Available online at www.sciencedirect.com

ScienceDirect

Procedia Computer Science 4 (2011) 658–667

Procedia Computer Science

International Conference on Computational Science, ICCS 2011

Paper Mâché: Creating Dynamic Reproducible Science

Grant R. Brammer, Ralph W. Crosby, Suzanne J. Matthews, and Tiffani L. Williams

[grb,rwc,sjm,tlw]@cse.tamu.edu Department of Computer Science & Engineering, Texas A&M University, College Station, TX 77843-3112

Abstract

For centuries, the research paper have been the main vehicle for scientific progress. From the paper, readers in the scientific community are expected to extract all the relevant information necessary to reproduce and validate the results presented by the paper's authors. However, the increased use of computer software in science makes reproducing scientific results increasingly difficult. The research paper in its current state is no longer sufficient to fully reproduce, validate, or review a paper's experimental results and conclusions. This impedes scientific progress. To remedy these concerns, we introduce Paper Mâché, a new system for creating dynamic, executable research papers. The key novelty of Paper Mâché is its use of virtual machines, which lets readers and reviewers easily view and interact with a paper, and reproduce key experimental results. For authors, the Paper Mâché workbench provides an easy-to-use interface to build an executable paper. By transforming the static research paper into a dynamic and interactive entity, Paper Mâché brings the presentation of scientific results into the 21st century. We believe that Paper Mâché will become indispensable to the scientific process, and increase the visibility of key findings among members and non-members of the scientific community.

Keywords: executable paper, virtual machines, scientific reproducibility, abstract management, reviewing

1. Introduction

Scientific progress depends on the effective dissemination and reproducibility of existing research. For centuries, scientific papers (as well as scientific books) have been the primary mechanism for disseminating scientific results. However, such a mechanism is based on the reader having access to the materials needed to validate the results discussed in the scientific paper. The increased use of computer software in science makes reproducibility of results quite difficult—especially since many scientists do not publish the source code nor the data needed to reproduce their results. As a result, the hypotheses and results discussed in a scientific paper are not validated since it is too difficult for the reader to recreate the authors' experimental environment. Given that our current dissemination practices impede scientific progress, how can we make scientific contributions more easily accessible (or executable) for the scientific community and public at large?

We introduce Paper Mâché, a novel paper management system under development that allows users to explore research papers interactively, reproduce results and test hypotheses of their own. That is, Paper Mâché supports the notion of an executable paper. Our expertise in high-performance computing and bioinformatics have provided the motivation for developing our system for a wide variety of users. More specifically, Paper Mâché divides the scientific community into three different individuals: authors, reviewers, and readers. For authors, our Paper Mâché system offers a simple interface to build an interactive and executable paper. The *novel* use of virtual machines in

doi:10.1016/j.procs.2011.04.069

Paper Mâché allows authors to easily reconstruct their experimental environment, which in turn, allows reviewers and readers to explore a paper interactively, reproduce and validate experimental results, and test their own hypotheses.

1.1. Existing paper management systems

Current systems for author-reviewer interaction involve the ability for authors to submit static, scientific papers to a server, which are then assigned to reviewers. The reviewers then review the paper by submitting their comments back to the server. This is usually done with the assistance of abstract management software (usually bundled with a conference management system). The purpose of these systems is to lessen the administrative workload of handling the large volumes of submitted scientific abstracts and articles. Within the computer science community, EasyChair [1] is by far the most popular conference management system, due to its ease-of-use and being free. In 2010, there were 3,306 computer science conferences managed using EasyChair [1]. Popular commercial options include START [2] and Linklings [3], the latter which is used by computing conferences such as SuperComputing (SC) and Grace Hopper Celebration of Women in Computing (GHC). Professional organizations for computing such as ACM and IEEE use ScholarOne Manuscripts [4] (formerly Manuscript Central) as their abstract management software. However, the key limitation to these systems is that they only permit the inclusion of the research paper itself. Experimental data and source code are not uploaded to these systems. If an author wishes to have these elements available for reviewers and readers, the author must find a way to host the source code and data during the review process. The reviewer then has to download the source code and then spend time figuring out how to execute the source, which can be a nontrivial process. As a result, the current techniques for managing papers makes it very hard to reproduce the experimental results in the paper, which is key to validating the claims the authors make in their scientific paper.

In addition, the forums in which authors can interact with readers is quite limited. At conference talks, audience members have a limited time span in which to directly interact with the author. In some scientific journals (e.g. Systematic Biology), readers can respond to authors and their work in the form of a "Points of View" section. However, this form of communication between authors and readers is not immediate since it can take months before the point of view appears in the journal—assuming the reviewers reading the point of view feel that the viewpoint is worthy of publication. Web-based systems like CiteULike [5] promote reader interaction within the scientific community through "social bookmarking". Here, users share references to papers they enjoy, and they can also see who likes the same paper they do. However, there does not seem to be a way for these readers to interact directly with the author themselves. Such interaction would be very valuable since authors can use the feedback as a way to improve their work.

1.2. Summary and software availability

We believe that Paper Mâché is indispensable for future scientific research, and provides a mechanism for increasing the visibility and accessibility of scientific findings to everyone. Currently, our proposed system is under development and is a finalist in the *Executable Paper Grand Challenge* sponsored by Elsevier. For more information regarding this challenge, please visit `http://www.executablepapers.com/index.html`. A demonstration of our Paper Mâché system will take place at the International Conference on Computational Science (ICCS'11) in June 2011. Thus, this paper discusses the underlying design of Paper Mâché, considers sample use cases, and discusses how our system will be evaluated, and explores the future capabilities of Paper Mâché.

2. Background: Virtual Machines

The *novelty* of Paper Mâché is the use of virtual machines (VMs) [6] to implement an executable paper that allows authors, reviewers, and readers to interact. A virtual machine is a system that performs software-level emulation of a system different than the host machine. While virtual machines have existed since the 1960s, they have been used in recent years as a mechanism to evaluate new operating systems, act as test environments for software, back-up a system, and run code on virtualized "clones" of legacy machines. A virtual machine consists of two main components: the hypervisor and (one or more) guest operating systems. The hypervisor is installed on the native operating system of the host machine, and is used to run one or more guest operating systems. The guest operating system is stored on the host machine in the form of an logical image, which is a disk file consisting of one or more physical disks that the guest requires to execute. This guest image contains the full installation of the guest operating system and

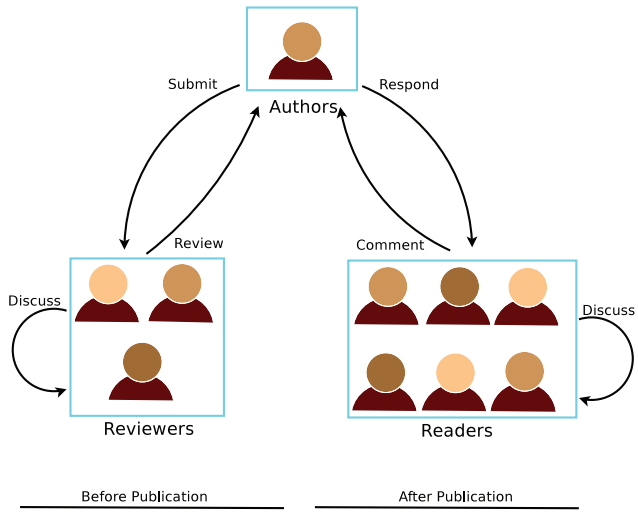

Figure 1: A diagram showing how authors, reviewers, and readers interact in Paper Mâché during the life cycle of a research paper. Before publication, authors submit their paper to the reviewers for review. After discussing the paper amongst themselves, the reviewers submit their reviews to the author. If requested, the authors can then resubmit the revised paper, and receive further revisions from the reviewers. After publication, the research paper is available to the community of readers. These readers can discuss the paper amongst themselves as well as comment on the paper. The Paper Mâché system will facilitate reader comments on a paper and expedite author responses.

software applications. When instantiated by the hypervisor, the guest operating system executes as if it is running directly on the host machine's hardware. In addition to commercial virtual machine software such as VMware [7] and Parallels [8], fully-functioning open source alternatives such as VirtualBox [9] and Xen [10] are available.

For Paper Mâché, the advantages of virtual machines that have allowed for their prolific use are now being applied to improve the quality of research papers and the interactions between authors, reviewers and readers. Virtual machines allow authors to create a snapshot in time of their experimental system, allowing them to easily package their results and data with their research paper in a single entity. Thus, the results of the executable paper can easily be reproduced in the future as a virtualized clone of the original experimental platform. Another added benefit to creating executable papers as virtual machine images is that is easy to enforce an added security levels. For example, during the reviewing process, the VM image of the paper can be locked as "read-only". This allows reviewers to simultaneously view unpublished source code and data, while preventing them from separating unpublished material from the package and co-opting them for personal benefit. Security controls will increase author confidence in the reviewing process, and help prevent plagiarism. Once the paper enters public domain, some of these restrictions may be lifted. Thus, the use of virtual machines in the creation of the executable paper within Paper Mâché simultaneously allows readers and reviewers to interact with a research paper and its experimental results, while protecting the author's sensitive data and source code during the pre-publication phase of the paper.

3. The Paper Mâché System

Paper Mâché is designed to support the requirements of the authors, reviewers and readers that comprise the scientific community. As illustrated in Figure 1, Paper Mâché is intended to support the needs of the full life-cycle of a research paper. Moreover, Paper Mâché does not replace, but in fact augments, the capabilities of scientists working to create new research. Before a paper is published, authors are responsible for creating the paper and submitting the paper to reviewers. At the core of this process is the Paper Mâché package (`.pm`) file. This file is the artifact that represents the "executable paper" and is the container for all the various elements of the paper. For a particular paper, a single `.pm` file is created. Users interact primarily with the Paper Mâché Workbench, which allows the user to create, update, manage and access the `.pm` files. While authors use the Paper Mâché Workbench to create, update and manage the `.pm` files, readers and reviewers use the workbench to view and access the `.pm` files. During the pre-publication phase, reviewers evaluate the paper and communicate with the author. Once the paper has been published, readers

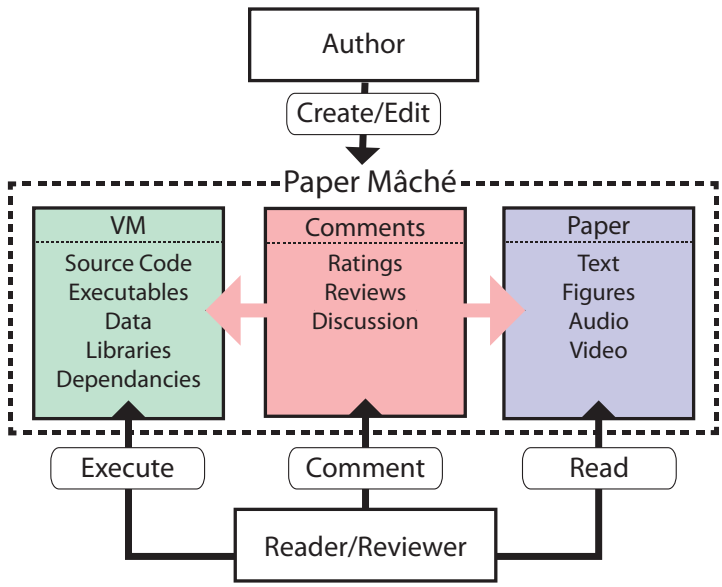

Figure 2: An overview of the Paper Mâché system showing the interaction between the actors and the major components of the system. The author is responsible for creating the paper, virtual machine (VM), and all underlying content. These sections are uploaded through the Paper Mâché workbench, which creates a `.pm` file. Once the components are uploaded, a web based comments section is made available to readers or reviewers. Online users can download and execute the VM, comment on different aspects of the system, and read the paper.

discuss the paper and send their comments to the authors via the Paper Mâché Workbench. The author, in turn, can respond to the comments. These interactions are explored in more detail in Sections 3.1 and 3.2.

As mentioned earlier, Paper Mâché uses virtual machines to replicate the environment in which code and scientific experiments were run. Image files containing virtual machines (`.vm` files) will be packaged within the `.pm` Paper Mâché package. In order to work with Paper Mâché virtual image files, all participants in the process (authors, reviewers and readers) will need to download a Paper Mâché hypervisor appropriate to their environment. Once downloaded, the hypervisor will be usable with any `.vm` files contained in `.pm` Paper Mâché packages. Authors will also need to do a one-time download of the Paper Mâché image tool that will help automate the process of creating virtual images for their research.

3.1. Creating an executable paper or `.pm` file

Here, we describe how an author (or authors) of a scientific paper use Paper Mâché to create an executable version (or `.pm` file) of the paper. We note that Paper Mâché does not replace conventional research leading to publishable scientific results. Prior to working with Paper Mâché, it is assumed that the author(s) of the paper have performed all of the necessary research and written the paper.

First, the primary author logs into the Paper Mâché workbench, creates an empty `.pm` file (the Paper Mâché wrapper in Figure 2) and identifies any additional authors authorized to edit the `.pm` file. Additional metadata (description, keywords, etc.) may be entered at this time or at any future point in the process. At this point, the empty `.pm` will be in an "editable" status indicating that the contents may be freely changed by the authors. To create the contents of the `.pm` file, the authors will upload the text of the paper (e.g., LaTeX or .doc(x) format) and also upload associated files (e.g., audio/video, graphics, figures) in their native formats. This is represented by the Paper section in Figure 2. While there will be no particular order required by the upload process, dependency checking (e.g. the existence of referenced figures in the document) may be requested and will be required prior to review.

To create the virtual machine (`.vm`) files associated with paper, the virtual machine image tool will be run on a test machine (or machines) to create one or more `.vm` files for machines that host any applications referenced in the text.

As part of the virtual image, the authors will define the scripts or commands necessary for a reader to recreate the tests referenced in the text. These instructions will be packaged as metadata associated with the `.vm` file. Authors will be responsible for testing and adjusting the generated `.vm` files using a previously downloaded Paper Mâché hypervisor appropriate to their environment. When completed the `.vm` files will be uploaded into the `.pm` file package creating the VM section shown Figure 2. We note that in Table 2, many of the steps listed under the "Traditional" column will need to be performed by the author. However, the use of virtual machines will eliminate this process for reviewers and readers. Once satisfied with the contents of the `.pm` file, the authors will transition the file to a "submitted" status. In this state, the `.pm` file will be locked preventing updates. Additionally, the file will only be visible to the authors and reviewers, which are assigned by a conference program chair or journal editor. For simplicity, it is best to think of journal editors and conference program chairs as "super reviewers" within Paper Mâché. As a super reviewer, they have the power to assign papers to reviewers as well as make decisions as to whether a paper has been accepted for publication.

Once the reviewing process has ended, authors will receive their reviews as well as the decision whether their paper has been accepted for publication. If changes are required prior to publication, the journal editor or program chair will change the `.pm` file state to allow the authors to make any changes requested. When the package is approved for publication, the editor or program chair will change the status to "published" and the `.pm` file becomes publicly available. In this state, the `.pm` file will be locked for changes. However, the authors will still be able to make changes (e.g., updates to source code) to the `.pm` file. All updates will be tracked separately from the base `.pm` file so that readers will easily be able to view the original contents of the file. Finally, we note that there are unfortunate situations where a published paper has to be retracted or formally corrected. Paper Mâché will be able to change the status of such published papers and make the new status clearly visible to readers.

Once completed, the authoring process will have generated a `.pm` file package containing everything necessary to not only understand the research but duplicate and further experiment with far into the future.

3.2. Reading an executable paper or `.pm` file

The following discussion is focused on the readers of the executable paper, but all operations are equally appropriate to reviewers. The only difference between reviewers and readers is the visibility of the materials. During pre-publication, only reviewers and the paper authors will be able to access the paper. Comments entered by the reviewers (and authors) will only be visible to the authors and reviewers during this period.

A reader starts their interaction with the Paper Mâché Workbench by logging into the web application and searching for a paper of interest. They will be able to read the abstract as part of the search results. If they decide to study the paper further, they will be able to click on the paper and open the `.pm` file in a web page. After opening the `.pm` package, the reader will be able to read the paper as well as view multimedia, figures and other content available (again represented by the Paper section of Figure 2). Hyperlinks help users navigate and view relevant portions of the paper, figures, charts and graphics associated with the package. At any point, the reader will be able to enter comments and ratings for the paper or specific elements of the paper. Comments will be available to the authors in the `.pm` package itself (see Figure 2), and shown on the web page associated with the `.pm` file. Authors will also be able to review and respond to comments from readers through the web page.

Clicking on the `.vm` file name on the web page will initiate the download of the virtual machine image onto the reader's computer. Once downloaded, the `.vm` file is executed on the previously downloaded Paper Mâché hypervisor. The reader is able to sign into the virtual machine and recreate the authors' experiments using the scripts and commands packaged within the `.vm` file. Since the `.vm` file is a fully functioning virtual machine, readers can easily adjust parameters and try running the source with different data.

An example. Figure 3 shows a sample prototype of a Paper Mâché virtual machine executing a published paper describing a MapReduce inspired algorithm called MrsRF [11]. Here, the executable paper (or `Matthews2010.pm`) is executing on a reader's desktop. Within the virtual machine window, the reader has executed an application from the command line and generated a graph. The source code and build files will be packaged within the `.vm` file and the reader is able to view and modify the source code for the application to further experiment with the application. Such changes may be saved in the reader's local copy of the virtual machine but will not be saved in the web based package.

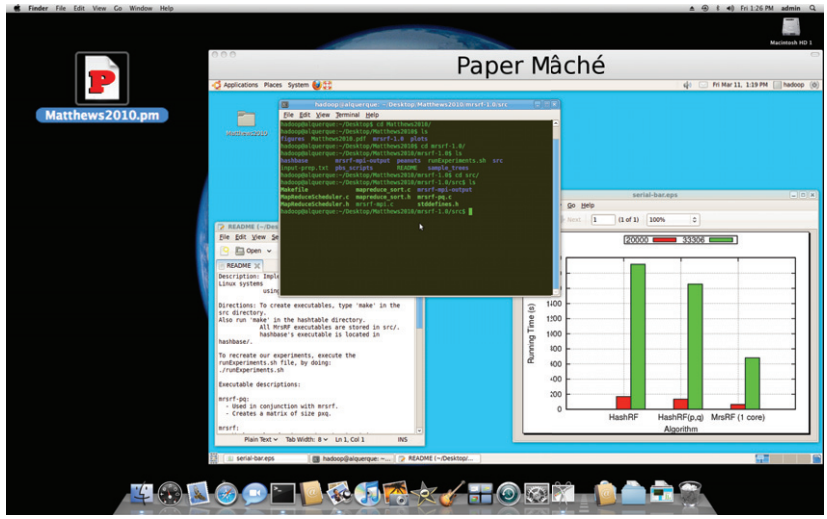

Figure 3: A prototype of the Paper Mâché system. Readers and reviewers download the `Matthews2010.pm` file to their desktop. Using a hypervisor to execute the `.pm` file opens up a new window, displaying the guest operating system (in this case, Ubuntu). In the context of this guest operating system, readers and reviewers can reproduce the experimental results discussed in the paper. When readers and reviewers are done interacting with the experimental environment, they can simply close the window.

Executing the paper within Paper Mâché is much easier than having each reviewer and reader set up the experimental environment on their own. In this example, the source code for MrsRF is available publicly from the web. The MrsRF source code takes advantage of Phoenix [12], an underlying MapReduce framework which was originally designed for the Solaris operating system. We then modified Phoenix to get working on some versions of Linux (e.g., Ubuntu and CentOS). However, if the readers and reviewers do not have access to those versions of Linux (or the correct version of the Gnu C Library [13]), then they cannot run our software properly. Gnu C Library (glibc) incompatibilities are especially difficult to deal with, since making haphazard updates to a system's glibc installation can lead to disastrous results. In the past, we had a real-life case involving a reader at another university who had Linux and glibc incompatibility issues with MrsRF. Despite the code being freely available on the web, and open, continuous communication between the authors and the user, the situation was only fully resolved when the reader ended up using a virtual machine to execute the MrsRF code. This finally allowed her to reproduce the results found in the paper. As a result, we believe that the integration of virtual machine files in Paper Mâché will quickly and easily allow users to recreate a paper's experimental framework and reproduce results.

Paper Mâché readers and reviewers will be able to easily interact with research. As a result of the `.vm` file, all components of the research (source code, libraries, etc.) will be exposed. Thus, if the reader desires to create an executable version outside of the virtual image (e.g., execute the paper on their operating system of choice), it will be far easier to construct such an environment with the working model that is available from within Paper Mâché.

3.3. The Paper Mâché file (`.pm`) and Workbench

An executable paper will be represented as a single `.pm` file within the Paper Mâché system. This file will be structured as a set of subdirectories as shown in Table 1, similar to the structure of a Java `.jar` file. The file will be built, updated and maintained by the Paper Mâché Workbench. It is not intended that the authors be directly responsible for updating the `.pm` files. Within the `.pm` file, there will be a set of directories corresponding to the sections of the executable paper as shown in Figure 2. The `paper` subdirectory holds the text of the paper as well as any figures referenced and any additional multimedia files. The `comments` subdirectory contains the comments and ratings entered by readers and reviewers as well as responses of the authors. The `metadata` subdirectory holds additional information (author's names and contact information, dates updated, etc.) associated with the paper. The `.vm` file will also be contained with the `.pm` file.

While readers and reviewers will primarily interact with the paper using the web-based Paper Mâché workbench, it will be possible to download the entire `.pm` file from the web if desired. The `.pm` file may then be expanded into a

\Matthews2010.pm	
\paper	• Files associated with the paper portion of the executable document
\text	• Text files such as html and pdf representations of the paper
\figures	• Figures and data referenced in the paper
\media	• Multimedia content (e.g. video, audio)
\comments	• Comments and ratings for the paper
\metadata	• Metadata associated with the package
MyPaper.vm	• Virtual machines image

Table 1: Contents of the Matthews2010.pm file shown in Figure 3. The .pm file will have a standard directory structure similar to Java .jar files. The paper, comments and .vm sections correspond to sections in Figure 2. The metadata section contains various information about the .pm package and its contents.

set of directories on the users machine. In this mode, any changes to the contents of the package will not be recorded in the web based copy of the .pm file.

The only required elements of the .vm file (in addition to an operating system) will be the source code to the applications, any dependencies required to build and run the applications, and scripts or instructions for reproducing the experiments referenced in the paper. Inclusion of other portions of the paper (e.g. pdf files) in the .vm file already contained in the .pm package will be optional.

The web based Paper Mâché Workbench will act as the point of contact for all users and provide robust capabilities for the management of .pm packages. Implementation details of the system include using contemporary web design (CSS, AJAX, etc.). The workbench will be developed using Ruby on Rails hosted on an Apache web server and interfacing with a MYSQL database. The workbench is organized around the actions associated with the various roles (author, reviewer, reader) an individual performs in a scientific community. For example, a casual reader just browsing a set of papers will only be able to view those elements that the author (and publisher) have enabled for view. For example, only authors can view the comments left to them by reviewers. Individual elements within the .pm file will also be secured. For example, to protect intellectual property, essential data may not be viewable until the paper is actually published.

All operations and functions within the workbench will be secure. For example, a standard role-based security model (e.g. author, reviewer) will be used in conjunction with an overall state for the .pm file (e.g. under construction, in review, published) to allow security to be varied depending on where the paper is in the publishing cycle. Each element within the .pm file will have an Access Control List (ACL) to provide highly granular control over security. Additionally, the workbench provides revision control on the contents of the package to allow those with appropriate security to view and revert changes to elements with the package. Whenever possible, elements within the package will be watermarked with codes that would allow for tracking of the elements back to the original package to reduce plagiarism.

To create and execute virtual machines, the workbench will provide a downloadable tool that will assist the author in creating a virtual image from an existing machine. This image may then be uploaded. Moreover, the workbench will provide the ability to execute the virtual image providing a remote desktop to the user (VNC, Windows Remote Desktop). Through the workbench, the reader will be able to interact with the paper, and participate in public discussions (see Figure 2). By taking advantage of the security features allowed by the Paper Mâché hypervisor, authors will be able to "lock down" portions of the virtual machine to prevent coping of unpublished research. Authors using the workbench can create new packages and modify packages they own. They will also be able to view and respond to comments, whether from readers (public), or from reviewers (private). Lastly, reviewers can use the workbench to interact with the paper and leave comments to the authors.

4. Evaluation of Paper Mâché

We will evaluate the performance of our Paper Mâché system based on two metrics: *speed* and *understanding*. For example, our first experiment will measure the time it takes for a reader/reviewer to interact with the science described in a paper. That is, consider Table 2. The traditional approach requires seven steps to interact with the science in a paper. Of course, this assumes that the software is available publicly or directly from the author and

that readers and reviewers can get the source code compiled on their experimental platform and that the data is also available for experimentation. For Paper Mâché, Table 2 shows that there are three steps that are needed to recreate the experiments that are discussed in the scientific paper of interest. For our performance evaluation, we will select a set of papers from various conference and journal publications, and measure the average amount of time that the traditional mechanism requires the reader/reviewer to obtain the science discussed in a paper. We will compare that number to the time required under Paper Mâché. Given that Paper Mâché is a new system, we will ask for volunteers to store their scientific results on the system so that we have a large enough sample for comparison with the traditional approach.

Secondly, we will measure the amount of time required by an author to use Paper Mâché to package the executable portion of their paper. The traditional approach places little overhead on the author in terms of making the executable portion of their paper available. Instead, the overhead is placed on each reviewer and reader to spend the time required to recreate the experimental environment used by the author (as shown in the Traditional column of Table 2). Clearly, there will be overhead placed on the author in order to use Paper Mâché. However, the time spent by the author is time saved for each reviewer and reader that accesses the paper. We are interested in the amount of time required by the author to share their executable environment in Paper Mâché. Our hope is that the additional time required by the author is minimal when compared to the savings gained by reviewers and readers to interact with the science described in a scientific paper.

Finally, we will experiment with accessing the improvement in understanding a scientific paper as a result of interacting with it through Paper Mâché. Improved understanding means a better experience interacting with a paper for readers and reviewers. Certainly, user studies will be conducted as a way to measure understanding. However, we will also consider other types of experiments. For example, one way to measure understanding is to measure the level of engagement with a scientific paper. We could measure the amount of time spent reading a traditional paper compared to the amount of time spent with a Paper Mâché package. Our hypothesis is that engaged readers will have better comprehension of the scientific content. We could compare the frequency of comments on papers with and with out an attached virtual machine as one metric of accessing interest and to some degree understanding. Furthermore, it will be interesting, to measure whether techniques (such as Paper Mâché) that increase participant's engagement in a paper positively affects the impact factor of scientific journals.

5. Extending the Capabilities of Paper Mâché

Section 3 provides a description of the primary features that are the focus of the current development of our Paper Mâché system. However, Paper Mâché can be extended in many ways in order to improve the experience of authors, readers, and reviewers. For example, cloud computing has become a major topic of interest and one that could provide further utility to Paper Mâché. By hosting the hypervisor in the cloud, users can execute papers from a web browser. This would offload the computational requirements from the user to the host server. With enough computing power server side, we could enable users to test and interact with super computing scale research from their commodity hardware.

Community features can significantly add to the reader's experience. Imagine two copies of the same paper: one fresh off the printer and the other annotated by a graduate student who has poured over the research. Wouldn't both paper types be helpful in understanding the work? While it might be preferential to first read the paper as it was published before turning to an annotated copy, those notes and comments could be invaluable to understanding complicated passages, figures, or algorithms. One of the goals of Paper Mâché is to allow readers to pick up right where the authors left off. Furthermore, the ability to share detailed comments and annotations allows readers to experience the paper from different perspectives.

Science does not remain static. New experiments are performed, tweaks to code are made, and different data sets are tested. Since research does not stop once a paper is published, it would be useful if these advances were represented in the executable paper. By combining Paper Mâché with version control systems, executable papers could be made to reflect the ever advancing nature of research. The beauty of version control systems is that they track changes. Hence, the original work can always be available while also making it easy to obtain the most recent version of the software. Small changes to projects do not often warrant a new publication. However even something as simple as a bug fix could have great impact on readers and researches interested in the research. Thus, version control systems are an important step in keeping the community up to date on the most recent iteration of a project.

Traditional	Paper Mâché
1. Obtain source code	1. Install VM framework*
2. Resolve OS / platform dependencies (32 bit vs. 64)	2. Download VM
3. Resolve library dependencies	3. Run Experiments
4. Compile with proper flags	
5. Obtain data set used in paper	
6. Obtain the commands used to run experiments in the paper.	
7. Run experiments	

Table 2: A table comparing the steps required for executing the science in a scientific paper using the traditional approach and Paper Mâché. The $*$ denotes that this step is only required the first time a user uses Paper Mâché.

6. Conclusions

In this paper, we introduce Paper Mâché, a novel system for creating dynamic, executable research papers. While virtual machines are widely used to maintain controlled, reproducible environments for software development and testing, Paper Mâché extends the use of virtual machines to facilitate the reproduction of scientific research. By allowing authors, reviewers, and readers to interact with not just the text but its programs and data in a virtual machine environment, the scientific paper becomes a dynamic, executable entity. Short and long-term compatibility is assured through the use of virtual machines. The programs and data associated with the paper will be runnable even if the actual source code no longer compiles in modern environments.

Virtual machines allow for easy, instant execution providing a quick method for validation of the programs and data. The robust security model associated with the Paper Mâché packages provides a ideal method for managing copyright and licensing issues. The capabilities of any user (or role) can be managed based on the licensing requirements of operating systems and applications. The cloud environment and the ability to seamlessly work with the authors host environment will provide the ability to work with large scale systems and large file sizes. By providing a single point of management (the Paper Mâché Workbench) it becomes possible to track the provenance of individual elements within the papers.

However, the benefits of Paper Mâché extend beyond the scientific community. The interactive aspects of our Paper Mâché system encourages the interest of science and the scientific process amongst the general public, thanks to an increase in visibility and accessibility of current research. The increase in accessibility to current findings changes the way that scientific research is performed and communicated. We believe paper management systems such as Paper Mâché have the ability to pave the way for more scientific collaborations, increases the communication and understanding of core concepts, and will consequently allow for earlier adoption of critical findings into existing research. Thus, our Paper Mâché system provides a bridge that allows everyone to actively participate in the scientific process.

7. Acknowledgements

This publication is based in part on work supported by Award No. KUS-C1-016-04, made by King Abdullah University of Science and Technology (KAUST). This work was also supported by the National Foundation under grants DEB-0629849, IIS-0713618, and IIS-1018785.

References

[1] A. Voronkov, Easy chair conference system, Internet Website, last accessed, March 2011., available from http://www.easychair.org.
[2] Sofconf.com, START v2, Internet Website, last accessed, March 2011., available from http://www.softconf.com/about/.
[3] L. LLC, Linklings, Internet Website, last accessed, March 2011., available from http://www.linklings.com/.
[4] T. Reuters, ScholarOne manuscripts, Internet Website, last accessed, March 2011., available from http://scholarone.com/products/manuscript/.
[5] Springer, Citeulike: Everyone's library, Internet Website, last accessed, March 2011., available from http://www.citeulike.org/.
[6] R. Figueiredo, P. Dinda, J. Fortes, A case for grid computing on virtual machines, in: Distributed Computing Systems, 2003. Proceedings. 23rd International Conference on, 2003, pp. 550 – 559. doi:10.1109/ICDCS.2003.1203506.

[7] B. Walters, Vmware virtual platform, Linux Journal 1999.
URL http://portal.acm.org/citation.cfm?id=327906.327912
[8] P. H. LTD, Virtualization and automation solutions for desktops, servers, hosting, saas - parallels optimized computing, Internet Website, last accessed, March 2011., available from http://www.parallels.com/.
[9] J. Watson, Virtualbox: bits and bytes masquerading as machines, Linux Journal 2008.
URL http://portal.acm.org/citation.cfm?id=1344209.1344210
[10] P. Barham, B. Dragovic, K. Fraser, S. Hand, T. Harris, A. Ho, R. Neugebauer, I. Pratt, A. Warfield, Xen and the art of virtualization, SIGOPS Oper. Syst. Rev. 37 (2003) 164–177. doi:http://doi.acm.org/10.1145/1165389.945462.
URL http://doi.acm.org/10.1145/1165389.945462
[11] S. Matthews, T. Williams, MrsRF: an efficient mapreduce algorithm for analyzing large collections of evolutionary trees, BMC Bioinformatics 11 (Suppl 1) (2010) S15. doi:10.1186/1471-2105-11-S1-S15.
URL http://www.biomedcentral.com/1471-2105/11/S1/S15
[12] C. Ranger, R. Raghuraman, A. Penmetsa, G. Bradski, C. Kozyrakis, Evaluating mapreduce for multi-core and multiprocessor systems, in: High Performance Computer Architecture, 2007. HPCA 2007. IEEE 13th International Symposium on, 2007, pp. 13–24. doi:10.1109/HPCA.2007.346181.
[13] S. Loosemore, R. Stallman, R. McGrath, A. Oram, The GNU C Library: Reference Manual, Free software foundation, 1996.