Available online at www.sciencedirect.com

ScienceDirect

Procedia Computer Science 4 (2011) 608–617

Procedia Computer Science

Executable Paper Grand Challenge
International Conference on Computational Science, ICCS 2011

# The Collage Authoring Environment

Piotr Nowakowski[a]*, Eryk Ciepiela[a], Daniel Harężlak[a], Joanna Kocot[a], Marek Kasztelnik[a], Tomasz Bartyński[a], Jan Meizner[a], Grzegorz Dyk[a], Maciej Malawski[b,c]

[a]*ACC CYFRONET AGH, ul. Nawojki 11, 30-950 Kraków, Poland*
[b]*Institute of Computer Science AGH, al. Mickiewicza 30, 30-059 Kraków, Poland*
[c]*Center for Research Computing, University of Notre Dame, USA*

**Abstract**

The Collage Authoring Environment is a software infrastructure which enables domain scientists to collaboratively develop and publish their work in the form of executable papers. It corresponds to the recent developments in both e-Science and computational technologies which call for a novel publishing paradigm. As part of this paradigm, static content (such as traditional scientific publications) should be supplemented with elements of interactivity, enabling reviewers and readers to reexamine the reported results by executing parts of the software on which such results are based as well as access primary scientific data.

Taking into account the presented rationale we propose an environment which enables authors to seamlessly embed chunks of executable code (called *assets*) into scientific publications and allow repeated execution of such assets on underlying computing and data storage resources, as required by scientists who wish to build upon the presented results. The Collage Authoring Environment can be deployed on arbitrary resources, including those belonging to high performance computing centers, scientific e-Infrastructures and resources contributed by the scientists themselves. The environment provides access to static content, primary datasets (where exposed by authors) and executable assets. Execution features are provided by a dedicated engine (called the Collage Server) and embedded into an interactive view delivered to readers, resembling a traditional research publication but interactive and collaborative in its scope.

Along with a textual description of the Collage environment the authors also present a prototype implementation, which supports the features described in this paper. The functionality of this prototype is discussed along with theoretical assumptions underpinning the proposed system.

*Keywords*: executable paper; high performance computing; e-Science; scientific publishing

## 1. Introduction

Many disciplines of modern computational science – high-energy physics, molecular biology, social research and material studies to name just a few – face an obvious need to enrich and enhance the current state of scientific

* Corresponding author. Tel.: +48-600-280-105.
*E-mail address*: p.nowakowski@cyfronet.pl.

 Selection and/or peer-review under responsibility of Prof. Mitsuhisa Sato and Prof. Satoshi Matsuoka
doi:10.1016/j.procs.2011.04.064

publications. Given traditional publishing methods, data sets, code and actionable software are absent when research is recorded and preserved as a journal article, book chapter or any other paper-based publication. Furthermore, the actual data which is the result of (and frequently the basis for) published research is not preserved and made accessible in a coherent manner – thus, the reader of a scientific paper must often take the author's word that the professed results and conclusions are, in fact, valid. Clearly, the scientific paper itself no longer conveys sufficient information to enable reviewers and other readers to judge it on its own merits. This phenomenon endangers the scientific process and carries serious implications for further progress in computational sciences, which necessarily build upon to-date published results.

The Collage Authoring Environment is an attempt at resolving this issue: just as the Web has come alive through the use of mashups and content embedding technologies, so too can research papers benefit from interactivity and dynamic content generation. The authors propose a system which enables a scientific publisher to deploy an infrastructure for the storage and provisioning of executable papers, consisting of static bits of text (much like traditional scientific publications) along with access to primary datasets and embedded assets facilitating execution of author-supplied code by publication readers.

This paper is organized as follows: in section 2 we present the motivation and objectives of our work. Section 3 lists the current state of the art in scientific publishing and outlines some ongoing initiatives which aim to extend the traditional publication model with the capabilities offered by modern computing and networking tools. Section 4 focuses on a more in-depth discussion of the Collage environment design, while section 5 outlines the features implemented as part of the first prototype of the tool. The paper ends with conclusions and prospects for future development.

## 2. Motivation and Objectives

Even though the past two decades have witnessed the development and spread of computer-aided research techniques (e-Science), the mechanism by which scientific advances are communicated to the general public has remained unchanged for more than a century: it is the scientific paper. The shortcomings associated with this mode of publication are well known: scientific papers do not yield themselves to rapid verification, reproducibility and reuse of research achievements. This is particularly troublesome given the fact that data management and access technologies have made great strides in the recent years, yet such progress is not reflected by the procedures and traditions associated with publishing scientific research. The pressing need for new solutions is illustrated – for instance – by the requirements imposed by certain publishers upon reproducibility of results and the provision of data which would enable such reproducibility. Rather than providing a simple means by which textual information is conveyed, much like displaying a motionless clock, the scientific publication should instead become a vessel for the enactment of the algorithms used to generate results, whereby the internal workings of the published research – its "movement" – can be directly studied in action.

On the basis of this goal we can derive some specific objectives which should – in the authors' view – be met by any infrastructure purporting to realize the executable paper vision. Accordingly, the following assumptions can be treated as a starting point for the framework proposed in this paper:

- **Executability**: The executable paper is necessarily interactive, i.e. it must enable the execution of arbitrary computations (as determined by the authors) on underlying computing resources. There must be a way to embed such interactive elements in the paper's predetermined structure (which follows the natural flow of the scientific publication – from the problem statement and initial assumptions, through details of the proposed algorithms, to validation of results and conclusions);
- **Compatibility**: The infrastructure should be compatible with a wide array of data sources and computational platforms, including modern Cloud environments. In addition, the environment should be sufficiently extensible to enable integration with potential future computing solutions;
- **Validation**: The infrastructure should support tracing and validating research results, particularly by reviewers of scientific papers who may be familiar with the given discipline;
- **Licensing**: The authors of the scientific paper should be free to expose only such data and computations as they are entitled to; moreover the authors, in conjunction with the publishers, should exercise control over who is allowed access to particular research papers;

- **Computational access**: The environment should be structured in such a way as to enable computations to be executed on high-performance computers, where available.
- **Data access**: The executable paper should also facilitate access to primary data – namely, data which is the basis for the results presented in a given paper. Such data may assume various guises and be represented by databases or flat files. Thus, the environment must support embedded links to data elements as part of the executable paper and retrieval of said data with the use of a dedicated engine, along with extensions for various storage mechanisms;
- **Collaborative development support**: The environment must be collaborative in the sense that the research paper can be co-authored and amended by a community of authors, each contributing to its content;
- **Multi-actor environment**: The environment must provide facilities for *authors* of scientific publications as well as for *readers* (including reviewers) as we assume that authors will have a higher degree of control over the content which is being displayed while the readers will be presented by a more constrained view (which will, nevertheless, still enable them to interact with the paper's content in a meaningful way);
- **Evolutionary approach**: The executable paper, while necessarily being rendered by a computer (contrary to a printed publication), should still formally resemble a traditional research paper so that its structure appears familiar to readers. Executable content should be embedded in the text as frames;
- **Security**: Security must be implemented to ensure that authors can limit access to their research data, and to prevent malicious code from being injected and executed in the infrastructure.

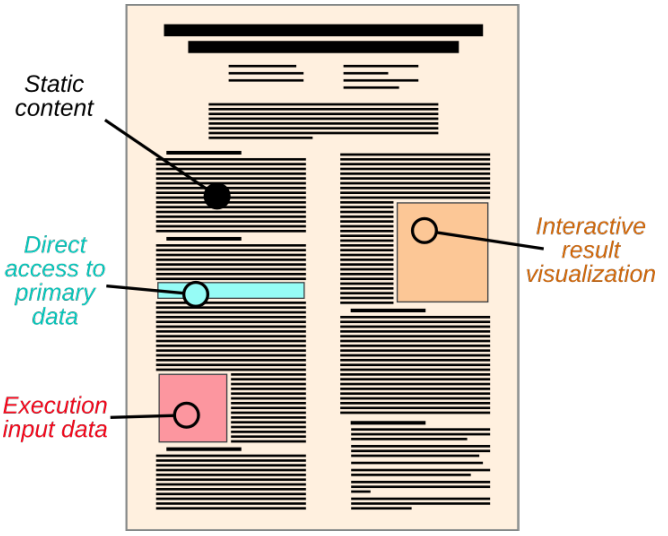


Fig. 1: Conceptual view of the executable paper. Static content (the body of the publication) is extended by interactive elements. Readers can access primary data and reenact computations in order to validate the presented conclusions or navigate result spaces. Subject to the authors' approval, readers can also obtain access to the underlying code of the experiments presented in the publication. The infrastructure is Web-based and can be integrated with the Publisher's portal.

A cursory depiction of the structure of the executable paper, as envisioned by the authors, can be seen in Figure 1. Of note is the fact that we intend to preserve the traditional "look and feel" of the publication, and not replace it with an entirely new interface. We believe that the way towards wide-scale adoption of novel publishing paradigms is not to eschew the existing model, but rather to extend it with additional functionality. In Sections 4 and 5 we explain how we intend to pursue the objectives stated above and bring this vision closer to reality.

## 3. State of the Art in Scientific Publishing

Much effort is afoot to address the development of richer publication formats and better integration between research data and the final published article (see http://users.emulab.net/trac/archive10/wiki/PublishStandard for an overview of funding agencies and reports pertaining to such increased integration developed for the NSF's 2010

"Workshop on Archiving Experiments to Raise Scientific Standards", https://www.protogeni.net/trac/archive10/). At the same time it becomes apparent that browsing and citing scientific papers is an activity that can largely benefit from social networking tools and websites. For example, CiteULike (http://www.citeulike.com) can be used to post bibliographic references to the papers into a personal database with a one click in the Web browser. Moreover, the references can be shared within a group writing an article, which greatly helps finding related papers and forming communities which share the common interest. Zotero (http://www.zotero.org) is a similar tool which integrates with the Web browser, while desktop tools such as Mendeley (http://www.mendeley.com/) can process local PDF files and integrate with text processing suites like Word or LaTeX. Linking the articles with primary data is pursued by the UTOPIA Documents project (http://getutopia.com) [1], which aims to bring PDFs to life by linking to live resources on the web and turning static data into live interactive content, such as curated database entries, molecular structures, sequence and alignment data and plots. myExperiment (http://www.myexperiment.org) is a Web 2.0-based Virtual Research Environment and Repository for the storage, social curation and community contribution of scientific workflows, supporting the sharing, reuse and execution of workflows by close integration with virtual laboratory frameworks. Research on new ways of utilizing the possibilities offered by Internet and social networks for scientific publications has been pursued in EU-funded projects like SciX (http://www.scix.net), PEER (http://www.peerproject.eu/) and Liquidpub (http://liquidpub.org) and also gains attention from the collaborations of professionals such as Concept Web Alliance (http://conceptweblog.wordpress.com/declaration/). An interesting example of how Web 2.0 technologies may be leverage in collaborative scientific endeavors may also be found in [2].

Several useful tools are being developed in the life sciences to help identify and enrich entities in biology. As an example, EBI's Reflect tool (http://www.reflect.ws), winner of the Elsevier Grand Challenge (http://www.elseviergrandchallenge.com), offers a commodity Firefox plugin that allows entity recognition of any biological (and, increasingly, nonbiological) entities that are given in one of the ontologies linked to the database. On mouse-over, the tool pulls in information from various content sources and represents it as tabs on a pop-up menu. A more expansive entity identification interface was recently launched by PubMedCentral (http://beta.ukpmc.ac.uk/), which affords users the ability to highlight terms from the Gene Ontology and/or protein thesauri. Clicking on an entity brings the user to the corresponding EBI page on a specific biological entity.

Recently, Web 2.0 technologies have enjoyed considerable success in entertainment and journalism and social media, enabling users to add value to Web applications [3]. The benefit of Web 2.0 for science has also been recognized [4]. For example, Giustini [5] describes the influence of blogs, wikis or RSS feeds on medicine. There are some Web 2.0 social network and resource sharing platforms for research, wherein the researchers can establish and maintain contacts, build communities, organize work space for collaborative projects, or share and reuse scientific content resources including collaborative curation through tagging and comments [6]. However, naïve approaches to sharing and reusing are insufficient and inappropriate in the realm of science [7]. Care must be taken to protect intellectual property, appropriately attribute ownership, credit and attribution and support the scientists' activities within the scientific communities' culture of reward and reputation. myExperiment (http://myexperiment.org) [8] is cited in [6] as an example of the second generation of social networking and sharing sites that are mindful of the specific needs of scientists. Originally designed to share scientific workflows that are expected to be reused and adapted, it has a sophisticated model of Versioning, Ownership, Sharing, Credit, Attribution and Permissions for individuals, groups and networks, under the direct control of the members.

Lastly, experimental scientific papers convince readers of their core claims by using data: research data, represented in figures and tables, and by equations, chemical formulae and such. There is a large move afoot to integrate, store, annotate and employ such data in an enhanced way to scientific papers; this is an important, challenging and exciting development. In this light, the wide adoption of the Linked Data philosophy seems very promising: here, pages are rendered either in HTML for human consumption, or in RDF for computer consumption. In pharmacology and therapeutics, there are ongoing efforts to enable greater "connectivity between data silos (...) to connect drug and clinical trials related data sources" [9]. In computer science, a recent plea [10] was for authors and publishers to "provide mechanisms for publishing software, inputs, and experimental data as metadata for the publications that report these experiments in a deep and rigorous manner."

While the projects mentioned above aim at addressing the problems of linking and referencing scientific data, publications, services and even executable workflows, they do not provide for a unified infrastructure which can link

together the computing and data e-infrastructure, execution engines and the resulting executable paper, which is the ultimate objective of Collage.

## 4. The Collage Authoring Environment: Concept and Design

The Collage Authoring Environment operates by presenting domain scientists with a framework which can be used to develop, test and run experiments in computational science, expressed in a variety of programming languages. This environment bases on the GridSpace2 platform, developed as the end-user interface for the PL-Grid HPC infrastructure. While it provides all the features traditionally expected of a programming environment, Collage also enables the developer to publish and expose fragments of the developed experiments (called *assets*) as external, embeddable entities which can subsequently be visualized in a digital edition of a research paper. Such assets support interactive visualization of research results (backed by the underlying computing resources), enabling reviewers and readers to enter arbitrary input data and witness the published scientific algorithms in action, and to efficiently browse data sets which would be difficult to publish by traditional means (e.g. attachments to standalone research papers). Moreover, the framework itself is capable of interfacing with popular scientific software suites, such as Matlab, Mathematica, Packmol, GAMESS and many others.

Collage is aimed at two principal user groups, namely *authors* of scientific papers (i.e. anyone who has a hand in carrying out a scientific experiment or authoring a publication which bases on its results) and *readers*, i.e. all users not directly involved in preparing the paper itself, but interested in its content and the scientific data it presents. Clearly, the latter group also includes reviewers of scientific papers.

### *4.1. External interfaces*

The view presented by the Collage environment to the end users (i.e. authors and readers) assumes the form of a HTML document in which static content is interspersed by dynamically generated forms, representing the "executable" part of the paper. While the document may superficially resemble a traditional research publication, it also contains embedded *assets*, each of which presents a visual interface to the end user (typically in the form of a diagram, graph, figure etc.) and is capable of requesting the underlying infrastructure to repeat a given part of the experiment or redisplay its results depending on the parameters specified by the reader. Initially this interface assumes a predefined default state, as dictated by the experiment; however it may change following execution (at the reader's request). While the executable paper is being loaded, each asset refers to a predetermined data piece, provided by the Collage server (which – if needed – can forward execution requests to underlying computing and storage resources). As results arrive, placeholders are replaced in the paper view with actual results of computations.

The authors of the paper are thus provided by two distinct user interfaces: a Web environment where they can code their experiments and determine the extent to which input data can be manipulated by the end user (also called the Experimentation UI) as well as a separate interface which enables them to develop the actual executable paper as a Web document with embedded assets (this is called the Authoring UI). Both types of interfaces are further discussed in section 6. Since generating results on the fly might prove unfeasible, assets can also access local storage (flat files and databases) where results of previous runs are cached.

### *4.2. Collage assets*

Three types of *assets* are envisioned in the infrastructure. All assets belonging to a given type share a common rendering widget by which they are represented in the executable paper. They also share a set of common actions which can be performed on their content. The asset types foreseen in Collage are:

- **input forms** – the goal of this asset is to visualize an input form in the executable paper view. The form can be used by the user to feed input data into the running experiment. This type of asset directly implements the interactivity aspects of the executable paper as the user will be able to browse large result spaces with the aid of input forms. Upon submission of an input form, the collage server will receive the input data and may further apply it in the course of processing the experiment, possibly generating further assets. An example of this functionality would be an interactive graph, which can be manipulated by the user through an input form. Each

time the input form is filled, and submitted, the server reruns the required computations and generates a fresh graph (which is rendered as a visualization – see below). Should computations become too complex to be performed in real time, the relevant results may be read from data sources (databases and/or flat files), according to the input specified by the reader. Note that this type of asset can also be used to upload data files to the infrastructure for processing;

- **visualizations** – the goal of this asset is to render an experiment result which can be directly visualized in the research paper. Typically, this would be a figure, diagram or chart, although the environment itself does not impose restrictions on the nature of the visualization. In the case of inherently static visualizations (such as images), either the publisher server or the client browser may issue periodical requests to the Collage server, to determine whether the payload of a given visualization has changed (which may often be the case – for instance as partial results are returned by the computing backend). Should this be the case, the contents of the visualization asset will be automatically updated in the client browser. Furthermore, the Collage server can also detect that a given visualization is not yet available (e.g. if the underlying computations are still in progress) and notify the Publisher server (or client browser) so that an appropriate message may be displayed as the client awaits results.
- **code snippets** – these assets embed an editable view of the code which enacts a specific computation and may be used to generate additional assets. The purpose of this type of asset is to enable the reader to exercise more in-depth control over the experiment execution process, and also to review the inner workings of the executable paper for the purposes of validation and/or reuse. Subject to the author's control, the code of experiment snippets may be stored and the experiment reenacted.

Together, the three asset types mentioned above cover the full spectrum of interactivity which is required by the executable paper, as presented in Section 2. The authors can prepare and deploy assets (including executable code) while the viewers can interact with them by means of input forms. Results are displayed via visualization assets and can be refreshed by the server as more output data becomes available. The system may also visualize external (non-Collage) assets referenced by static URLs.

### *4.3. Conformance with stated objectives*

In order to better justify the proposed architecture it may be beneficial to refer to Section 2 and explain how Collage matches each of the objectives described there. This is done in Table 1, which lists each of the presented objectives and explains where Collage fits in.

Table 1. How Collage addresses the objectives stated in Section 2

| Stated objective | Collage Contribution |
|---|---|
| **Executability** | The Collage environment supports exposure of executable code, which can be embedded in the structure of the publication with the help of suitable assets. The user may enter input data into a form, whereupon the system proceeds with computations and retrieves results as an embedded fragment of the paper |
| **Compatibility** | The Collage server is capable of submitting computations to a variety of computational resources, as well as retrieving data from networked data storage elements. Moreover, as the server is capable of executing arbitrary code in a number of scripting languages (including Ruby, Python and Perl), any resources which can be interfaced with the use of these languages are automatically available to Collage assets. |
| **Validation** | The readers and reviewers of scientific papers are capable of rerunning experiments (where supported by the given publication) and validating their results, as well as reviewing the computations which produce these results by accessing the underlying code. Subject to the authors' approval, the environment may also facilitate modifications in code snippets and repeated execution of experiments with the use of the updated code. |
| **Licensing** | As the Collage environment is served through a Web-based environment, it can be easily integrated with the Web portal of any scientific publisher, whereupon the Publisher's licensing schemes become applicable to Collage papers. The authors of scientific papers may additionally control access to primary data by supplying custom user credentials to be applied by Collage when accessing such data. |
| **Computational access** | The Collage engine can interface computing infrastructures, including existing Grid infrastructures |

| | |
|---|---|
| | operated across the EU and in other areas of the world. Support for Cloud computing is also foreseen. In addition, the authors of scientific papers may schedule computations on their own resources contributed to the executable paper (for instance, a public Web Service operated by a scientific institution). |
| **Data access** | Data access is provided by means of custom assets, which may be mediated by the Collage server or refer to data elements directly. The only requirement here is that a given resource is available under a known URL. |
| **Collaborative development support** | The Authoring UI (described in Section 6) enables multiple authors to collaborate on the development of a single paper. Moreover, the Collage environment supports decomposition of virtual experiments into *snippets*, each of which may be coded by a different person and apply different programming tools. |
| **Multi-actor environment** | Collage is specifically designed to offer different features to *authors* of scientific papers (who, in addition to authoring the paper itself, may use the environment to further develop and execute their virtual experiments) and *readers* who interact with the paper by means of assets specified by the authors. |
| **Evolutionary approach** | The environment attempts to simulate a traditional paper view, by enabling authors to author their publications as HTML documents, embedding executable assets where necessary. |
| **Security** | Collage provides secure login features guarding access to sensitive data or computations. Moreover, as the environment is Web-based and may integrate with the Web resources of a scientific publisher, it is also possible to extend the access control mechanisms applied therein to Collage features. |

On the basis of this comparison, we can conclude that the Collage environment constitutes a good match with the stated goals of an executable paper platform, as described in Section 2.

## 5. Implementation Details

The authors have substantial experience in developing computing solutions for domain scientists, as evidenced by their longstanding involvement in development of environments for collaborative applications [11]. Among the outcomes of our work is the GridSpace virtual laboratory and workbench, on which the Collage environment is partly based [12]. This section is intended as a broad overview of the design and operation of the Collage environment, including technical aspects of its implementation.

### *5.1. Interaction with end users*

As already mentioned in Section 2, two user roles can be distinguished in the Collage environment: the *author*, i.e. the person responsible for preparing the paper and its associated assets, and the *reader* who interacts with the paper once it has been prepared and served by the Collage infrastructure. Each paper can have multiple authors and, likewise, multiple readers (some of whom may act as reviewers, although this issue is not relevant in the scope of this paper). This situation is depicted in Figure 2, which lists the major building blocks of the Collage infrastructure along with pairwise interactions between actors and/or software modules. It should be noted that the architecture restricts reader access to the Presentation UI exposed by the Publisher server, while the author may additionally access a dedicated Experimentation UI and Authoring UI, used – respectively – to deploy executable code and set up the actual structure of the target publication.

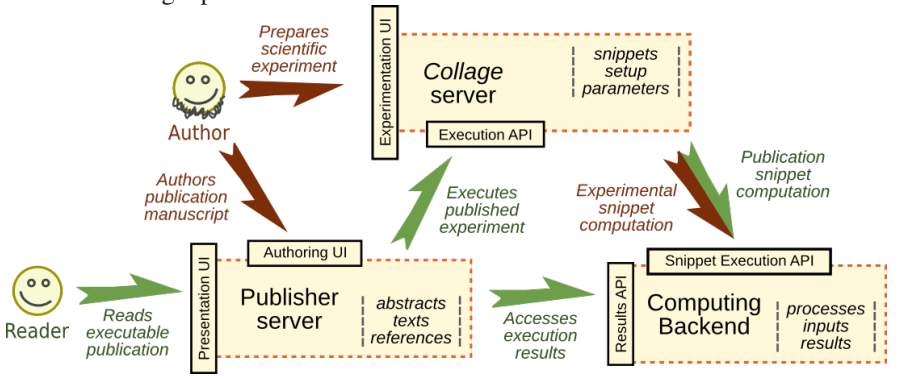

Fig. 2. Interaction of user groups with the Collage environment. Authors of scientific publications can set up executable content by exploiting the Experimentation UI and Authoring UI as appropriate. Readers have access to a dedicated Presentation UI, which visualizes the publications and permits interaction with Collage assets.

The Publisher server refers to the Collage server via a separate API (the Execution API), whenever an asset needs to be visualized or updated. This is further explained in the following subsection. In turn, the Collage server may delegate computations to the Computing backend (comprising computational and/or data storage resources, either operated by the Publisher or supplied by the publication authors). The Publisher server commands the Collage server to execute the entire experiment, which may consist of multiple snippets. The Collage server is responsible for management of snippets and can externalize their execution according to the experiment workflow (represented by a snippet sequence). Results are retrieved from the Computing Backend by the Publisher server for direct visualization by means of the Presentation UI.

### *5.2. Rendering executable papers*

In accordance with the above schema, the core functionality of Collage will be provided by the execution environment, which is interfaced by the Publisher Server (PS); essentially a web server capable of delegating execution requests to the underlying Collage Server (CS), as depicted in Figure 3. The user requests a document, triggering a HTTP/S GET request which is dispatched to PS. PS serves the contents of the document (HTML) and may issue an embedded experiment execution request to CS. The request contains the input data necessary for running the experiment asynchronously, returning a unique run identifier (RUNID) to PS. Whenever "live" links to assets appear, PS inserts RUNIDs into the links and then embeds the links themselves as separate IFrame elements. The resulting page is presented to the user. As the browser renders the document, each link issues further HTTP/S requests directly to CS.

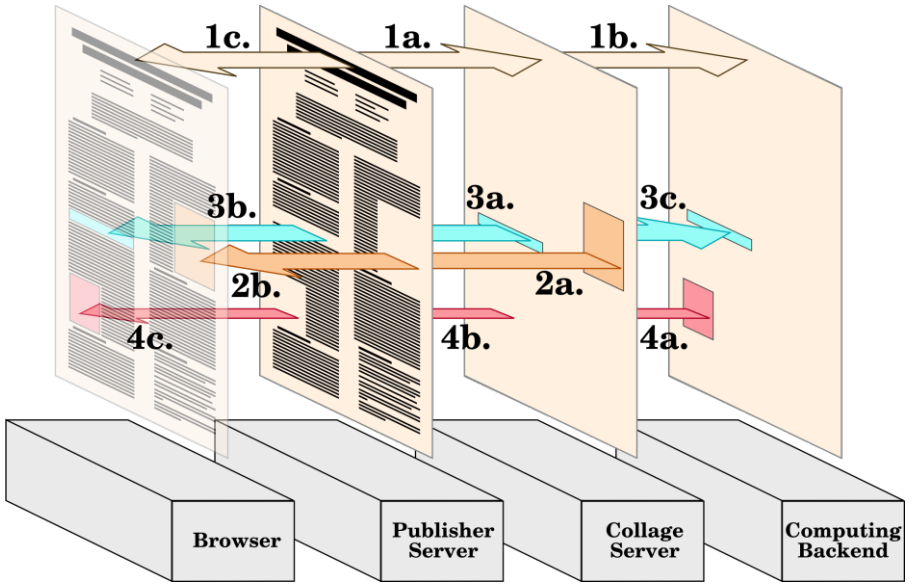


Fig. 3. The Executable Paper rendered by Collage. Static elements of the document are served by the Publisher Server (PS) while interactive elements are delegated to the Computing Backend (CB) via the Collage Server (CS). Results are visualized in the user's browser.

Each link requests CS (via HTTP/S GET) to provide a specific asset (e.g. an image, text file etc.) Parsing the presented RUNID enables CS to determine whether a given computation has concluded. From the reader's perspective, parts of the document, representing specific assets, may remain in a wait state until results are available and can be visualized. The end result resembles a successive sequence of updates, akin to the "notebook" function provided by Mathematica.

In order to enable users to run experiments with arbitrary input data, CS serves data input assets as HTML forms which the users may fill and send back to CS via HTTP/S POST. Such form consists of predefined widgets which may be specified by the authors with the use of the Experimentation UI. The input data is then processed and can be fed into the experiment code in order to generate further assets (which are again forwarded to the browser for rendering).

The detailed sequence of actions is as follows:

- (**1a**) Publisher Server (PS) requests Collage Server (CS) to execute an experiment which generates the paper. In response, the CS returns a list of assets which are provided by the paper, along with the URLs at which these assets are expected to be found;
- (**1b**) CS deploys an instance of the experiment on the Computing Backend (CB);
- (**1c**) PS serves the stub of the paper to the user's Browser (B). The Browser uses IFrames to represent interactive assets, initially populating them with placeholder content pending reception of the expected assets from CB;
- (**2a**) Static elements may be served by CS directly (without waiting for CS to respond to execution requests);
- (**2b**) Static elements are forwarded to the Browser as a mashup, without passing through PS;
- (**3a**) For dynamic assets, CS may serve an input form (if input data is required) or output data (in the case of a visualization or snippet asset), which is then rendered within an IFrame directly in the Browser;
- (**3b**) Data requested and collected from input forms does not pass through PS – this enables the users to retrieve large data sets and does not introduce undue load on PS;
- (**3c**) User input is fed into the experiment as required;
- (**4a**) The experiment generates results;
- (**4b**) CS forwards the results from CB to B, to be rendered as a visualization asset in a separate frame belonging to the paper;
- (**4c**) The result does not pass through PS – instead, it is downloaded directly from the Computing backend by means of a dedicated URL (containing a specific RUNID).

As already mentioned, the browser uses IFrames to render the output of assets. The advantages of this approach are threefold. First, the asset payload can be retrieved and updated in an asynchronous manner, irrespective of the static content of the paper. Second, each IFrame may host code (for instance, a JavaScript library) which is specific to the type of asset in question and facilitates its proper visualization in the asset window. Such code is provided by the Publisher server upon instantiation of the executable paper. Finally, the assets may be represented by URLs, which means that they may directly point to the resources included in the paper (particularly datasets), even if such resources are external to the Collage framework.

### *5.3. Interfacing computing and data storage resources*

Collage is intended as a generic environment which does not force users to use a specific programming language or hardware platform. Owing to the functionality already present in GridSpace [13] we can execute computations on a variety of resources, from local machines, through Web Services to PBS queues and Cloud systems [14]. The system supports a variety of scripting languages, including Ruby, Python and Perl, as well as direct shell programming on the Computing backend. Likewise, the developer of a virtual experiment is not constrained in the choice of data storage solutions that can be interfaced. In most cases it should be sufficient to paste existing code into the Experimentation UI environment as separate snippets, registering assets and instructing the Publisher server to embed them in the executable paper as IFrame links. With the aid of browser mechanisms, the executable paper may periodically query the assets which are listed as part of the publication, replacing placeholder content with proper visualization payload, as results become available.

## 6. Summary and Conclusions

We believe that our experience with issues pertaining to e-Science and the tools developed in support of computerized research leave us uniquely poised to meet the goals set forth in Section 2. Much of the functionality on which Collage bases is already present in the GridSpace platform [11, 12]; therefore we consider it a good starting point towards the development of a targeted solution which would be easy to interact with and to maintain. At the time of preparation of this paper a working demo of Collage was being developed, based on actual research papers,

showing how the solutions outlined above are applied in practice and describing in detail the tools available to both the author of an executable paper and to its readers.

While work on implementing the Collage Environment currently focuses on its core engine and data/computation access features, as well as on preparing suitable demonstrations, in the future we intend to progress to further development of end-user interfaces, particularly the Authoring UI and the Presentation UI (discussed in Section 5), thus ensuring that Collage becomes a marketable solution.

## Acknowledgements

The authors wish to thank the PL-Grid project, developed with support of the European Union within the European Regional Development Fund program no. POIG.02.03.00-00-007/08-00. The authors would also like to thank Tomasz Gubała of ACC CYFRONET AGH for his valuable contributions.

## References

1. T. K. Attwood, D. B. Kell, P. McDermott, J. Marsh, S. R. Pettifer, and D. Thorne, "Utopia Documents: linking scholarly literature with research data". In Proceedings of 9th European Conference on Computational Biology, Ghent, Belgium, Sep 2010.

2. G. Allen, F. Löffler, T. Radke, E. Schnetter and E. Seidel, "Integrating Web 2.0 technologies with scientific simulation codes for real-time collaboration". CLUSTER 2009: 1-10.

3. T. O'Reilly, *What is Web 2.0. Design patterns and business models for the next generation of software*. O'Reilly Media, 2005. www.oreillynet.com/pub/a/oreilly/tim/news/2005/09/30/what-is-web-20.html

4. D. DeRoure (2010), "e-Science and the Web". IEEE Computer, 43(5):90-93.

5. D. Giustini, "How Web 2.0 is changing medicine". BMJ 2006; 333, pp. 1283-1284.

6. V. Gewin (2008), "The new networking nexus", Nature 451, 1024-1025.

7. D. Crotty, Why Web 2.0 is Failing in Biology, Cold Spring Harbor Protocols, Feb 14, 2008; http://www.cshblogs.org/chsprotocols/2008/12/14/why-web-20-is-failing-in-biology.

8. D. DeRoure, C. Goble and R. Stevens, "Designing the myExperiment Virtual Research Environment for the Social Sharing of Workflows." e-science 2007 – Third IEEE International Conference on e-Science and Grid Computing, 2007, Bangalore, India; 10-13 December 2007; pp. 603-610.

9. A. Jentzsch, B. Andersson, O. Hassanzadeh, S. Stephens and C. Bizer (2009), "Enabling Tailored Therapeutics with Linked Data, Proceedings Linked Data on the Web" (LDOW2009), April 20th, 2009, Madrid, Spain.

10. M. Hall, D. Padua, K. Pingali (2009), "Compiler Research: The Next 50 Years: the next 50 years", Communications of the ACM, Vol. 52 No. 2, Pages 60-67 Communications - February 2009 - Research Highlights (Page 67).

11. M. Bubak, M. Malawski, T. Gubala, M. Kasztelnik, P. Nowakowski, D. Harezlak, T. Bartynski, J. Kocot, E. Ciepiela, W. Funika, D. Krol, B. Balis, M. Assel, and A. Tirado-Ramos: Virtual Laboratory for Collaborative Applications, In: M. Cannataro (Ed.) *Handbook of Research on Computational Grid Technologies for Life Sciences, Biomedicine and Healthcare*, Information Science Reference, 2009, IGI Global.

12. E. Ciepiela, D. Harezlak, J. Kocot, T. Bartynski, M. Kasztelnik, P. Nowakowski, T. Gubała, M. Malawski and M. Bubak, "Exploratory Programming in the Virtual Laboratory", in Proceedings of the International Multiconference on Computer Science and Information Technology pp. 621–628. Best paper award.

13. M. Malawski, T. Bartynski, and M. Bubak, "Invocation of operations from script-based grid applications", Future Generation Computer Systems, Volume 26, Issue 1, January 2010, Pages 138-146. Available: http://dx.doi.org/10.1016/j.future.2009.05.012.

14. J. T. Dudley and J. Atul, "In silico research in the era of cloud computing", Nature Biotechnology, vol. 28, no. 11, 1181-1185 (2010).