NCBI Bookshelf. A service of the National Library of Medicine, National Institutes of Health.

Journal Article Tag Suite Conference (JATS-Con) Proceedings [Internet]. Bethesda (MD): National Center for Biotechnology Information (US); 2017.

Introducing Texture: An Open Source WYSIWYG Javascript Editor for JATS

Alex Garnett,[1] Michael Aufreiter,[2] Oliver Buchtala,[2] and Juan Pablo Alperin[1].

[1 1] Public Knowledge Project, Simon Fraser University
[2 2] Substance Software GmbH

Texture is a WYSIWYG editor app that allows users to turn raw content into structured content, and add as much semantic information as needed for the production of scientific publications. Texture is open source software built on top of Substance (http://substance.io), an advanced Javascript content authoring library. While the Substance library is format agnostic, the Texture editor uses JATS XML as a native exchange format. The Substance library that Texture is built on already supports real-time collaborative authoring, and the easy-to-use WYSIWYG interface would make Texture an attractive alternative to Google Docs. For some editors, the interface could be toggled to more closely resemble a professional XML suite, allowing a user to pop out a raw attribute editor for any given element. Texture-authored documents could then be brought into the journal management system directly, skipping the conversion step, and move straight into a document-centric publishing workflow.

Introduction

Microsoft Word's dominance as an authoring tool creates substantial inefficiencies in the scholarly authoring ecosystem. Many journals and journal management platforms are designed around uploading and downloading incrementally updated drafts of Word manuscripts, creating a difficult-to-manage ecosystem of individual change-tracked files and annotated PDFs. For most end users, there is no sufficiently easy to use or widely accepted alternative to this; although some researchers author in LaTeX or Markdown which can at least be *diff*ed line by line, getting most authors away from Word has been a fairly fruitless task so far. We believe that this is mostly due to ease of use. Yet, when it comes to publishing, the scholarly publishing industry has (mostly) settled on a structured format —JATS XML. This disconnect between the tools and formats used for authoring and the formats required for publishing has meant that, for several decades now, manuscripts received from authors will need to be entirely XML-typeset by publishers at considerable expense (Eve, 2016).

The role of the XML typesetting is so crucial that there are several companies that specialize in just this task. Right now, most academic journals with adequate budgets have outstanding contracts with third-party typesetting shops, resulting in a process that is highly professionalized, but also highly impractical and unavailable to many smaller journals. This work, requiring specialized labour and more person-hours than can be easily justified, has been one of the main differentiating factors between well-resourced journals and those on shoestring budgets. This is especially unfortunate, as having an XML-first workflow that allows journals to produce multiple formats from a single source would be especially beneficial to such low-budget journals.

Efforts to bring XML markup, in particular JATS, to a broader group of journals have continued to appear in recent years. These efforts, in the form of automatic Word-to-XML converters include the Public Knowledge Project's Open Typsetting Stack (Garnett, Alperin & Willinsky, 2015), Martin Eve's meTypeset (Eve, 2015) redalyc.org's Marcalyc (http://marcalyc.redalyc.org/) However, none of these tools, and, to the best of our knowledge, none of the proprietary tools used by those offering professional XML markup services, produce production-quality JATS without user mediation, and still require manual or semi-manual intervention. From the perspective of publishers who are currently not producing JATS, this last step requires having XML expertise on hand, or the resources to acquire them.

Introducing Texture

Texture is a WYSIWYG editor app that allows users to turn raw content into structured content, and add as much semantic information as needed for the production of scientific publications. The primary goal of Texture is to remove this requirement for XML expertise by providing a solution for publishers to bring accepted papers to production more efficiently.

Texture reads and produces valid JATS files. This allows Texture to work seamlessly in existing publishing workflows. For instance, Texture can take the output of a Word to JATS converter and enhance the content until it is

ready to be published, integrating into current toolchains. Texture produces normalized JATS, applying a nearly* lossless conversion and following strict rules and best practices (JATS4R).

With Texture, we propose an **iterative refinement** process. Starting from the best available output from a **automated conversion process**, different steps are taken to turn the article into a structured, semantically tagged article, augmented with metadata. For example, the author of a document could be represented as plaintext under the title. As a refinement, this piece of text can be tagged as *author*. On the next level, this can be replaced by a structured version, with <family-name>, <given-name>, etc, and even more metadata such as an ORCID, for instance. The same would be true for citations. Similarly, references can be initially tagged as *mixed citations,* but at a later stage, Texture can connect to third party APIs (e.g. DataCite, CrossRef) and convert them into fully structured element citations.

Current state of development

Texture is open source software built on top of *Substance* (http://substance.io), an advanced Javascript content authoring library. While the Substance library is format agnostic, the Texture editor uses JATS XML as a native exchange format.

Texture ships with a fairly complete set of document elements already implemented in the user interface, including support for tables, figures, citations, equations, and so on. Rendering of each of these elements is implemented on a modular basis, so that any supported JATS element can be added as a local customization and merged back into the upstream codebase. As-yet unimplemented elements can still be added or edited by an end user in raw XML as needed.

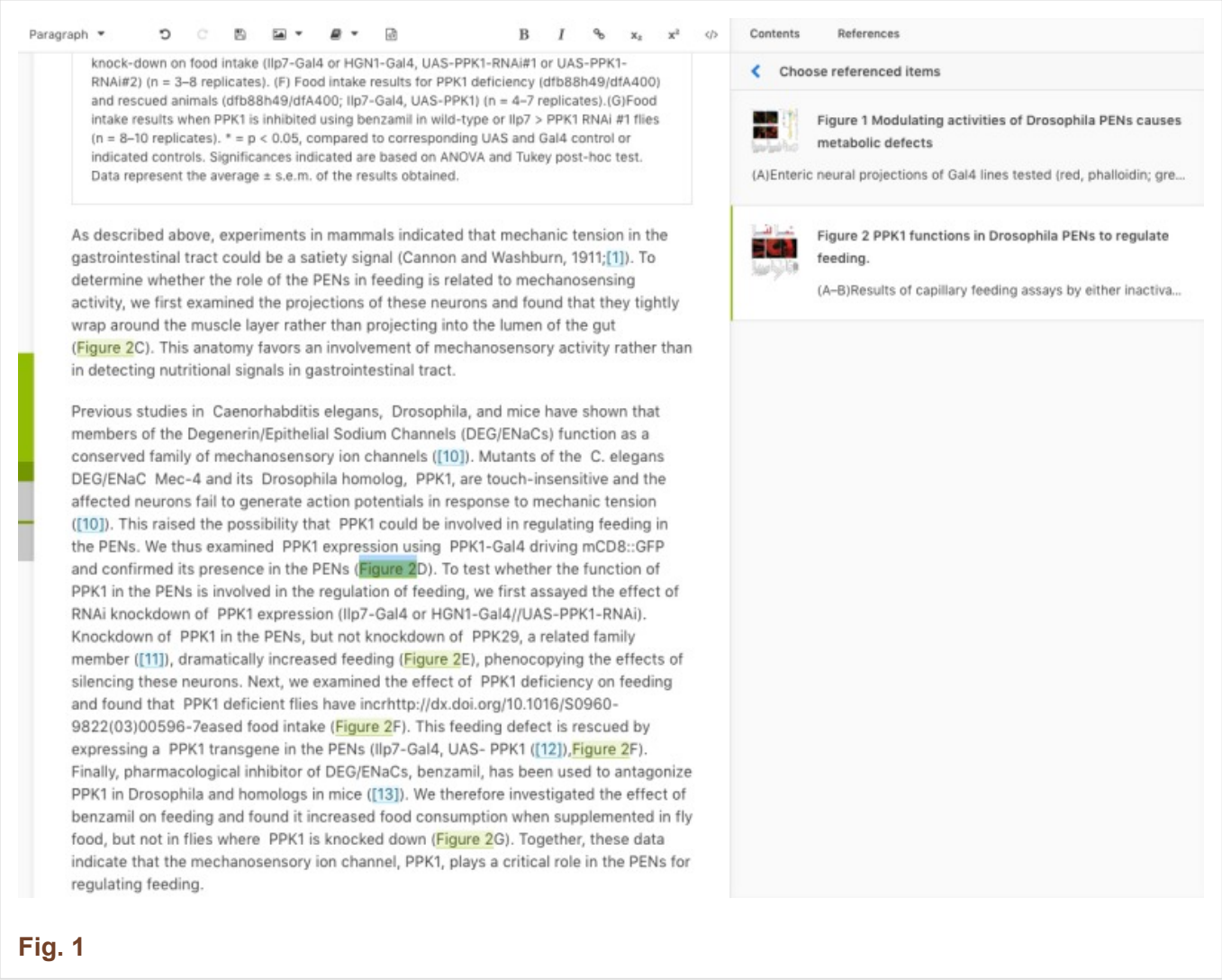

Fig. 1

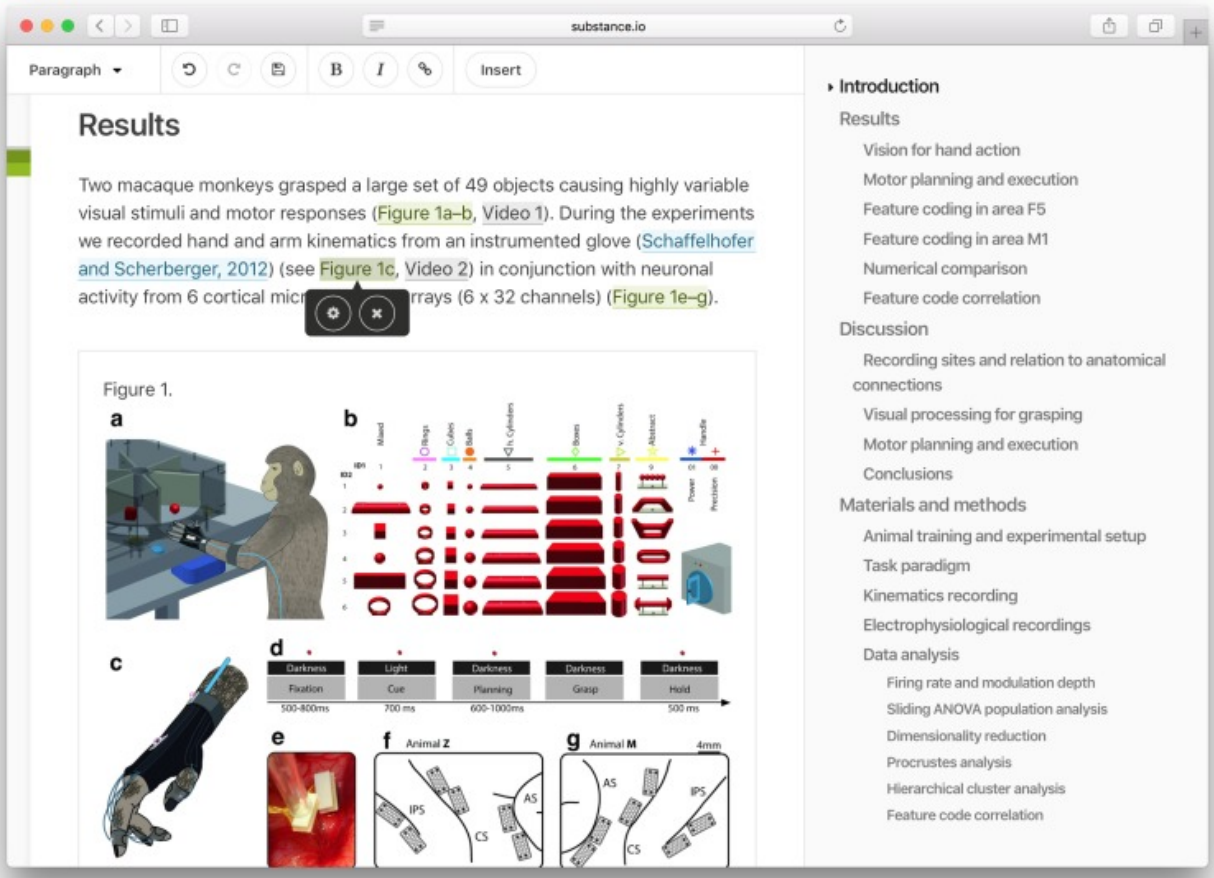

Fig. 2

With Texture, editors can use an interface similar to a classic word processor. This makes the difficult job of manually crafting XML (such as in the *Oxygen* software) unnecessary for most workflows and provides a more natural approach to content curation. The user interface can be extended to reflect specialized viewing modes; Texture can be customized for different user roles to provide different levels of complexity. For some editors, the interface could be toggled to more closely resemble a professional XML suite, allowing a user to pop out a raw attribute editor for any given element.

Texture is designed to be embeddable. It provides a set of components that can be configured to fit different integration scenarios. For example, Texture will be soon available as a stand-alone desktop application (using Electron), ready to read and write JATS XML files from local filesystems. In contrast, Open Journal Systems (OJS) will integrate Texture into their web-based journal management software, where documents are read from and saved to a database.

Texture is different from other web editors as it maintains a local state of the complete document in memory. This means the editor is always responsive and does not require a permanent internet connection while editing. This design allows for both offline editing and real time collaboration, where each collaborator maintains a local copy and only synchronizes changes with the other peers. If content is enhanced through online-services (e.g. DataCite, CrossRef) Texture stores a snapshot of the current data and a URI to the canonical resource so it can be automatically refreshed. Per the JATS standard, documents are fully self-contained. Complex XML stores and versioning are generally not in the scope of Texture, but users can provide their own backends. The editor can always be asked for a snapshot of the current JATS serialisation, or a listener can be registered to be called on each change, allowing for storing the complete editing history, on keystroke level if needed.

Unlike other HTML/XML editors, Texture keeps annotations separate from the text internally. A reference to the text element being annotated is stored along with the character offset (start, end) of the annotation (e.g. bold). Using this flat model, Texture can effectively handle overlapping of annotations, without ending up with very deep nesting or lots of fragments in the resulting XML.

Texture provides lossless conversion in that it recreates the schema of JATS in-memory, by utilising the XML schema. Most elements from the <body> and <article-meta> will then be editable out of the box. Some JATS elements, such as the <contrib> element, allow many different ways of tagging, which would require many different user interfaces for each tagging scenario. With Texture, we want to work towards flexible tagging conventions that can power the best possible user interfaces. Users of Texture (the consortium members) are defining those conventions collaboratively. According to agreed conventions Texture can then normalise certain elements providing extremely consistent tagging, which would be hard to achieve in a manual typesetting scenario.

Current and future use cases

At this initial stage, Texture is being developed to **be used by a production team** seeking to take the author's final version of a manuscript and produce production quality JATS for publishing purposes. In this case, a conversion tool is used to make an initial attempt at converting the peer-reviewed manuscript from Word or PDF (or another input format) to a(n imperfect) JATS article. This JATS would then be loaded into Texture, and, with the aid of visual cues from the Texture UI, a user would make iterative improvements to the underlying JATS. It is anticipated that Texture would be used to add additional detail (such as author affiliation information), correct conversion errors (such as misclassified information), or to provide additional granularity to existing tags (such as breaking up a <mixed-reference> into its constituent parts[**]).

In the future, however, Texture could be brought into the workflow at an earlier stage. Instead of providing Texture at the end of the editorial process, Texture could **be used within the review or editing workflows**. The automated conversion could be performed on the author's original manuscript, allowing the JATS XML, with Texture as a front-end, to be exclusively used for the review, copyediting, and proofs. While many journals and journal management platforms are designed around the uploading and downloading incrementally updated drafts of Microsoft Word manuscripts, there is an emerging trend toward document centric workflows (e.g., the Collaborative Knowledge Foundation platforms, eLife's Continuum, and PLOS' Aperta) which use an HTML-rendered version of the manuscript for authors, reviewers, and journal staff to interact with. Texture could provide this HTML rendition (as a Web editor, it provides an HTML rendition of the underlying JATS XML). This approach works well with Texture's iterative improvement philosophy; at each stage in the workflow, authors and journal staff interact with the document and have the opportunity to improve the document towards its publication-ready final form.

Finally, yet another iteration of Texture could **be used by authors** even earlier in the process. In this case, instead of serving as an editor, Texture would be a from-scratch authoring tool, allowing authors to create production-ready XML documents from the beginning. The Substance library that Texture is built on already supports real-time collaborative authoring, and the easy-to-use WYSIWYG interface would make Texture an attractive alternative to Google Docs. Texture-authored documents could then be brought into the journal management system directly, skipping the conversion step, and move straight into a document-centric publishing workflow.

Proof-of-concept implementation

The Public Knowledge Project has continued to develop their *Open Typesetting Stack* (OTS) (Garnett, Willinsky, and Alperin, 2015) application for automatically transforming Word or PDF articles into JATS XML. We currently have an alpha plugin for integrating OTS into our Open Journal Systems publishing platform; this plugin includes Texture. Our solution, using the Open Typesetting Stack and Texture, aims to address the impracticalities of trying to "reverse-engineer" an author's work in Word while still supporting a polished, professional typesetting workflow.

The Open Typesetting Stack provides a toolchain for automatically converting Word or PDF documents to National Library of Medicine JATS XML. More than a dozen open source libraries are used to produce this result; parsing accuracy is better than any other known tool, at around 80% averaged across all document elements on a real-world corpus, but still generally not good enough for production-quality output by itself. Improving our automated parsing quality beyond 80% has, expectedly, slowed due to the many different ways to non-syntactically format a Word or PDF article, necessitating a supervised solution to finish the job. Unfortunately, because many publisher workflows are fairly idiosyncratic, it is not necessarily helpful to hand off this 80% accurate result to a typesetter on the assumption that they have specialized XML editing software to finish the job. Thus closing this 80% gap is still an issue.

When using the Open Typesetting Stack plugin in Open Journal Systems, authors and editors have the option to hand off any uploaded Word or PDF documents to OTS, whereupon they are replaced with OTS' JATS output for the

remainder of the publishing workflow. As OTS' JATS output is generally not good enough to go directly to publication, users of OJS have the ability to continue editing directly in a Texture instance that is accessible from the OJS interface. The goal is to get authors and editors away from the original Word manuscript as early as possible in the publishing process, so that carrying forward changes to the eventual layout draft entails minimal duplicated effort. When articles are ready for publication, they can be resubmitted to OTS through another plugin hook, and it will return PDF and ePub transformations to OJS to be published alongside the XML.*** More information, and a video demonstration, is available at https://pkp.sfu.ca/open-typesetting-stack/.

Outlook

As far as "getting away from Word" is concerned, adoption of Texture by editors, reviewers, and late-stage collaborators should drive uptake, as will integration into journal publishing platforms such as Open Journal Systems. While in the short term Texture will focus on publishers' use-cases, we are exploring future use cases for Texture as an authoring interface. For example, peer-review could be realized entirely in Texture, including communication between peers (reviewers, authors, editors) which currently is fragmented over different channels such as Email or Messengers.

The eventual goal is to use Texture as an integral building block in modern and customised end-to-end publishing systems, where the document sits in the center (single-source) and is edited by all involved parties (author, editor, reviewer) in a collaborative way.

Open collaborative roadmap

Texture is developed by the Substance Consortium, which was established by the Public Knowledge Project (PKP), the Collaborative Knowledge Foundation (CoKo), the Scientific Electronic Library Online (SciELO), Érudit, and, of course, the Substance team itself. This consortium welcomes additional members and is intended to provide a steering committee and coordinating body to support the development of Texture and the underlying Substance library.

Development of Texture will be prioritized to meet the needs of the consortium members and the broader scholarly publishing community, with clear avenues for community input and discussion.

References

1. Eve Martin. Of LaTeX and labour. https://www.martineve.com/2016/09/14/of-latex-and-labour/ 2016.
2. Garnett Alex, Alperin Juan Pablo, Willinsky John. The Public Knowledge Project XML Publishing Service and meTypeset: Don’t call it "Yet Another Word-to-JATS Conversion Kit.". https://www.ncbi.nlm.nih.gov/books/NBK279666/ 2015.
3. Eve Martin. Building a real XML-first (XML-in) workflow for scholarly typesetting. https://www.martineve.com/2015/07/20/building-a-real-xml-first-workflow-for-scholarly-typesetting/ 2015.

Footnotes

* Technically, because we're serializing and de-serializing the document on input and export, we aren't targeting a *fully* lossless conversion -- some front matter elements will be naturally "smoothed" from one allowable JATS convention to another (e.g. <aff> within <author-group>), because we only support generating one convention through our interface -- but data should never be lost.

** Although, as much as possible, Texture tries to prioritize the use of DOI metadata lookup over direct citation editing.

*** OJS 3 also ships with the open source *Lens Viewer*, the product of an earlier collaboration between Substance Software GmbH and the journal eLife, which provides a dynamic in-browser view of JATS XML, eliminating the need for a static HTML version of the finished article.

Bookshelf ID: NBK425544