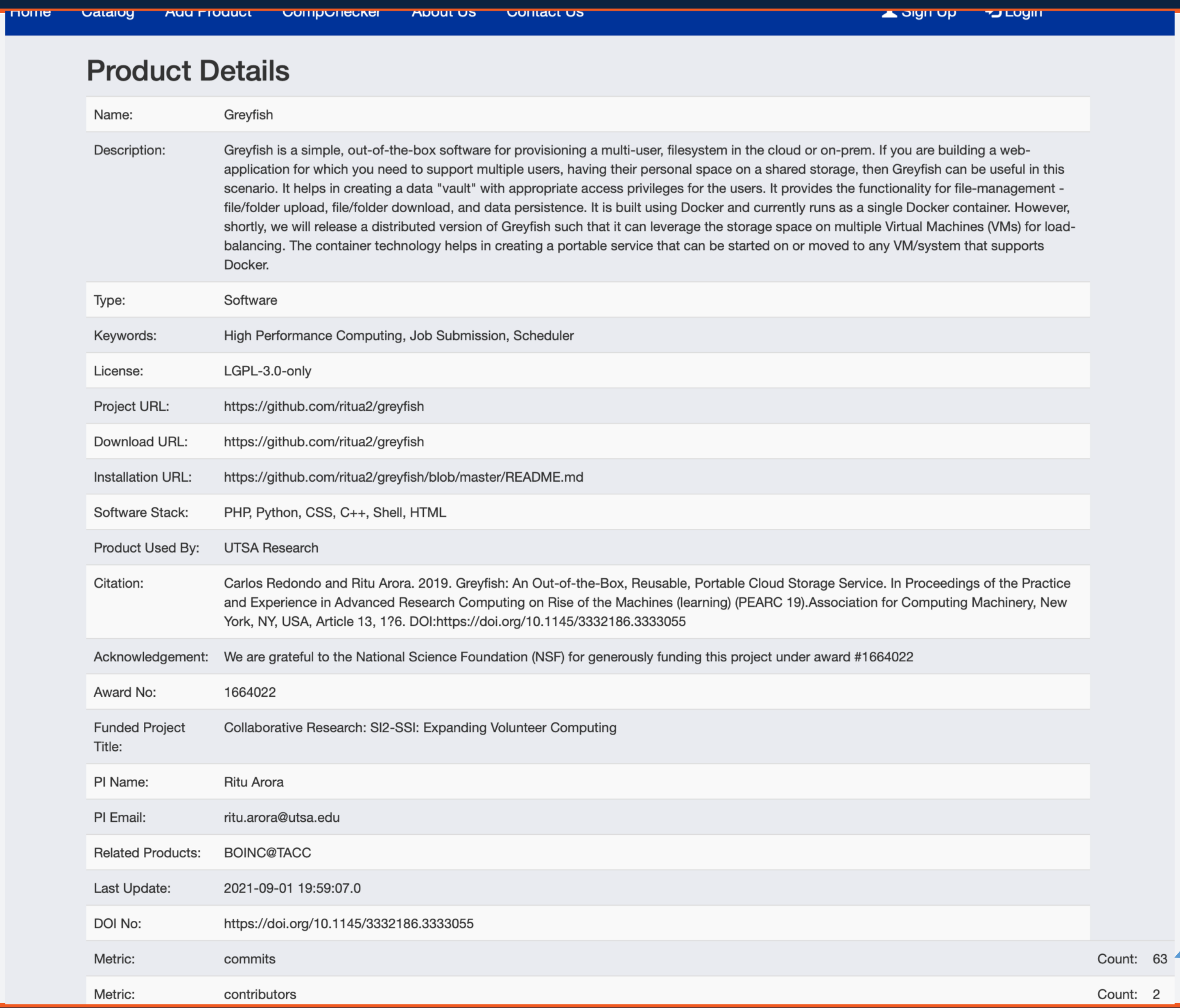
Metric | Usage

Git Username

Git Repository

Product owners contribute to populating the details of their products in their products' catalog page – their participation is critical to the success of this project

Sample Catalog Entries and Template for Capturing Product Details



Product Details

Name:	Greyfish		
Description:	Greyfish is a simple, out-of-the-box software for provisioning a multi-user, filesystem in the cloud or on-prem. If you are building a web-application for which you need to support multiple users, having their personal space on a shared storage, then Greyfish can be useful in this scenario. It helps in creating a data "vault" with appropriate access privileges for the users. It provides the functionality for file-management - file/folder upload, file/folder download, and data persistence. It is built using Docker and currently runs as a single Docker container. However, shortly, we will release a distributed version of Greyfish such that it can leverage the storage space on multiple Virtual Machines (VMs) for load-balancing. The container technology helps in creating a portable service that can be started on or moved to any VM/system that supports Docker.		
Type:	Software		
Keywords:	High Performance Computing, Job Submission, Scheduler		
License:	LGPL-3.0-only		
Project URL:	https://github.com/ritua2/greyfish		
Download URL:	https://github.com/ritua2/greyfish		
Installation URL:	https://github.com/ritua2/greyfish/blob/master/README.md		
Software Stack:	PHP, Python, CSS, C++, Shell, HTML		
Product Used By:	UTSA Research		
Citation:	Carlos Redondo and Ritu Arora. 2019. Greyfish: An Out-of-the-Box, Reusable, Portable Cloud Storage Service. In Proceedings of the Practice and Experience in Advanced Research Computing on Rise of the Machines (learning) (PEARC 19).Association for Computing Machinery, New York, NY, USA, Article 13, 1?6. DOI:https://doi.org/10.1145/3332186.3333055		
Acknowledgement:	We are grateful to the National Science Foundation (NSF) for generously funding this project under award #1664022		
Award No:	1664022		
Funded Project Title:	Collaborative Research: SI2-SSI: Expanding Volunteer Computing		
PI Name:	Ritu Arora		
PI Email:	ritu.arora@utsa.edu		
Related Products:	BOINC@TACC		
Last Update:	2021-09-01 19:59:07.0		
DOI No:	https://doi.org/10.1145/3332186.3333055		
Metric:	commits	Count:	63
Metric:	contributors	Count:	2

- A hybrid approach is being developed for <u>populating the product details</u> for a cataloged product – both automatic and manual

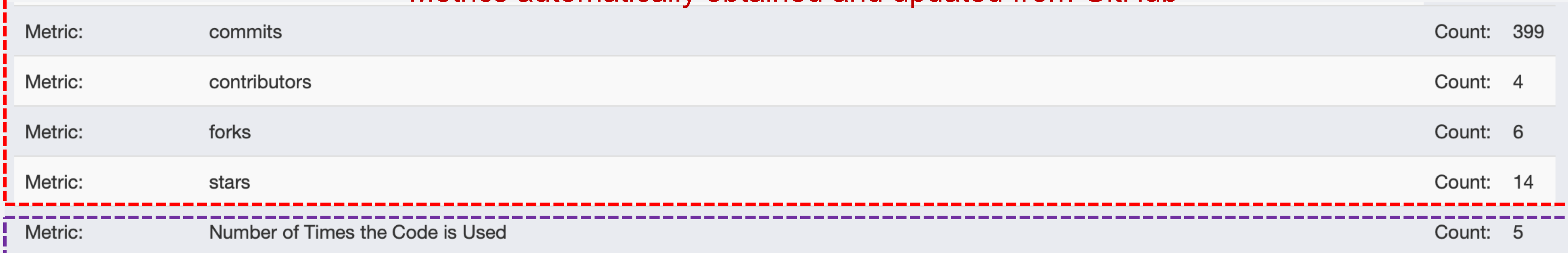
- Product owners can specify the metrics of their choice and the source for gathering the metric data while cataloging their products

- The metrics specified by the product owners is displayed on the product detail pages for the cataloged products

Metrics

Snapshot of Details of a Cataloged Product in Opuntia

Metrics automatically obtained and updated from GitHub



Metric:	commits	Count:	399
Metric:	contributors	Count:	4
Metric:	forks	Count:	6
Metric:	stars	Count:	14
Metric:	Number of Times the Code is Used	Count:	5

Custom-defined metric that is updated automatically using iTracker API – shows aggregated data

iTracker includes scripts and APIs using which one can specify the metrics of their choice, how to get the data associated with the metrics, and where from

On the basis of the information provided by users, iTracker semi-automatically updates the product pages in Opuntia with the metric information at a pre-defined frequency

If users provide the links to their GitHub repositories, the standard metrics such as # of commits, # of contributors, # of forks, and # of stars are also collected

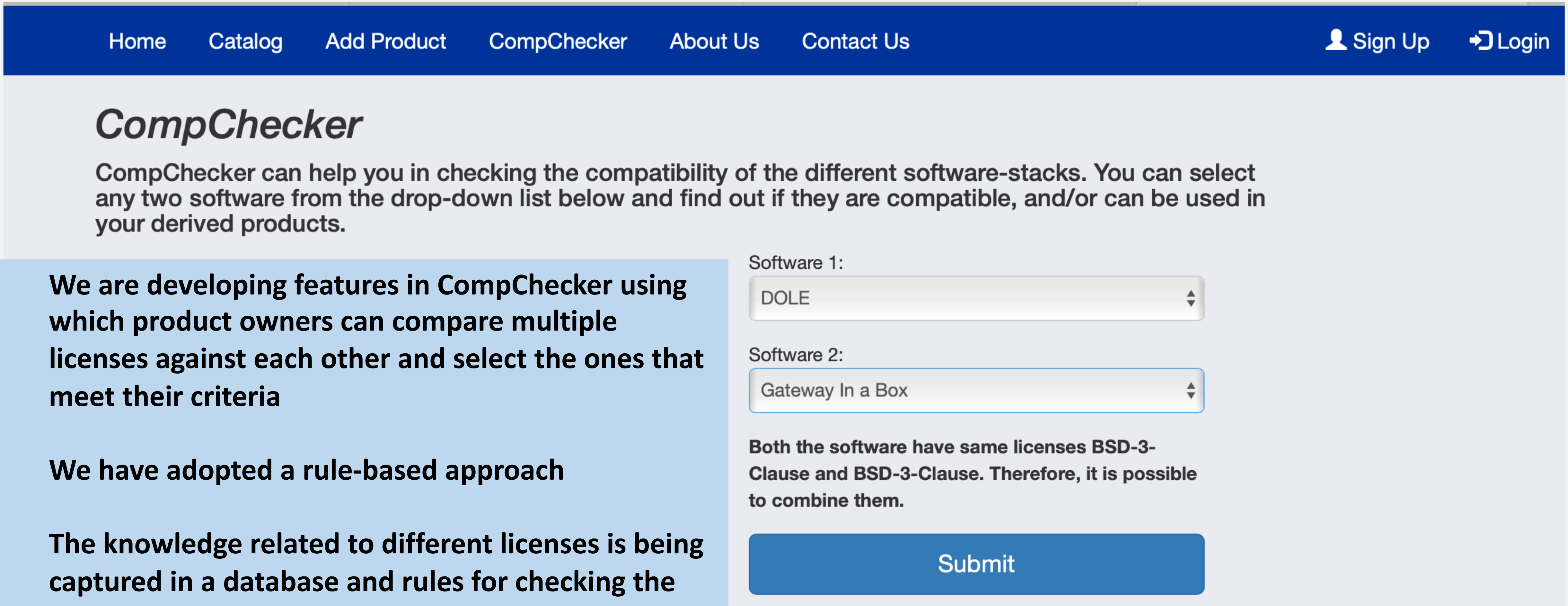
iTracker: how is it being implemented?

iTracker also includes libraries for supporting automatic tracking of product-use for those products that are written in Java and C++ and run in different environments

iTracker can also automatically parse Google Analytics data and update the Catalog if the required files are provided

iTracker can gather metrics from publicly accessible pages such as GitHub and Digital Rocks portal

We have developed Google Analytics like functionality for products that cannot use Google Analytics but need functionality similar to it (for both web-based and non-web applications)



CompChecker

CompChecker can help you in checking the compatibility of the different software-stacks. You can select any two software from the drop-down list below and find out if they are compatible, and/or can be used in your derived products.

Software 1: DOLE

Software 2: Gateway In a Box

Both the software have same licenses BSD-3-Clause and BSD-3-Clause. Therefore, it is possible to combine them.

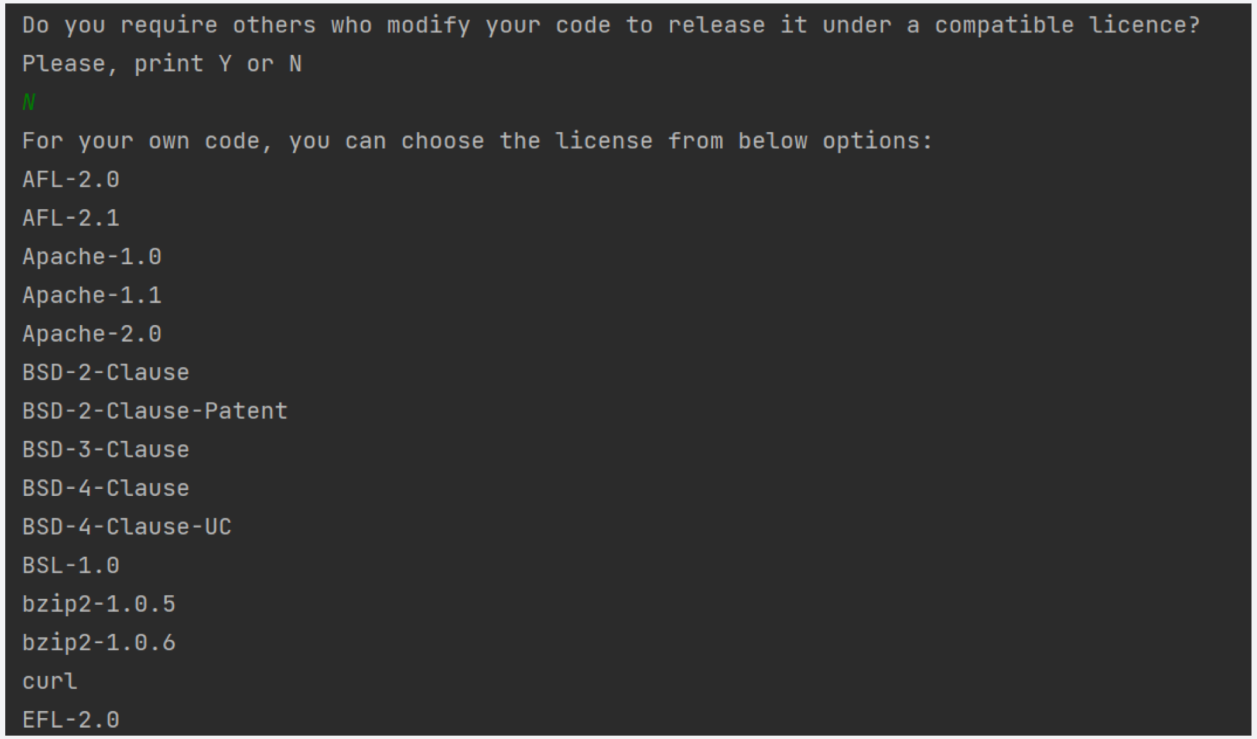
We are developing features in CompChecker using which product owners can compare multiple licenses against each other and select the ones that meet their criteria

We have adopted a rule-based approach

The knowledge related to different licenses is being captured in a database and rules for checking the compatibility of the licenses is implemented

Next, software stack compatibility will be supported

CompChecker: for Checking the License and Software-Stack Compatibility of the Cataloged Products



```
Do you require others who modify your code to release it under a compatible licence?
Please, print Y or N
N
For your own code, you can choose the license from below options:
AFL-2.0
AFL-2.1
Apache-1.0
Apache-1.1
Apache-2.0
BSD-2-Clause
BSD-2-Clause-Patent
BSD-3-Clause
BSD-4-Clause
BSD-4-Clause-UC
BSL-1.0
bzip2-1.0.5
bzip2-1.0.6
curl
EFL-2.0
```

Snippet of the CompChecker decision support system in action

As the knowledge related to different licenses is already being captured in a database, we are able to extend CompChecker for suggesting appropriate licenses for different categories of software and data products

There are more than 61 licenses that can be used for open-source products – our tool can help you in narrowing down on the licenses that could best serve you and your user base

Additional CompChecker Feature: Suggesting Licenses for Products

Questions, Comments, or Concerns?

Thank You!

Ritu Arora
Email: ritu.arora@utsa.edu

UTSA®

utsa.edu