CONCURRENCY AND COMPUTATION: PRACTICE AND EXPERIENCE
Concurrency Computat.: Pract. Exper. 2015; **27**:4412–4422
Published online 19 May 2015 in Wiley Online Library (wileyonlinelibrary.com). DOI: 10.1002/cpe.3542

SPECIAL ISSUE PAPER

Araport: an application platform for data discovery

Matthew R. Hanlon[1,*,†], Matthew Vaughn[1], Stephen Mock[1], Rion Dooley[1], Walter Moreira[1], Joe Stubbs[1], Chris Town[2], Jason Miller[2], Vivek Krishnakumar[2], Erik Ferlanti[2] and Eleanor Pence[2]

[1]*Texas Advanced Computing Center, The University of Texas at Austin, Austin,TX, USA*
[2]*J. Craig Venter Institute, Rockville, MD, USA*

SUMMARY

Araport is an open-source, online community resource for research on the *Arabidopsis thaliana* genome and related data. Araport is developed through a partnership between J. Craig Venter Institute, the Texas Advanced Computing Center at The University of Texas at Austin, and The University of Cambridge. Part of the open architecture of Araport is the Science Applications Workspace. Taking an 'app store' approach, users can choose applications developed both by the Araport team and community developers to create a customized environment for their work. Araport also provides tooling and support for developing applications for Araport, including an application generator, a rapid development and testing tool, and a straightforward deployment path for publishing applications into the Araport workspace. Copyright © 2015 John Wiley & Sons, Ltd.

Received 20 January 2015; Revised 24 March 2015; Accepted 14 April 2015

KEY WORDS: Arabidopsis; bioinformatics; APIs; app store; web applications

1. INTRODUCTION

Araport is an open-source, online community resource for research on the *Arabidopsis thaliana* genome and related data, developed through a partnership between J. Craig Venter Institute (JVCI), the Texas Advanced Computing Center (TACC) at The University of Texas at Austin, and The University of Cambridge. *A. thaliana* is a model organism in plant science and was the first plant genome to be sequenced. Araport is defining a new model for what a data portal can provide to the community. Traditionally, data portals have hosted curated data sets, databases, and tools for researchers to search and refine, and download data to be analyzed using their own software tools and workflows. Many portals also provide tools for common analyses and visualization. The novel aspect of Araport is that it is designed to be the pinnacle of a data federation ecosystem. The portal is completely open source[‡] and community extensible. In addition to the software tools and data provided by Araport, users are able to bring their own data and develop their own analysis tools to build a customized research environment.

A cornerstone feature of Araport, and the subject of this paper, is the science applications workspace. Users can create multiple workspaces where they can choose from a catalog of 'science apps' to customize their research environment. These science applications are developed both by the Araport team and community developers. Araport provides tooling and support for developing science applications, including an application generator, a rapid development and testing tool, and a straightforward deployment path for publishing an application into the Araport workspace.

*Correspondence to: Matthew R. Hanlon, Texas Advanced Computing Center, The University of Texas at Austin, Austin, TX, USA.
[†]E-mail: mrhanlon@tacc.utexas.edu
[‡]Find all of Araport's source code at https://github.com/Arabidopsis-Information-Portal/.

2. BACKGROUND

In 2012, a call was made by the International Arabidopsis Informatics Consortium [1] for a new resource to provide access to community data and outputs in a single interface. This resource was to provide core functionality while also being community extensible to encourage constant innovation from a wide range of contributors. Araport has been built to answer that call.

Araport builds upon work by the iPlant Collaborative [2], an National Science Foundation (NSF) funded cyber-infrastructure project targeting life sciences research. Much of what iPlant has done to democratize access to high-performance computing resources has helped to pave the way for Araport. One iPlant project, the Agave Platform, is critical to the federated architecture of Araport. How Araport is using the Agave Platform is discussed in Section 3. In addition to technology, Araport leverages iPlant's cyber-infrastructure directly using, for example, the iPlant Data Store to host publicly downloadable datasets.

There are existing frameworks that enable the development of individual applications that are deployed to a web application container. Portlets and portlet containers (Java Specification Request (JSR) 168 [3] and 286 [4]) are a well-defined framework for deploying modular functionality. Another example is the OpenSocial specification, originally developed by Google and MySpace and later becoming Apache Shindig [5], also provides a means for hosting 'gadgets' of various trust levels. However, both portlets and gadgets have specific server-side application requirements, limiting the deployment options for these applications to environments that implement the specifications. In contrast, Araport science applications are designed specifically to be framework-agnostic and can be embedded into any existing application without the need for additional server development. Araport apps conventions are designed specifically so that apps can be embedded in third-party applications, allowing contributors to write code that works for their own purposes while at the same time extending Araport. Araport science applications are written in JavaScript and are fully client-side. The 'server' component of these applications comes from the Agave API or other web services, such as those integrated into Araport's API space using the Araport Data Mediator API (ADAMA). These applications can be hosted on almost any platform that can serve HyperText Markup Language (HTML) to client browsers, and can even be run as standalone web applications with little to no modification.

Araport is focused on facilitating the discovery, query, and visualization of structured data sets, many of which are hosted outside of the immediate Araport platform, and are made accessible via Araport's federated architecture. Other web-based science platforms, such as Galaxy [6, 7], Taverna [8, 9], and the iPlant Discovery Environment [2, 10], provide powerful and extensible workflow systems, but are largely focused around the orchestration of computation. Data sharing features are focused on object-level sharing of files. These workflow systems are appropriate downstream targets of information gathering conducted at Araport. Consequently, integration with and the ability to export data to such systems is explicitly on the project road map.

3. ARCHITECTURE AND TECHNOLOGY

The Alpha release of Araport includes several components that are loosely coupled together via web service APIs and OAuth2 [11] (Figure 1). Even the core components of Araport conform to the model for federation and interoperability. This allows the core of Araport to also serve as an example to third party applications and data providers interested in integration with Araport. It has the additional advantage of allowing each component to operate on an independent development and release cycle. Araport is distributed across TACC and JCVI infrastructures. The primary Araport web portal[§] is hosted at TACC and is built on the Drupal content management system [12]. Drupal was selected as a the web framework due to its wide community support, extensibility, and the development team's familiarity with the framework. JCVI hosts Thalemine, the Araport deployment of InterMine [13]. InterMine is an open source data warehouse platform built specifically for hosting

[§]https://www.araport.org.

DOI: 10.1002/cpe

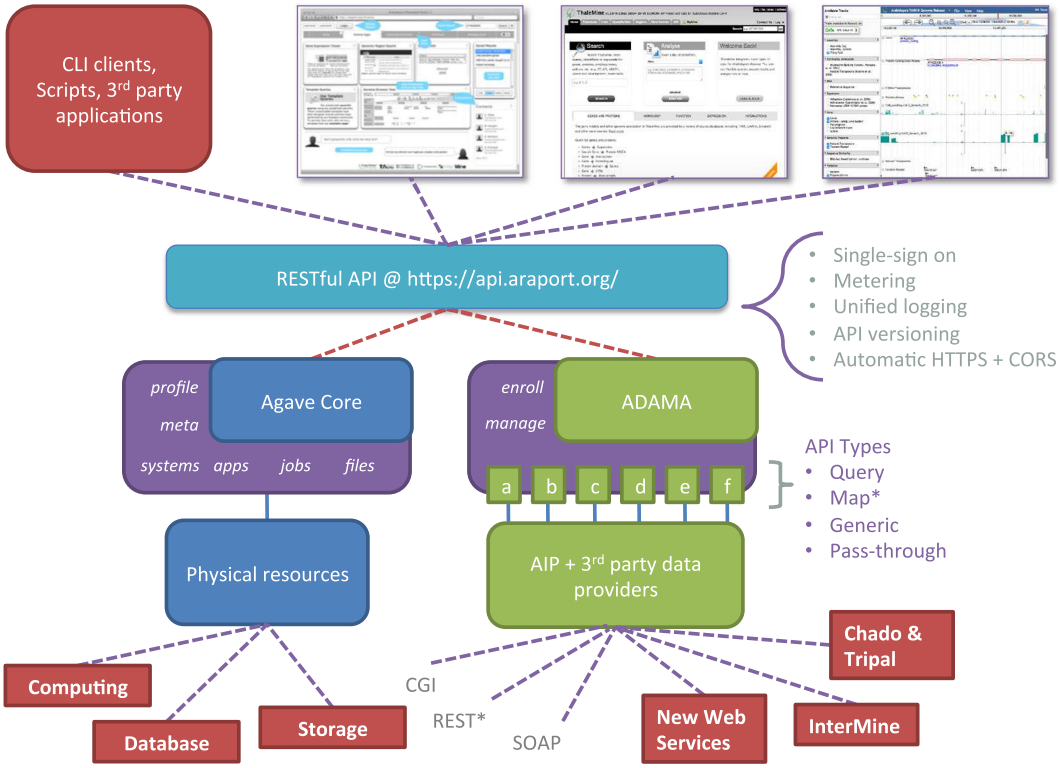

Figure 1. The Araport architecture. The top row represents Araport user applications, including, from the left, Araport Science Apps and other third party applications, the Araport portal, Thalemine, and JBrowse. All user applications communicate with the rest of the Araport architecture using a single, RESTful API.

complex biological data. InterMine has a web interface for searching and exploring the data mines as well as powerful APIs that enable remote applications and services to interact with the data mine. The InterMine data warehouse allows Araport to host both curated datasets as well as third-party datasets in the case where a particular third party did not want to or was not able to host that dataset remotely. Also at JCVI are deployments of thick-client applications including JBrowse [14] and GBrowse [15]. Araport also uses the iPlant Data Store, a hosted data service provided by iPlant, for storing datasets. These datasets are made available for download on the Araport web portal.

In order to choreograph these services, Araport has adopted the Agave Platform. Agave [16] is 'Science-as-a-Service' platform that provides the plumbing for constructing a virtual cyber-infrastructure on top of existing resources. It is a hosted, multi-tenant system, allowing platform developers to quickly bootstrap using Agave without the need to install or run any software. Agave provides a complete middleware with identity management, authentication, and authorization using OAuth2, APIs for interacting with compute and storage systems, application management, job execution, data lifecycle management, monitoring, and notifications. Agave also provides an API mediation layer for extending the platform to add additional capabilities. It was originally developed by iPlant and is maintained and extended by TACC.

The Araport team worked with the Agave team to develop a set of Gateway DNA components [17–19] that allow native integration of Agave within the Drupal framework. The Gateway DNA Drupal modules currently used in the Araport portal provide native Drupal interaction with Agave Authentication, Profiles, Systems, and Files APIs. Araport also worked with both the Agave team and the InterMine team build support for Agave and OAuth2 identity management into InterMine. When users create an account and authenticate at Araport their identity is stored within the Agave identity management system. Whether a user is accessing the primary web portal, Thalemine, or any future application or service, they can use their same Araport credentials. This provides a consistent user experience across all Araport services and applications, from the command line to the web.

DOI: 10.1002/cpe

4. SCIENCE APPLICATIONS

For real scientific discovery and exploration to occur, researchers need to have more than just access to data. They need to have tools to which they can interact with the data: (1) create mashups combining multiple data sources to discover hot spots and other points of interest; (2) analyze their own data against curated, reference data sets; and (3) visualize interactions within the data.

Some early examples of applications developed by the Araport team are the Bio-Analytic Resource for Plant Biology (BAR) Interaction Viewer, the BAR Expression Viewer, and the European Bioinformatics Institute (EBI) Interaction Viewer. These applications use APIs mediated through Agave from the BAR [20] and the EBI [21]. The first two applications are the earliest applications deployed on the platform. The BAR Interaction Viewer allows users to query a gene by name or Arabidopsis Gene Identifier (AGI) and returns an interactive visualization of its protein–protein interactions. The BAR Expression Viewer takes the same query term and returns an image visualizing the gene's expression on an Arabidopsis Developmental Map.

The EBI Interaction Viewer application is an example of an early success using the Araport Science App generator (discussed in the succeeding paragraphs). The EBI web services were identified as providing additional data that could be used in an interaction viewer similar to the BAR Interaction Viewer. An intern working at JCVI was able to fork the BAR Interaction Viewer code and wrap that code in the generated Araport app. Using the EBI web services, she added additional features that showed, in addition to the protein interactions, the type of interaction, confidence score values, and sourcing information for supporting literature.

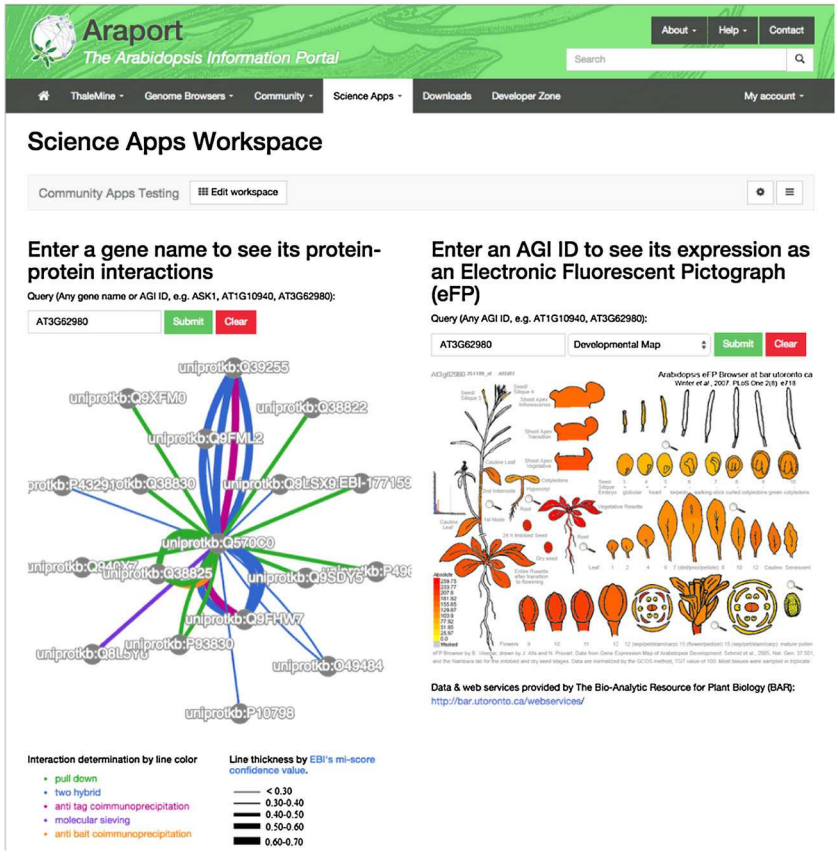

Figure 2. A view of the developer release of the science apps workspace showing two applications: the EBI Interaction Viewer and the BAR Expression Viewer. EBI, European Bioinformatics Institute; BAR, The Bio-analytic Resource for Plant Biology.

DOI: 10.1002/cpe

The Araport team has since developed several more applications, both to serve as examples or starting points for community developers and to provide some of the basic functionality needed in the workspace on day one. These apps range from computational apps such as BLAST, to query apps for querying ATTED-II or RT-qPCR expression data from remote/legacy data services, to more general-purpose productivity apps such as a notebook application.

Figure 2 shows an example of how the science apps workspace appears on the Araport Developer Portal at the time of this writing. The workspace will be moving into the main Araport Portal in the next major release. Users can create and switch between multiple workspace views. Each workspace is configured with apps arranged in rows and columns, with up to four applications in each row.

Araport science applications are client-side web applications written in JavaScript. They are primarily lightweight tools that provide querying, display, and browsing of data, as well as some visualization and analysis. These applications leverage RESTful APIs that are part of the Agave platform, third party data mediated through Agave or the ADAMA, or third party APIs that support cross-origin resource sharing (CORS). If heavier processing than is feasible in JavaScript is required, computational apps are possible using either the Agave API or other APIs to offload processing to a computational backend. The Araport BLAST application is an example of this, using Agave to run BLAST jobs on a remote compute resource and then retrieve the results back to the user's web browser.

No single team can build a set of tools or analyses that will solve the needs of all users. To this end, Araport provides tools and support for data scientists and bioinformaticians to develop a novel, data-driven applications and deploy them to the Araport portal to use and eventually share with other users. To support the development of these applications, Araport has developed a 'science app generator' using Yeoman [22].

Yeoman is a tool for quickly scaffolding projects and helping those projects adhere to best practices, coding standards, and accepted libraries. There are a other similar tools such as Lineman [23] and Brunch [24]. These tools are built for rapid prototyping, interactive development, and encouraging best practices and coding standards. Yeoman and Lineman are both built on top of the Grunt [25] task runner. Yeoman and Brunch both use Bower [26] for web library package management. In the end, previous experience with Yeoman led the Araport team to select it for use.

Node.js [27] is a prerequisite to using the Yeoman generator. The first time you use the generator, you will need to install Yeoman, Bower, and Grunt, as well as the Araport science app generator [28] using the Node Package Manager. This can be done with a simple, one-line command. Once installed, kickstarting a new Araport science application is as simple as issuing the command: `yo aip-science-app`.

The application generator scaffolds a new application within the current directory. The included 'test runner' application contains a built-in, web server, allowing development on the local machine without the need to install or configure a web server. When running within the test runner, the app is accessible in a browser at the address `http://localhost:9000`. The application's source is contained within the `app/` directory. The rest of the files created by the app generator are related to the test runner environment.

The test runner environment is a mini-web application, consistent with the environment of the Araport workspace. Both jQuery [29] and Bootstrap [30] are provided. The Araport theme is derived from Bootstrap, and by simply following the Bootstrap user interface guidelines, developers can build applications that are consistent with Araport and other science apps. The test runner also configures the Agave API with ADAMA, so that developers can build against those APIs without needing to configure them in their application. Simply create an API key using your Araport credentials and begin using the API. The test runner will also persist the API credentials (API key and token; not the developer's Araport username/password) in the browser's local storage for future use. When apps are deployed to the Araport workspace, the current user's authentication token is used to make API queries on behalf of that user.

The test runner environment also includes additional features to improve developer productivity. The environment incorporates JSHint [31] to watch the application source for common coding mistakes such as syntax errors and misspellings, as well as encourage coding best practices. The developer is alerted in real time to the areas of concern in their code so they can double check

DOI: 10.1002/cpe

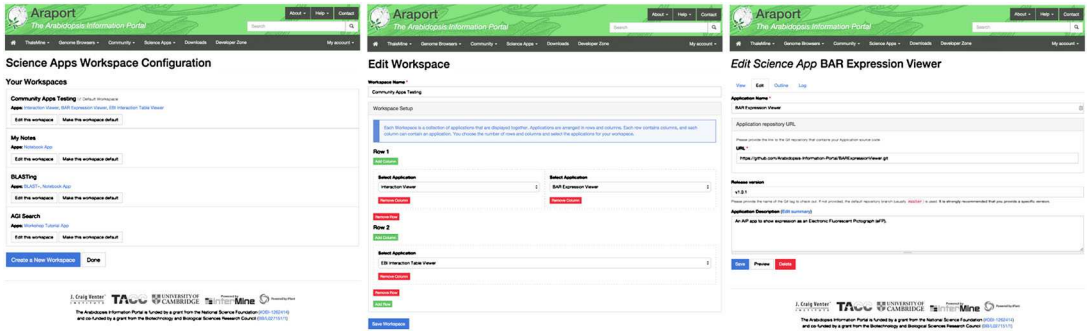

Figure 3. The Science Apps Workspace allows users to create and manage multiple customize workspaces, and also allows app developer to load their own applications into the Araport environment.

for mistakes without having to wonder why something is not working. The test runner also will automatically trigger the browser to reload when a source file changes, immediately displaying the change in the browser.

4.1. The workspace

Araport science apps can be run as standalone applications or integrated into other environments, but the primary interface for using these apps is on the Araport portal, through the Science Apps Workspace.

The first version of the science apps workspace, available in the Alpha release of Araport, was based on the Homebox module [32] for Drupal. This module builds upon the Blocks system in Drupal core and allows administrators to configure pages that users can subsequently customize by enabling and arranging instances of Drupal blocks. By leveraging Homebox and Blocks, Araport was able to rapidly deploy early versions of several science apps without having to develop the functionality for managing user configurations and rendering applications. However, it was quickly apparent that Homebox would not suffice for much beyond a rapid prototype, as it had several shortcomings that made it unable to fulfill the requirements of the Araport workspace. These shortcomings include in the following: (1) blocks can only be created by administrators; (2) access permissions to blocks are coarse; (3) the layout of a Homebox page (the number of columns and rows) is fixed in the page configuration by an administrator; and (4) users can only customize a single Homebox layout per page.

The Alpha Prime release of the workspace is a completely redesigned, custom Drupal module specifically designed to overcome these shortcomings as well as add additional functionality. This module is build upon Drupal's Node system, which brings with it much finer, customizable access permissions, automatic revision history, a publishing workflow, and more.

The application workspace allows users to manage multiple workspaces for different tasks, customize each workspace with the apps and arrangement as desired, and easily add their own apps to the Araport apps catalog (Figure 3).

4.2. Araport science app lifecycle

The lifecycle of an Araport science app is fairly typical: development, local testing, production testing or staging, quality assurance, and publication. After publication, apps can be further developed, re-tested, and redeployed with an updated version. Once an app has been developed using the app generator and test runner application, before a developer can deploy that app to Araport, the developer must commit the application to a public git repository that supports anonymous cloning over Hypertext Transfer Protocol (HTTP), such as Github or Bitbucket.[¶] This is in concert with Araport's commitment to open source.

[¶]Github and Bitbucket are not the only solutions for public git services. There are many options available including both cloud-hosted and self-hosted solutions.

DOI: 10.1002/cpe

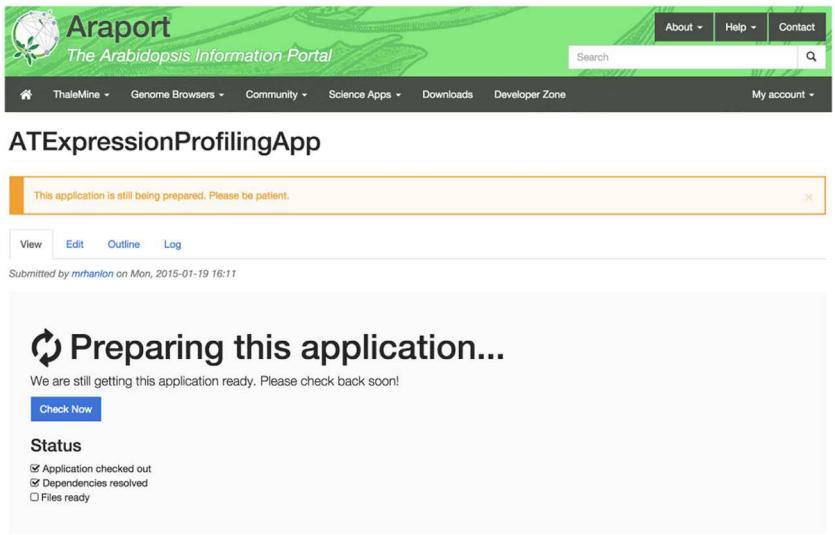

Figure 4. Creating a new science app on Araport. The Araport portal must download the apps source and then resolve its runtime dependencies before the application can be used.

After committing the application to a git repository, from the Araport workspace, users can select 'Create a New App' from the workspace toolbar. This form (Figure 3, right) prompts the user for a name for the app, the URL to the application's git repository, and an optional release version and description. After saving, the user will be taken to the app's dedicated page and will initially see a progress page indicating that the application is still being processed (Figure 4). In the background, the Araport portal will download the app's source for the specified version, defaulting to the latest revision in the `master` branch, and resolve any third-party dependencies before the app can be used.

Once the app has been processed, it is available to use in the Araport workspace; however, the application is restricted to the creator's own sandbox environment. Only the user who created the app will be able to access it in Araport. Application sharing is a future development item. In the interim, there is a workaround that will enable application sharing: if a developer wants to share an app with another Araport user, that user can create their own version of the application within their own sandbox environment simply by creating a 'new' app with the same source repository.

Users will notice that when configuring their workspaces they have available to them two sets of applications: User applications and Public applications. User applications are those applications which reside in the user's own sandbox. Public applications are applications that the Araport team has reviewed and published for use by all Araport users. After application sharing is implemented, a third set for 'Shared applications' will be available. A formalized publication pipeline within the Araport portal is also being developed, but in the meantime users can contact Araport and request that an application be published. For publishing, Araport staff will review the application for coding errors, to ensure that the app will behave nicely with other applications in the workspace, and to check for any possible security vulnerabilities or misuse. After approval, published apps are available for use by any user.

4.3. Application security

Drupal provides support for two filesystem classes: public and private. The public filesystem is web-accessible, that is, assets and media in the public filesystem are directly accessible via some uniform resource identifier (URI). The private filesystem is not web-accessible, and assets stored in the private filesystem cannot be accessed via a URI. Instead, they pass though the application layer for logical processing to ensure, for example, access permissions. As mentioned earlier, applications created by users in the Araport workspace are available for use only by the users who create them. To ensure this, when the portal downloads an app's source, it does so to a private filesystem location

DOI: 10.1002/cpe

unique to that app. When a user that has permission to access that app navigates to a workspace page that hosts it, the app's assets are added to the page. However, they are not directly accessible via a URI. This prevents, for example, a nefarious user creating an application that dumps a malicious payload into the Araport web space, and subsequently attempts to infect other sites using Araport URIs in a cross-site scripting attack.

Furthermore, the private filesystem ensures that the assets downloaded with an app are not executable on the server, are partitioned from both the portal's own code and filesystem as well as partitioned from the source code of other apps. Araport science apps only execute in the browser of the user, and as such, apps from 'unknown' users cannot be executed by anyone other than that same user.

Finally, all API actions taken by an app execute within the context of the current user. The API token used by an app grants only the permissions that that user already has. This prevents the development of an app that allows a user to access protected data or information belonging to another user.

4.4. Data APIs in Araport: Araport data mediator API

Araport science apps are only useful if there are data available to use within those apps. While there are many data services available, much of the data of interest is accessible only via legacy web applications that may or may not offer web service API. Even if a web service API is available, it almost certainly does not conform to a uniform data standard. In order to provide a solution to this, Araport has developed the ADAMA. ADAMA works alongside of Agave to allow users to quickly modernize legacy data services into an HTTP web service that supports Secure Sockets Layer, standard HTTP verbs (notably, GET), and supports protocols such as CORS for interoperability. It also provides an easy and direct path for creating new data services, much like the science app generator. Finally, ADAMA serves as a collection point for data services, allowing these services to be discoverable through an Araport 'data store' much like the app store.

ADAMA provides four API types: query, map, generic, and passthrough. The primary API type encouraged by Araport is the query API. The query API serves as a 'gold standard' for data APIs in Araport, as these APIs must have an input parameter space and output specification that is consistent for all query APIs. Query APIs output JSON responses and can be implemented for any legacy data source. Map APIs are useful when the existing legacy service already serves JSON, but when it is useful to transform that JSON into, for example, JSON according to Araport object specifications. The generic API allows the return of non-JSON data, for example binary image data. The final API type, passthrough, simply allows existing APIs to become discoverable through ADAMA, but inputs and outputs remain under the control of the remote service.

By collecting data services through ADAMA, Araport is hoping to create a data store alongside the app store for science applications. This will serve both as a resource for the community, but also help to give existing data services and data providers additional visibility. ADAMA development is continuing, and additional discussion of it in detail can be expected in future papers by the authors. Current documentation for ADAMA can be found in the Araport Developer Zone.

5. ADOPTION AND USAGE

The first launch of the Araport portal that was open to user registration was made available on 1 April 2014. At that time, the science apps workspace and platform was still under development. Six months later, the first version of the apps workspace and platform was released and user registration has doubled since then. Araport is currently seeing over 2 000 unique visitors per month with about 50% of those being returning visitors. Much of the data currently hosted by Araport is available through Thalemine and JBrowse without registration. The newest features, such as the Science Apps Workspace, require registration. Since the launch of the workspace at the end of 2014, we have seen our registered user count double. In addition, Araport has over 30 registered app developers.

DOI: 10.1002/cpe

Currently, Araport hosts the TAIR10 annotated genome release, which is makes available for download. The next annotated release, AIP11, is in progress. In addition to TAIR10, there are many other data sets that are integrated into the Thalemine data warehouse. The currently available data sets are listed on the Thalemine Data Sources page.[‖]

6. FUTURE WORK

The Araport project is the result of a 2-year planning grant that solicited requirements from the entire Arabidopsis research community. The portal, as it exists at the time of this writing, is technically a prototype and still in an active development phase with a regular release cycle. Despite its prototype status, Araport is already engaging with users and developers within the Arabidopsis community. In November of 2014, Araport held a developer workshop at the TACC. At this workshop, attendees were introduced to Araport apps, the workspace, and ADAMA. Attendees participated in guided sessions, in which they developed a science application from scratch and deployed it onto the Araport developer portal. Optional breakout sessions were held, where attendees were able to get familiar with ADAMA as well as collaborate in small groups to design and build applications of their own. Much experience was gleaned from that workshop, both by attendees and the Araport team. Many new development items were added to the roadmap taking into account the experiences of the workshop attendees and their feedback. While a great portion of this included hardening of the environment, better error reporting and logging available to users, and improved documentation, one item in particular stands out for discussion: the development of a formalized science app manifest.

6.1. Science apps manifest

The currently supported science apps have an implicit application manifest, which is the files created by the app generator in the `app/` directory. Prior to the workshop, the Araport team had discussed the need for apps to be able to host additional assets beyond these, such as images. However, the workshop showed that in reality, apps in general need a more flexible hosting structure that not only allows for images, but multiple HTML, JavaScript, and CSS files, as well as support for other media types. For example, apps built using the Processing.js framework [33] use a script type `text/processing` and a file extension `*.pde`. The science apps themselves and the apps workspace are framework agnostic, and the current generation of Araport apps and the workspace can work frameworks like Processing. However, to do so, developers are made to construct apps out of composable libraries managed with Bower. This was designed to keep the apps themselves lightweight and encourage reuse, but early developer feedback has shown than this is not how the community would like it to work. The science apps manifest should be flexible enough to accommodate more flexible asset loading for frameworks like Processing or any other of the every growing catalog of client-side frameworks.

6.2. Attribution in apps and services

Another major area that is actively being designed centers around the topic of attribution. The developers of applications know the source of the data provided by their ADAMA APIs, and used by their science apps. However, for published apps, unless the developer explicitly sources the data and provides proper attribution, there is little to no chance that the user will know the data's source or that the data provider will receive acknowledgment. It is obvious that this is unacceptable, even in the short run. While the Araport team has made an effort to establish best practices for data attribution in apps, it will be necessary to ensure that sourcing information is contained in every API response and, ideally, automatically included in every science app and workspace page, to ensure that data providers receive proper attribution.

[‖]https://apps.araport.org/thalemine/dataCategories.do.

DOI: 10.1002/cpe

ACKNOWLEDGEMENTS

Araport is funded by a grant from the National Science Foundation (DBI-1262414) and co-funded by a grant from the Biotechnology and Biological Sciences Research Council (BB/L027151/1). The Araport team would also like to acknowledge the iPlant Collaborative for the use of the Agave Platform and the iPlant Data Store.

REFERENCES

1. The International A I C. Taking the next step: building an Arabidopsis information portal. *The Plant Cell Online* 2012; **24**(6):2248–2256.
2. Goff SA, Vaughn M, McKay S, Lyons E, Stapleton AE, Gessler D, Matasci N, Wang L, Hanlon M, Lenards A, Muir A, Merchant N, Lowry S, Mock S, Helmke M, Kubach A, Narro M, Hopkins N, Micklos D, Hilgert U, Gonzales M, Jordan C, Skidmore E, Dooley R, Cazes J, McLay R, Lu Z, Pasternak S, Koesterke L, Piel WH, Grene R, Noutsos C, Gendler K, Feng X, Tang C, Lent M, Kim Seung-jin, Kvilekval K, Manjunath BS, Tannen V, Stamatakis A, Sanderson M, Welch SM, Cranston K, Soltis P, Soltis D, O'Meara B, Ane C, Brutnell T, Kleibenstein DJ, White JW, Leebens-Mack J, Donoghue MJ, Spalding EP, Vision TJ, Myers CR, Lowenthal D, Enquist BJ, Boyle B, Akoglu A, Andrews G, Ram S, Ware D, Stein L, Stanzione D. The iPlant collaborative: cyberinfrastructure for plant biology. *Frontiers in Plant Science* 2011; **2**(34):1–16.
3. The Java Community Process(SM) Program - JSRs | Java specification requests - detail JSR#168.html. (Available from: https://www.jcp.org/en/jsr/detail?id=168) [Accessed on August 26, 2014].
4. The Java Community Process(SM) Program - JSRs | Java specification requests - detail JSR#286.html. (Available from: https://www.jcp.org/en/jsr/detail?id=286) [Accessed on August 26, 2014].
5. Shindig - welcome to apache Shindig.
6. Goecks J, Nekrutenko A, Taylor J, Team The Galaxy. Galaxy: a comprehensive approach for supporting accessible, reproducible, and transparent computational research in the life sciences. *Genome Biology* 2010; **11**(8):1–13.
7. Galaxy. (Available from: https://usegalaxy.org/) [Accessed on March 23, 2015].
8. Wolstencroft K, Haines R, Fellows D, Williams A, Withers D, Owen S, Soiland-Reyes S, Dunlop I, Nenadic A, Fisher P, Bhagat J, Belhajjame K, Bacall F, Hardisty A, Nieva de la Hidalga A, Balcazar Vargas MP, Sufi S, Goble C. The taverna workflow suite: designing and executing workflows of web services on the desktop, web or in the cloud. *Nucleic Acids Research* 2013; **41**(W1):W557–W561.
9. *Taverna - open source and domain independent Workflow Management System*. (Available from: http://www.taverna.org.uk) [Accessed on March 23 2015].
10. iPlant collaborative discovery environment. (Available from: https://de.iplantcollaborative.org/de/) [Accessed on 23 March 2015].
11. OAuth 2 specification. (Available from: http://oauth.net/2) [Accessed on 29 March 2013].
12. Drupal - open source CMS | Drupal.org. (Available from: https://www.drupal.org) [Accessed on August 26 2014].
13. Smith R N S, Aleksic J, Butano D, Carr A, Contrino S, Hu F, Lyne M, Lyne R, Kalderimis A, Rutherford K, Stepan R, Sullivan J, Wakeling M, Watkins X, Micklem G. Intermine: a flexible data warehouse system for the integration and analysis of heterogeneous biological data. *Bioinformatics* 2012; **28**:3163–3165.
14. JBrowse | A fast, embeddable genome browser built with HTML5 and JavaScript. (Available from: http://jbrowse.org/) [Accessed August 26 2014].
15. GBrowse - GMOD. (Available from: http://gmod.org/wiki/GBrowse) [Accessed on August 26 2014].
16. Dooley R, Vaughn M, Stanzione D, Terry S, Skidmore E. Software-as-a-service: the iPlant foundation API. *5th IEEE Workshop on Many-Ttask Computing on grids and Supercomputers*, Salt Lake City, UT, 2012.
17. Dooley R, Hanlon M R. Recipes 2.0: building for today and tomorrow. *Concurrency and Computation: Practice and Experience* 2014. DOI: 10.1002/cpe.3285.
18. deardooley / agave/gateway-dna – Bitbucket. (Available from: https://bitbucket.org/deardooley/agave-gateway-dna) [Accessed on August 26 2014].
19. mrhanlon / gateway-dna-drupal – Bitbucket. (Available from: https://bitbucket.org/mrhanlon/gateway-dna-drupal) [Accessed on 26 August 2014].
20. The BAR Webservices. (Available from: http://bar.utoronto.ca/webservices/) [Accessed on 26 August 2014].
21. *www.ebi.ac.ul/intact/*. (Available from: http://www.ebi.ac.uk/intact/) [Accessed on 26 August 2014].
22. The web's scaffolding tool for modern webapps | Yeoman. (Available from: http://yeoman.io/) [Accessed on 26 August 2014].
23. Lineman | Build awesome web apps, easily. (Available from: http://linemanjs.com/) [Accessed on 26 August 2014].
24. *Brunch | ultra-fast HTML5 build tool*. (Available from: http://brunch.io/) [Accessed on 62 August 2014].
25. Grunt: the JavaScript task runner. (Available from: http://gruntjs.com/) [Accessed on 26 August 2014].
26. Bower. (Available from: http://bower.io/) [Accessed on 26 August 2014].
27. node.js. (Available from: http://www.nodejs.org/) [Accessed on 26 August 2014].
28. AIP science app generator. (Available from: https://www.npmjs.org/package/generator-aip-science-app) [Accessed on 26 August 2014].

29. jQuery. (Available from: http://jquery.com/) [Accessed on 18 January 2015].
30. Bootstrap - the world's most popular mobile-first and responsive front-end framework. (Available from: http://getbootstrap.com/) [Accessed on 18 January 2015].
31. JSHint, a JavaScript Code Quality Tool. (Available from: http://jshint.com/) [Accessed on 18 January 2015].
32. Homebox | Drupal.org. (Available from: https://www.drupal.org/project/homebox) [Accessed on 26 August 2014].
33. Processing.js. (Available from: http://processingjs.org/) [Accessed on 18 January 2015].

DOI: 10.1002/cpe