Dynaslides

Francisco Campaniço
francisco.campanico@tecnico.ulisboa.pt

Instituto Superior Técnico, Lisboa, Portugal

December 2020

Abstract

In order to display information to a relatively large audience, we often make use of multimedia presentations. However, creating these presentation documents requires the use of specific applications based on outdated presentation techniques and that do not take full advantage of contemporary technologies. Most of these tools are only meant to create documents that showcase information and which do not offer the possibility for the audience or the presenter to interact with the content during the actual presentation. Therefore, DynaSlides proposes a solution that relies on a standard, established, open-source framework to provide several interactive and dynamic features without needing to exit the presentation environment. It also features an extensible plugin structure, capable of handling additional interactive components more easily, excluding the current functionalities. Moreover, usability tests are performed on the initial prototype to make sure it meets the desired requirements.
Keywords: Dynamic; Slideware; Presentations; Powerpoint; Hyperpresentations; Multimedia; Interactivity;

1. Introduction

Sharing knowledge and communicating ideas is part of what makes us human. From narrating a simple tale to describing a solution for a complex problem, our society thrives when information is analyzed, distributed, and stored for future generations. There are many different systems created to help diffuse and preserve this valuable information, and there is a contemporary technique, in particular, we are going to analyze carefully, often referred to as a multimedia presentation.

One major dilemma regarding this subject arises when a presenter wants to interact with an audience and requires an external application. Stopping the presentation and using a different tool to showcase an interactive feature might interrupt its flow and change how the audience perceives the information displayed. It is also inconvenient for the presenter to switch between applications since it often requires preparing before the presentation starts and dragging windows into the right view, leading to unexpected problems. Considering that nowadays, interaction with the audience becomes more and more of a necessity and that most presentation environments have access to the internet, a presentation application should offer these features while being practical and intuitive.

Another glaring problem with contemporary presentation environments is the lack of interaction between the audience and the presenter, and the one that exists happens via verbal communication and feedback, such as questions or commentaries. If the presentation utilizes a slideware application in a technological environment with access to the internet, there should be an easy and intuitive approach to receive feedback from the audience. Not only could the presenter offer an immediate response based on the reactions, but he could also change future presentations based on that same feedback.

The relevance for this type of application is also exacerbated by the current COVID-19 pandemic we currently face. Due to the increasingly higher usage of online lecture and presentation environments, giving oral feedback became even more difficult than before. To have an application that can gather feedback information without interrupting the flow of the presentation appears to be even in more demand than ever before.

The goal of this project is to **design a dynamic presentation tool that allows the inclusion of rich multimedia content while fostering interaction between the presenter and the audience**, and without compromising relevant traits already acquired by contemporary presentation applications, such as how difficult it is to produce a presentation. We intend our developed application to be a proof of concept of an interactive presentation environment that shows the possibility of increasing the interactivity of slide-based presentations with-

out losing the possibility of using an already established presentation application.

2. Related Work

To solve the aforementioned problems, we start by finding the current practices in multimedia presentations and use them to draft a suitable solution. The following material is, therefore, categorized into three different sections to showcase similarities between different papers:

- **Navigation** - Different styles of presenting and how the information reaches the audience. Navigation techniques on presentation slideware have not changed must in the past. The industry standard, Microsoft PowerPoint, uses a linear sequence of slides to convey a message and is often lacking at telling the complete story since slides are envisioned one by one without requiring a plan or overview to connect them. In particular, PowerPoint has been associated with having a ”dubious reputation” [2], turning the vast number of users ”into bullet-point dandies” [6], criticized for encouraging oversimplification of complex information [10][13], and described by experts as a negative contribute for ”dialogue, interaction, and thoughtful consideration of ideas” [3].

  Some existing applications already undertake this issue by using animations and zoom capabilities as a tool to escape traditional linear navigation. However, this is only one approach for non-linear navigation and refers specifically to presentations' visual characteristics.

  HyperSlides [4], which offers another look to this problem, is a tool that started by conducting a grounded theory to understand better the idea of presenting with a slideware application and discovering the potential for a more dynamic approach to prototyping slide presentations.

  NextSlidePlease [11], a support presentation tool, pursues the navigation and authoring problem of standard presentation tools by creating an editing environment that supports several features.

  Some presentation tools like MindXpres [9], a content-driven cross-media platform with a modular architecture and a plugin mechanism, try to solve many issues in today's standard presentations such as linear navigation, separation of content and presentation, extensivity and interactivity.

  Multipresenter [5] is yet another application that offers a different look into the navigation problem. It states that most slideware tools are focused on presentation content authoring and do not support the dynamic aspect of presentations.

- **Interactivity** - Different approaches to interacting and conveying information to a target audience. Interactivity is one of the most important features in this report since the main goal is to give the end-user the ability to have real-time interaction with the audience and adjust the presentation to the audience's needs. Some of the following solutions already mentioned the need for this interactivity with the audience.

  SlideDeck.js [12] is a multimedia presentation framework that includes a custom markup language with its respective macro processor, an interaction engine, and a course content browser. A custom markup language ensures the user follows a strict set of rules. A macro processor can correctly generate the respective HTML tags and make the desired visualization, making the tool have a certain level of abstraction to avoid the user creating the HTML visualization. An interactive engine creates the basic interaction between slides to ensure the application works as a standard presentation tool.

  There is an article that aims at enhancing programming lectures using interactive web-based lecture slides [1] by conducting a case study using a well-defined web presentation framework called **Reveal.js** [1]. It allows for server technology to be used, giving more options when implementing interactive content and allowing more flexibility by using web-based languages like HTML, CSS, and Javascript.

  The paper about the MindXpress interactive data plugin [7] also provides great insight on another form of interactivity, which relies on different interpretations for a given dataset. It achieves that by using a menu designed to change between different views while the presenter conveys the correlated narrative. It also allows the audience to further understand the dataset by viewing different variants and extracting as much from it as possible.

- **Authoring** - different ways of authoring documents/presentations (how to create and edit presentations) and how several existing techniques can lead to better authoring environments. Fast and reliable authoring is something a creator always values in a presentation

[1] https://revealjs.com/

tool since it allows for better time management and an easier replication of previous work.

Some other tools today already take advantage of the aforementioned features, such as HyperSlides [4], which links slides to follow a given storyline or Microsoft PowerPoint, which includes a presentation view with the time since the beginning of the presentation.

NextSlidePlease [11] touches on the same subject with its two lasting points, **Content reuse and Path suggestion** and **Time management**. Having the ability to generate paths and connections between slides automatically and control the presentation time are adequate contributions for a more efficient and planned authoring technique.

HyperSlides [4] also tries to reduce the authoring time by giving an idea, which we discussed in the navigation section, referred to as *Planning with Points*. This subject resurfaces because it forces the creator to produce a storyline before committing to the actual slides, saving time while authoring.

Some aforementioned applications such as SlideDeck.js [12], or MindXpres [9] use custom formats as a way to generate presentations but do not refer to the automatic translation of previous content made on other platforms.

We can observe the state of the art of alternative presentation mechanisms by analyzing the aforementioned concepts and that there is an increasing need for an application more agile and interactive than the field of presentation software currently presents.

HyperSlides is very thorough when it comes to the navigation and authoring topics, as it aims to escape linear navigation and improve the authoring experience by requiring a top-level storyline. However, the topic of interactivity, which happens to be the main focus of this project. NextSlidesPlease [11] also focus more on the first two topics by showing other ideas to approach them, such as using mental concepts or using several whiteboards in a presentation, but again not discussing how to interact more with the target audience.

SlideDeck.js [12] shares some key objectives with what we want to develop, such as the conversion of some arbitrary language to HTML or interaction with code editing and display through the use of the HTML tag *iframe*. It does so with only basic functionalities, and it does not refer to other key components, such as dynamic navigation (with the standard slide presentation format), full conversion from other presentation tools, or interactive data visualization.

MindXpres [9] appears to be the tool that offers the most to tackle the aforementioned subjects, although not presenting a public release to explore further the possibilities stated in the respective articles. The two aforementioned plugins for this platform [8] [7] are great examples of concepts when interacting with the audience and show that a more interactive focused approach is something desired by the industry.

On the other hand, **Reveal.js** provides an open-source framework with some solutions for the problems related to linear navigation and client-server based presentations, and the possibility to extend the current functionality with the use of plugins, in the same way as the MindXpres [9] application.

3. Implementation

To design a suitable solution, we had to define the objectives based on the problem's description. In the following items, we are going to categorize them with the respective description while ordering them by relevance:

- **Translatable** - Since developing a user interface from scratch would require a workload that does not match the one from this project and also happens to be outside of its main purpose, utilizing a well-established interface to structure a given presentation seems like the logical step to take. It will make the learning curve of the new application more accessible while providing the possibility of reusing previous content with added functionality without having to change it entirely to a new format. Furthermore, finding a structure that does not diverge from current presentation tools and provides the user with common up-to-date techniques while having interactive functionality is also a request feature for this platform. The end-user must not have to learn a completely new environment to use the interactive features, but instead, use them as new possibilities in standard presentation formats. Therefore, the user must use a standard presentation application and transform it into a new format without learning entirely how the new format works.

- **Interactable** - Being able to interact with the audience, whether by gathering the answers of a question without escaping the presentation environment, receiving instant feedback, or utilizing external resources inside the presentation view, is crucial for the desired application. It must help the user interact with the target audience without restricting or stopping the presentation's flow, i.e., without forcing the presenter to exit the presentation view. It must

also guarantee the user can use the different forms of interaction without being dependent on external factors, that is, the presentation application must display the content if it does not require an external connection to the internet;

Then we needed to find a **base/starting point** for our project. We chose to take advantage of the presentation platform **Reveal.js**. The following statements are major factors for choosing the aforementioned framework:

- It is developed for the web (HTML, Javascript, and CSS), which allows for easy usage on any compatible browser, and integration of new functionality also becomes accessible;
- It has grown in popularity over recent years, gaining a large community while becoming the standard in web-based presentations. It is crucial to ensure the impact and longevity of the proposed solution;
- It offers the possibility for multiplexing, allowing the audience to view the presentation slides the creator is controlling on their own mobile devices such as a portable computer, phones, tablets, or any device with an internet connection and access to a web browser. It employs a Socket.io server (real-time application framework for bidirectional and event-based communication) and is going to be useful when implementing features of audience interaction;
- Extra functionality is possible due to a plugin architecture, which highly enhances the extensibility possibilities;
- It is registered under the MIT license, allowing for more flexibility when adding new functionality.

All the above statements establish the proposed framework as the best candidate for the solution's core.

Afterwards, we decide to design our **architecture**. The new features are joined with the core application by designing a specific plugin for each feature. A **Reveal.js** presentation works like a typical front-end application, where the user has the main HTML file, the corresponding Javascript and CSS files. To manage other useful features more efficiently, such as connections between servers and clients, SCSS compilers, and task runners, We will use Node.js [2] in these specific situations.

Furthermore, **Reveal.js** offers the possibility of adding plugins to increase its functionality, and to accomplish that, it registers the plugins before initializing a presentation. They are be grouped in a bundle alongside the translation module (DynaSlides project), which is responsible for registering all the plugins and handle all the necessary interactions between the foundation, the translation module, and the respective plugins.

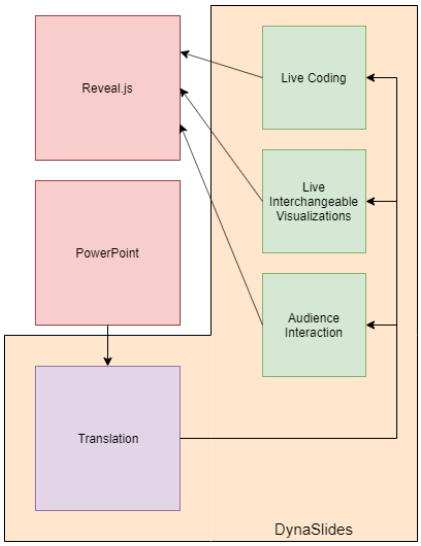

Figure 1: Architecture structure for the DynaSlides bundle

After finishing the architecture, we decided to facilitate the user of the dynaslides application with two different versions:

- **Website version** - This version is intended for the user with no experience with a command-line interface and wants to translate the presentation more easily. It is built like a normal website, i.e., it has a back-end/front-end configuration and a server that can connect with the audiences' devices.

  The back-end takes care of all the features mentioned in the architecture, such as the translation and the plugins present in the translation, and then sends it to the front-end to be downloaded as a .zip file.

  The front-end gives the client an easier way to translate the presentation, with the different required options.

  Finally, it also offers server code that needs to be set in a publicly accessible environment, such as a public endpoint, to provide the live feedback that the audience interaction plugin requires, as we will see later on.

- **Command line version** - This version is for users acquainted with a command-line interface and prefers to handle the different available options. It has the same functionalities

[2] https://nodejs.org/en/about/

as the website version and is also explained in the application's reference guide.

The command-line interface uses node.js to run the application that is developed in javascript.

The application will try to find the Microsoft's Powerpoint file, then the data files associated with the live visualization plugin, and finally the options. For that reason, if no Microsoft's Powerpoint file is made available or a data file is missing, the application will launch an error and stop the execution.

Moving forward, we had to find a way to **translate** a Microsoft's Powerpoint presentation into a Reveal.js presentation. Therefore, the translation module started with an already working solution[3] that translated Microsoft's Powerpoint presentation to a new HTML format, which then uses the **Reveal.js** presentation framework.

We first started by changing the application so it would work with node.js. The previous version would take advantage of *web workers* which are a simple implementation for web content to run scripts in background threads. Since these workers are not required and are not the best solution in node.js, we used the node.js package system to implement the same functionality it had while working on the browser. This change also enabled us to have a command-line version and a more organized version of the application since it now uses the node package manager or npm for short.

Later on, we started by reviewing what was already done and the current problems. Firstly, we began by fixing the hierarchy of the information in each slide. A slide is made from the combination of three components forming the said hierarchy. At the bottom, we have the master slide, which the slide is based upon. It encapsulates default properties of the slide, such as the positions, font styles, background, title, body, and footer. In the middle, we have the slide layout applied to the slide in question, which contains information that overrides the one present in the master slide. Finally, we have the slide itself that contains information not already specified in the master slide or the layout. This information is only used by one particular slide.

After that, we focused on implementing each element's features in HTML. Since there are many elements provided by the original presentation, we worked on the most important features and tried to make them as close as possible in the new format:

- **Position** - The elements' position is arranged by taking the slide size and the most common computer monitor DPI (dots per inch) to translate the position given in English Metric Units or EMUs to the respective pixel size. The starting application had this system implemented correctly. However, it had to be improved by ensuring the sizes, rotation, margins, and padding of elements were the same as those displayed by the original presentation. To do this, the hierarchy of the elements had to be used to get the correct properties.

- **Shapes** - The starting application already had core shapes implemented, such as rectangles, round rectangles, ellipses, connectors, and basic arrows. However, it lacked the ability to translate custom shapes into SVG elements, such as background themes. It also lacked the ability to recreate some properties of the shapes, such as borders and colors. To do this, the application converts the custom shape elements present in the .xml file of the slide or the respective layout/master components to inline SVG elements in the new format.

- **Color** - Colors were a big problem to translate since the starting application did not consider the same hierarchy stated earlier. Another problem was that colors could have effects embedded in them, such as saturation, luminosity, alpha, and more. We implemented the effects as referenced in the *Office Open XML* presentation reference guide [4] by reproducing them in the CSS of the new format.

- **Images** - The main problem with images was that the only translated element in the starting solution was the image itself. It did not consider the effects images can have in the original presentation. We implemented the basic effects, such as duotone, tint, luminosity, and cropping, according to their description in the *Office Open XML* presentation reference guide 4.

- **Themes** - Themes had a problem related to their respective backgrounds since the starting application did not consider the custom backgrounds that Microsoft's Powerpoint offers. We implemented a way to translate background themes configured as custom shapes that can be converted to inline SVG elements.

Now, we needed to implement the **plugins**. The plugin structure offers basic functionality common to all plugins, and then that functionality is complemented by the plugins. Each plugin works by using

[3] https://github.com/g21589/PPTX2HTML

[4] http://officeopenxml.com/anatomyofOOXML-pptx.php

a translated text box from Microsoft's Powerpoint while taking advantage of a property common to all objects in the original presentation called `alt text`.

The `alt text` property is perfect for choosing which text box belongs to a given plugin since it is always present at the slide level of the translation and it is not visible on the actual slide during the presentation. This means we can use that field to identify the plugin and gives us the ability to control the plugins' inputs.

Therefore, we created a function that reads every alt text property available in each text box of a given slide. If it finds a custom set of characters predefined by us (in this case, %%), it will know that the text box is used for a particular plugin. This custom set of characters has custom commands inside, and they are explained in each plugin, although we represent them as so:

```
%%<custom command>(<options>)%%
```

This allows the presenter to create a simple text box, and it's content to recreate the interactive plugins as we see fit. We can even use the text from inside the text box and the custom command to create even more functionality, such as in the live coding plugin.

Some plugins **share basic functionality**. The first feature, which is common to all the plugins, uses the custom command system represented above to allow the presenter to show a website of his choice while on the presentation.

Another feature shared among all plugins is the ability to have audience feedback. This feature will impact how the file presentation document is constructed. The application will create two directories instead of one in the presentation's final .zip file. One directory will correspond to the audience's view and enables them to interact with their own devices, i.e., the client version. The other directory will correspond to the presenter's presentation, shown on a big screen for all of the audience, i.e., the master version.

For the particular plugin functionality, we developed the following:

- **Live coding plugin** - The live coding plugin consists of displaying, executing, and editing code inside the presentation view.

  The plugin works by gathering the code as text inside the text box and using the same system shared by all the plugins. The reason why we chose to use the text inside the text box, in this case, is because the presenter probably wants the keep the text formatting the same for the code. And if an error occurs and the application can not translate the code, the presenter will always have the code available to show in the original presentation.

  The translated code is then placed in a `div`, in conjunction with a button to execute the respective code and with another `div` that will show the results of the execution.

  The HTML `div` that contains the code uses a **Reveal.js plugin** to highlight the code, which will make the code much easier to perceive. This container also provides the ability to alter the coding during the presentation.

  The execution button uses a Javascript function called `eval`, which takes the code and runs it, like the console present in most browsers. Since this function can be dangerous if the user does not know or understand the code's output, a message will appear and ask the presenter for confirmation before running the code.

  Finally, the last HTML `div` of the plugin will showcase the result from pressing the button.

- **Live visualization plugin** - The plugin consists of creating an interface where the user can change how the data is visualized. The presenter can add, edit, or remove data while remaining in the presentation environment. The process starts by providing a path to the data file with a predefined format (.cvs or .json). To do formatting in the new slide, the plugin uses the same shared method as the other plugins.

  Due to the complexity and amount of different datasets available, we decided that we would only use this simple format to show it's possible to interact with visualizations while performing the presentation. Any other datasets that are not two connected dimensions and not in the .cvs or .json format will generate an error.

  During the presentation, the presenter will have access to a menu that offers the possibility of choosing between a limited set of different visualizations according to the data provided, much like the one discussed in the related work. The presenter will also have access to another section that allows adding, editing, or removing data during the presentation.

- **Live audience interaction** - This plugin offers a live question functionality to interact with the audience. It allows the presenter to create a yes or no question to a given audience, where they can consequentially vote on their

own devices and see the results in the respective slide.

In this case, the master presentation will transform the text box from the original presentation into a bar plot with two bars and with the respective question. One shows the count for the number of positive answers, and the other shows the count for the negative answers. This is done using the aforementioned **d3.js** Javascript library, which is the main library of the live visualization plugin.

The client presentation will use the text box from the original presentation to create a set of HTML buttons in conjunction with the respective question, where one casts a vote for a positive answer and another for a negative response. The communication system between the presentations is the same as the one used by the audience feedback shared plugin functionality.

4. Evaluation

In this chapter, we opted for two methodologies to properly evaluate the application developed, each with a different purpose:

- **Translation Tests** - To ensure the translation to the new dynaslides format (HTML5 with reveal.js) meets the original presentation standards as much as possible, we devised a process to guarantee that most of the features in the Microsoft's Powerpoint presentations are present in the new format. The process consists of Microsoft's Powerpoint presentations, divided by groups of different features (Position, Image and Effects, Shapes, Color, Themes, and Complex Tests) that we translate when new changes to the code or process are executed.

- **Usability Tests** - We asked 24 volunteers to execute several tasks, each one targeting a particular feature in the process of translating a Microsoft's Powerpoint presentation with or without an interactive plugin.

  We started by giving a quick explanation of what the test was about, the purpose of the application developed, and what was expected from the volunteers in the tests.

  The volunteers were then asked to answer the first section of the user questionnaire about themselves.

  We then conceived nine different tasks, each one focused on a particular functionality or path that our application demands. The tests always started with task number 1. This decision guaranteed the basic functionality of translating and opening a dynaslides presentation for the other tasks. The remaining tasks were grouped randomly for each subject so that each task's metrics were independent of the order they were performed.

  Furthermore, in each task, we measured the time it took to finish the task, the number of errors performed during the task, and if they were able to complete the task to understand and analyze the efficiency, effectiveness, and completeness of the set of tasks.

The **results** capture both analyses of the methodologies presented before and explain if the features developed for this project are well implemented and easy to use:

- **Translation** - The process mentioned above regarding the translation tests guarantees that the features present in the predefined presentations are almost identical in the new format.

- **Usability** - Firstly, we wanted to test if the process of translating a Microsoft's Powerpoint presentation into a dynaslides presentation was straightforward, and the results showed exactly that.

  Moreover, we focused on two different sides of the process. Creating a Microsoft's Powerpoint presentation with one of the plugins and using the presentation's plugins while in the browser. Creating a Microsoft's Powerpoint presentation with a plugin appeared to be a difficult process to grasp for some participants since it required the reference guide, and some features explain in it were not coherent.

  The process of using the browser's plugins, corresponding to tasks three, seven, and nine, went well overall, with most participants understanding how each plugin worked. So we consider that they were well implemented and intuitive.

After the tests, we asked the volunteers to answer the System Usability Scale (SUS), which scored 82 out of 100. Although this questionnaire is a subjective view of the overall system's usability, we find it very useful and tell us the application implemented the features as intended.

After that, we did the NASA Task Load Index questionnaire, which scored 13.82 out of 100. This was also an excellent score overall. Therefore, we can say that the tests were very effective in measuring if the application implemented the final features correctly.

5. Conclusions

Today's slideware applications are based on old techniques. They do not take advantage of all the available resources during a presentation. The presenter should interact with the target audience, have access to a dynamic environment while being in the presentation view, and improve each lecture's experience based on feedback as standard available mechanisms.

Therefore, we developed DynaSlides, an application offering a solution that undertakes the aforementioned issues by creating a flexible plugin-based structure, built using a strong foundation and with the help of a transition mechanism from the standard presentation market leader.

Due to the size and complexity of a presentation application such as Microsoft's Powerpoint, our developed presentation environment focused only on the functionality we believe is the core for every presentation technology. This means we only addressed the problems we thought were important to maintain in the new format, and even those are not perfect.

There are many possibilities to pursue in this project. The translation still is far from a complete transformation from Microsoft's Powerpoint. Furthermore, with the current plugin architecture, a new plugin can easily be implemented, and thus adding a new feature becomes much more accessible. The currently implemented plugins also have room for improvement. Since they are proofs of concept, they can still have a lot of different new features implemented.

References

[1] P. Albinson. Enhancing programming lectures using interactive web-based lecture slides. 2016.

[2] R. J. Craig and J. H. Amernic. Powerpoint presentation technology and the dynamics of teaching. *Innovative Higher Education*, 31(3):147–160, 2006.

[3] D. Cyphert. The problem of powerpoint: Visual aid or visual rhetoric? *Business Communication Quarterly*, 67(1):80–84, 2004.

[4] D. Edge, J. Savage, and K. Yatani. Hyperslides: dynamic presentation prototyping. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, pages 671–680. ACM, 2013.

[5] J. Lanir, K. S. Booth, and A. Tang. Multipresenter: a presentation system for (very) large display surfaces. In *Proceedings of the 16th ACM international conference on Multimedia*, pages 519–528. ACM, 2008.

[6] I. Parker. Absolute powerpoint. *The New Yorker*, 28:76–87, 2001.

[7] R. Roels, Y. Baeten, and B. Signer. An interactive data visualisation approach for next generation presentation tools. In *Proceedings of the 8th International Conference on Computer Supported Education*, pages 123–133. SCITEPRESS-Science and Technology Publications, Lda, 2016.

[8] R. Roels, P. Meştereagă, and B. Signer. An interactive source code visualisation plug-in for the mindxpres presentation platform. In *International Conference on Computer Supported Education*, pages 169–188. Springer, 2015.

[9] R. Roels and B. Signer. Mindxpres: An extensible content-driven cross-media presentation platform. In *International Conference on Web Information Systems Engineering*, pages 215–230. Springer, 2014.

[10] G. Shaw, R. Brown, and P. Bromiley. Strategic stories: How 3m is rewriting business planning. *Harvard business review*, 76(3):41–49, 1998.

[11] R. P. Spicer, Y.-R. Lin, and A. Kelliher. Nextslideplease: Agile hyperpresentations. In *Proceedings of the 17th ACM International Conference on Multimedia*, MM '09, pages 1045–1048, New York, NY, USA, 2009. ACM.

[12] M. Thorogood. slidedeck. js: A platform for generating accessible and interactive web-based course content. In *Proceedings of the 21st Western Canadian Conference on Computing Education*, page 13. ACM, 2016.

[13] E. R. Tufte. *Beautiful evidence*, volume 1. Graphics Press Cheshire, CT, 2006.