Vrije Universiteit Brussel

Faculteit Wetenschappen
Departement Informatica
en Toegepaste Informatica

Content Migration and Layout for the MindXpres Presentation Tool

Proefschrift ingediend met het oog op het behalen
van de graad van Master in de Toegepaste Informatica

Joris Vandermeersch

Promotor: Prof. Dr. Beat Signer
Begeleider: Reinout Roels

Mei 2015

Vrije Universiteit Brussel

Faculty of Science
Department of Computer Science
and Applied Computer Science

Content Migration and Layout for the MindXpres Presentation Tool

Graduation thesis submitted in partial fulfillment of the
requirements for the degree of Master in Applied Computer Science

Joris Vandermeersch

Promotor: Prof. Dr. Beat Signer
Advisor: Reinout Roels

May 2015

ABSTRACT

Microsoft PowerPoint continues to be used worldwide in staggering numbers. We try to provide an alternative with MindXpres, facilitating the switch by converting existing Microsoft PowerPoint presentations into MindXpres presentations, and automatically fixing the layout in the process.

ACKNOWLEDGEMENTS

"Simplicity is a great virtue,
but it requires hard work to achieve it and education to appreciate it.
And to make matters worse: complexity sells better."

— *Edsger W. Dijkstra*

CONTENTS

1. INTRODUCTION

For over 25 years, Microsoft PowerPoint has been the market leader in digital prsentations. Admittedly, it was a revolutionary software package when it was first introduced, and its ease-of-use combined with its supreme graphical capabilities — at least compared to other software in the same era – quickly made it one of the most popular software packages in history. 25 years later, Microsoft PowerPoint can claim over 90% market share in presentation software, and on average 30 million Microsoft PowerPoint presentations are created every day.

In this time, Microsoft PowerPoint has gotten many new features, and certainly improved and grew with every new version, but it never really changed its core approach. It started out mimicking the then-popular and widespread use of dia and overhead projection slides, which was at the time a good way to convince people of its purpose, allowing them to feel comfortable with a familiar format instead of alienating potential customers with a new and potentially confusing interface.

However, this interface is quite restricting, and in recent years different approaches have seen the light of day. The zoomable user interface of Prezi is probably the most well-known, but apart from abandoning the traditional slide format it does little to improve or extend the concept of presenting information to an audience.

This is where MindXpres comes in. Its extensible plugin system allows anyone with some knowledge of programming to create new functionality to use in presentations. Examples are interactivity with the audiencer through various means, controlling the presentation from another device — or several! — and (re)modelling data while presenting it, based on feedback from the audience.

While this is obviously a big improvement on the traditional presentation model of Microsoft PowerPoint and the likes, it remains hard to convince the general public of its merits. People are generally afraid of change, and it is important to make the transition as smooth as possible. On top of that, people are often worried that the work they did in the past may be lost — or worse, irrelevant — after switching to something new. This alone may be a huge factor in deciding wether or not to start using new software, or to stick

with what they know.

That is where the subject of this thesis comes in. We aim to provide a way for people to convert their existing Microsoft PowerPoint presentations into MindXpres presentations, allowing them to take their previous work with them in their switch to MindXpres. This way, we lower the treshold for them to make the decision to start using MindXpres as their presentation software of choice. Once all their existing Microsoft PowerPoint content is available, usable and editable in MindXpres, it should be obvious to anyone why MindXpres is the better option for their presentations.

Another common problem with Microsoft PowerPoint presentations is the way they look. This is not necessarily the fault of the software; most people just are not trained in graphical design, and as such they know very little about proper layout, color choices, or slide content limits. Everyone has probably encountered slides with full paragraphs of text, too small to read and / or too much to process in the short time the slide is visible — (too) many people have made those slides themselves.

When we say this is not the fault of the software, that is mostly true, as the creators of these slides obviously made a conscious choice to make their content appear like that. It could be said however that Microsoft PowerPoint and other presentation tools are guilty through inaction. We believe it is possible to have software either warn its users against these choices and practices, or — even better — have the software fix these problems automatically.

One of the primary goals of MindXpres is to provide automatic layout, much like LaTeX does, ensuring that the content creator only has to worry about the actual content, while the software takes care of layout. In practice, both LaTeX and MindXpres currently use template-based layouts, where the contents' position is predefined in the template and not related to or based on its size, shape or nature. In the end, everyone who has ever used LaTeX knows that sooner or later you will struggle to get a certain image incorporated in the text correctly, ending up doing the layout yourself anyway, because the predefined template just doesn't work properly for you specific content.

As such, the second part of this thesis focuses on implementing true automatic layout in MindXpres. Again primarily to convince Microsoft PowerPoint users to switch, showing that their presentations actually could look better in MindXpres, while thus also providing new functionality to existing MindXpres users.

2. RELATED WORK

This chapter's content is largely based on "MindXpres: An Extensible Content-driven Cross-Media Presentation Platform" (Roels, 2014).

2.1 Background

The importance of digital presentations in this day and age cannot be understated. Millions of presentations are created every day, supporting the oral transfer of knowledge and playing an important role in educational settings. Their origins as tools for creating physical media such as photographic slides or transparencies for overhead projectors are still reflected in the underlying concepts and principles of slide-based presentation tools. The rectangular boundaries of a slide, and the linear navigation between slides, are still restrictions we face today in digital presentations. Tufte argues that these concepts of slideware have a negative impact on the effectiveness of knowledge transfer (Tufte, 2003). While the presenter is compelled to squeeze complex ideas into a linear sequence of slides, those ideas are rarely sequential by nature, resulting in a loss of relations, overview and details. An initial approach to address these issues might involve creating minimalistic presentations or introducing some structure via a table of contents. Sadly, when complex knowledge or other pieces of rich information need to be presented as is (Farkas, 2006) — as in the domain of learning — this does not work.

One of the main issues with traditional slideware presentations is their monolithic nature, especially when content is spread over many self-contained presentation files. "Reusing" previous work involves either switching between files while giving a presentation or duplicating some slides in the new presentation. It should be noted that this issue is not limited to the reuse of single slides: there is an ever increasing wealth of resources available for reuse, spread over a wide spectrum of distribution channels and formats. The possibility to include content by reference or transclusion (Nelson, 1995) may contribute in crossing the boundaries between different types of media and prove beneficial in the context of modern cross-media presentation tools.

The difference in functionality between the authoring of content and its visualisation is striking as well. The primary editors consist of mostly tool-

bars and buttons used for selecting and specifying the way content should be visualised, while support for authoring the content itself is not quite as extensive. Modern slideware has grown to include basic multimedia types such as videos, but most content is still rather static. It is, for example, not possible during a presentation to easily switch from a bar chart to a pie chart data visualisation, or to dynamically change some values in the represented data and immediately see the effect in the graph, which could be beneficial for knowledge transfer (Holzinger et al., 2008). The audience could also be more actively involved in the presentation, through audience response and classroom connectivity systems providing multi-device interfaces allowing to share knowledge and results during as well as after a presentation. The evolution of presentations is reminiscent of the Web2.0 movements where users have switched roles from purely consuming content to contributing as well, content has become more dynamic and interactive, and service-oriented architectures ("The Cloud") have ensured decentralisation of content.

In order to move a step towards the next generation of cross-media presentation tools, it is essential to allow the rapid prototyping and evaluation of new concepts for the representation, visualisation and interaction with content.

TODO After introducing existing slideware solutions, we discuss the requirements for next generation presentation tools. This is followed by a description of the extensible MindXpres architecture and its plug-in mechanism. The web technology-based implementation of MindXpres is validated based on a number of use cases and MindXpres plug-ins and followed by a discussion of future work. TODO

2.2 Existing solutions

The impact, benefits and issues of slideware have been studied ever since digital slideware has been introduced. While some studies acknowledge slideware as a teaching aid (Holzinger et al., 2008), Tufte (Tufte, 2003) heavily criticises slideware for sticking to outdated concepts. He addresses the many consequences of spatial limitations or linear navigation and relates them to the human mind which works differently. One of Tufte's conclusions, which is also confirmed by Adams (Adams, 2006), is that slide-based presentations are not suitable for all kinds of knowledge transfer and in particular not in scientific settings. Recent work shows that it is important for the learning process that content is well integrated in the greater whole, both structurally and visually (Gross and Harmon, 2009), which is influenced by the navigation and visualisation. There have been a number of different approaches to offer

non-linear navigation. Zoomable User Interfaces (ZUIs) as used by CounterPoint (Good and Bederson, 2002), Fly (Lichtschlag et al., 2009) or Prezi, offer virtually unlimited space. Also Microsoft has experimented with zoomable interfaces in pptPlex. While ZUIs are one way to escape the boundaries of the slide, we have seen other approaches such as MultiPresenter (Lanir et al., 2008) or tiling slideshows (Chen et al., 2006). PaperPoint (Signer and Norrie, 2007a) and Palette (Nelson et al., 1999) further enable the non-linear navigation of digital presentations based on a slide selection via augmented paper-based interfaces. Finally, there is a category of authoring tools that use hypermedia to enable different paths through a set of slides. NextSlidePlease (Spicer et al., 2012) allows users to create a weighted graph of slides and may suggest navigational paths based on the link weights and the remaining presentation time. Microsoft follows this trend with their HyperSlides (Edge et al., 2013) project. The potential of Microsoft PowerPoint as an authoring tool for hypermedia-based presentations has further been investigated by Garcia (Garcia, 2004).

Existing presentation tools require content to be duplicated for reuse, resulting in multiple redundant copies that need to be kept up to date. Even though some attempts have been made to address this issue, there is room for improvement. When it comes to document formats for more general educational purposes, there are formats such as the Learning Material Markup Language (LMML) (Sü and Freitag, 2002), the Connexions Markup Language (CNXML) and the eLesson Markup Language (eLML) (Fisler and Bleisch, 2006). The common factor of these formats is their focus on the reuse of content, but always at a relatively high granularity level. Content is organised in lessons or modules and users are encouraged to use these, as a whole, in their teaching. When we examined the formats in more detail, we noticed that they support outgoing links to external content but not the inclusion of content via references (transclusion). In the context of presentations, Microsoft's Slide Libraries are central repositories for the storage of slides in order to facilitate slide sharing and reuse within a company. However, one needs to set up a SharePoint server which might represent a hurdle for some users. Slides still need to be searched and manually copied into presentations. Furthermore, users are responsible to push back updates to the repository or update slides when they have been modified on the server side. Other commercial tools with similar intentions and functionality for content reuse are SlideRocket or SlideShare. The SliDL (Can os et al., 2010) research framework provides a service-oriented architecture for storing and tagging slides in a database for reuse. However, it shares some of the shortcomings of Microsoft's Slide Libraries. The ALOCOM (Verbert et al., 2008) framework for flexible content reuse consists of a content ontology and a

(de)composition framework for legacy documents including Microsoft PowerPoint documents, Wikipedia pages and SCORM content packages. While ALOCOM succeeds in the decomposition of legacy documents, it might be too rigid for evolving presentation formats and the tool is furthermore only supporting the authoring phase.

There is not only a similarity in the evolution of the Web and presentation environments, but a number of the issues presented in this section have solutions in the setting of the Web. It is therefore not surprising that more recently we see the use of web technologies for realising presentation solutions. The Simple Standards-based Slide Show System (S5)[1] is an XHTML-based file format for slideshows which enforces the classical slideware model. The Slidy (Raggett, 2006) initiative by the W3C introduces another presentation format which is based on the standard slideware model. While these two formats are too limited for our needs, they have some interesting properties. Both formats show a clean separation of content and presentation via CSS themes. The visualisation is resolution independent and the layout and font size are adapted to the available screen real estate. Finally, we would like to mention recent HTML5-based presentation solutions, including projects such as impress.js, deck.js, Shower or reveal.js. A major benefit of applying a widely used open standard such as HTML is the cross-device support. Nevertheless, also these solutions show some restrictions in terms of visualisation, navigation and cross-media support.

Most of the tools and projects presented in this section focus on specific novel ideas for presentations. However, there is no interoperability between the concepts introduced in different tools. While one project might focus on the authoring, another one focusses on novel content types and a third solution introduces radically new navigation mechanisms. Some slideware tools can be extended via third-party plug-ins but the functionality that is exposed to the developers is often limited by the tool's underlying model. For example, Microsoft PowerPoint allows plug-ins to interact with the presentation model, but the model dictates that a presentation must consist of a sequence of slides. This lack of freedom is also a shortcoming of existing web-based presentation formats. We therefore see a need for an open presentation platform such as MindXpres which supports innovation by providing the necessary modularity and interoperability (Bush and Mott, 2009).

[1] http://meyerweb.com/eric/tools/s5/

2.3 Microsoft PowerPoint

Microsoft PowerPoint was officially released in 1990, with Windows 3.0 (Austin, 2009). It had originally been developed as Presenter, but trademark issues caused a name change early on. It was also originally build for the Macintosh, which may seem surprising nowadays but was actually common practice back then since the Macintosh was widely regarded as a better development environment, more mature, more stable and capable of far better performance and visualisations. Some may argue this still rings true today.

Since then, it has grown to be the world's most popular slide show presentation program, alledgedly having been installed on over 1 billion computers worldwide, and being used on average 350 times *per second* (Parks, 2012). In 2012, it had a market share of 95%, leaving the other 5% to be shared by alternatives such as Apple's Keynote, Prezi, SlideRocket and others. While this number is declining, it may not be going as fast as many people think. As most readers of this thesis have heard before, over 30 million Microsoft PowerPoint presentations are created every day, for all kinds of purposes, with good and bad results both presentation-wise and goal-wise.

TODO this probably needs more content.

TODO rework below

2.4 New solutions

We now introduce a number of requirements to support a broad range of presentation styles and visualisations which have been compiled based on a review of the more recent presentation solutions presented in the previous section.

Non-linear Navigation As outlined earlier, the linear traversal of slides is a concept that has been taken over from early photographic slides. Nowadays, users are accustomed to this form of navigation even if it might come with some disadvantages. Any navigation outside of the predefined linear path (e.g. to answer a question from the audience) is rather complicated, since the presenter either needs substantial time to scroll forwards or backwards to the desired slide or has to switch to the slide sorter view. It is further impossible to include a single slide multiple times in the navigational path without any duplication. There are different ways how this lack of flexible navigation can be addressed, including presentation tools that allow the presenter to define non-linear navigation paths (Spicer et al., 2012) (Edge et al., 2013) or zoomable user interfaces (ZUIs) (Good and Bederson, 2002) (Lichtschlag

et al., 2009) (Haller and Abecker, 2010).

Separation of Content and Presentation In order to facilitate experimentation with different visualisations, there should be a clear separation between content and presentation. This allows the authors of a presentation to focus on the content while the visualisation is handled by the presentation tool. Note that this approach is similar to the LaTeX typesetting system where content is written in a standardised structured way and the visualisation is automatically handled by the typesetting system. There is also a LaTeX document class for presentations called Beamer and we were inspired by its structured and content-driven approach. However, the content-related functionality and the visualisation are too limited to be considered as a basis for an extensible presentation tool.

Extensibility In order for a presentation tool to be successful as an experimental platform for new presentation concepts, it should be easy to rapidly prototype new content types and presentation formats as well as innovative navigation and visualisation techniques. It has to be possible to add or replace specific components without requiring changes in the core. In order to be truly extensible, a presentation tool should provide a modular architecture with loosely coupled components. Note that this type of extensibility should not only be offered on the level of content types but also for the visualisation engine or content structures.

Cross-Media Content Reuse In the introduction we briefly mentioned the lack of content reuse in existing presentation tools. There is a wealth of open education material available but it is rather difficult to use this content in presentations. On the other hand, the concept of transclusion works well for digital documents and parts of the Web (e.g. via the HTML img tag). A modern presentation tool should also support the seamless integration of external cross-media content. This includes various mechanisms for including parts of other presentations (e.g. slides), transcluding content from third-party document formats as well as including content from open learning repositories.

Connectivity With the rise of social and mobile technologies, connectivity for multi-device input and output becomes more relevant in the context of presentation tools. Support for multi-directional connectivity is required for a number of reasons. First, it is necessary for the previously mentioned cross-media transclusion from external resources. Second, multi-directional

connectivity forms the backbone for audience feedback via real-time response or voting systems (Dufresne et al., 1996) as well as other forms of multi-device interfaces.

Interactivity We mentioned that content might be more interactive and the extensibility requirement addresses this issue since the targeted architecture should support dynamic or interactive content and visualisations. Nevertheless, the use of mouse and keyboard might not be sufficient for components offering a high level of interaction. Therefore, a presentation tool should enable the integration of other forms of input such a gesture-based interaction based on Microsoft's Kinect controller or digital pen interaction (Signer and Norrie, 2010) as offered by the PaperPoint (Signer and Norrie, 2007a) presentation tool.

Post-Presentation Phase Even if it was never the original goal of slide decks, they often play an important role as study or reference material. While the sharing of traditional slide decks after a presentation is trivial, this changes when the previously mentioned requirements are taken into account. For instance, the nonlinear navigation allows presenters to go through their content in a non-obvious order or input from the audience might drive parts of a presentation. Special attention should therefore be paid to the post-presentation phase. It should not only be easy to play back a presentation with the original navigational path, annotations and audience input, but its content should also be made discoverable and reusable. In accordance with the Web 2.0, we see potential for the social aspect in a post-presentation phase via a content discussion mechanism.

2.5 MindXpres Platform

In this section, we present the general architecture of our MindXpres[2] cross-media presentation platform which is outlined in Figure 2.1 and addresses the requirements presented in the previous section.

2.5.1 Document Format and Authoring Language

Content is stored, structured and referenced in a dedicated MindXpres document format. An individual MindXpres document contains the content itself and may also refer to some external content to be included. A new MindXpres document can be written manually similar to the LaTeX approach introduced

[2] http://mindxpres.com

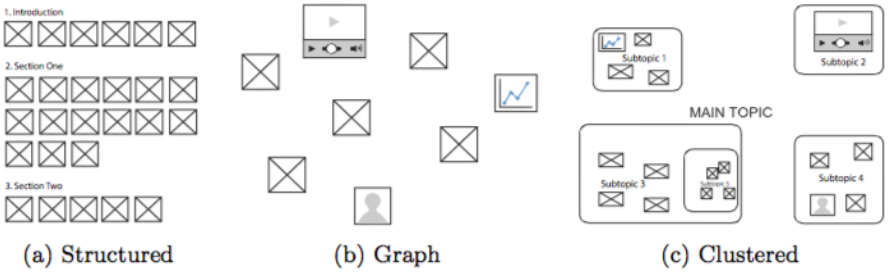


Fig. 2.1: MindXpres architecture

earlier or in the near future it can also be generated via a graphical authoring tool. In contrast to other presentation formats such as Slidy, S5 or OOXML, the authoring language eliminates unnecessary HTML and XML specifics and focusses on a semantically more meaningful vocabulary. The vocabulary of the authoring language is almost completely defined by plug-ins that provide support for various media types and structures. In order to give users some freedom in the way they present their information, the core MindXpres presentation engine only plays a supporting role for plug-ins and lets them define the media types (e.g. video or source code) as well as structures (e.g. slides or graph-based content layouts).

This is also reflected in the document format as each plug-in extends the vocabulary that can be used. Any visual styling including different fonts, colours or backgrounds is achieved by applying specific themes to the underlying content.

2.5.2 Compiler

The compiler transforms a MindXpres document into a self-contained portable MindXpres presentation bundle. While a MindXpres document could be directly interpreted at visualisation time, for a number of reasons we decided to have this intermediary step. First, the compiler allows different types of presentations to be created from the same MindXpres document instance. This means that we can not only create dynamic and interactive presentations but also more static output formats such as PDF documents for printing. Similarly, we cannot always expect that there will be an Internet connection while giving a presentation. For this case, the compiler might create an offline version of a presentation with all necessary content pre-downloaded and included in the MindXpres presentation bundle. Last but not least, the compiler might resolve incompatibility issues by, for instance, converting unsupported video formats.

2.5.3 MindXpres Presentation Bundle

The dynamic MindXpres presentation bundle consists of the compiled content together with a portable cross-platform presentation runtime engine which allows more interactive and networked presentations. Similar to the original document, the compiled presentation content still consists of both, integrated content and references to external resources such as online content that will be retrieved when the presentation is visualised. Note that the content might have been modified by the compiler and, for example, been converted or extracted from other document formats that the runtime engine cannot process. References to external content may have been dereferenced by the compiler for offline viewing.

A presentation bundle's core runtime engine consists of the three modules shown in Figure 2.1. The *content engine* is responsible for processing the content and linking it to the corresponding visualisation plug-ins. The *graphics engine* abstracts all rendering-related functionality. For instance, certain presenters prefer a zoomable user interface in order to provide a better overview of their content (Reuss et al., 2008). This graphical functionality is also exposed to the plug-ins, which can make use of the provided abstractions. The *communication engine* exposes a communication API that can be used by plug-ins. It provides some basic functionality for fetching external content but also offers the possibility to form networks between multiple MindXpres presentation instances as well as to connect to third-party hardware such as digital pens or clicker systems.

In addition to the presentation content and core modules, the presentation bundle contains a set of *themes* and *plug-ins* that are referenced by the content. Themes may contain visual styling on a global as well as on a plug-in level. When the content engine encounters different content types, they are handed over to the specific plug-in which uses the graphics engine to visualise the content.

2.5.4 Plug-in Types

In order to provide the necessary flexibility, all non-core modules are implemented as plug-ins. Even the basic content types such as text, images or bullet lists have been realised via plug-ins and three major categories of plug-ins have to be distinguished:

- *Components* form the basic building blocks of a presentation. They are represented by plug-ins that handle the visualisation for specific content types such as text, images, bullet lists, graphs or videos. The

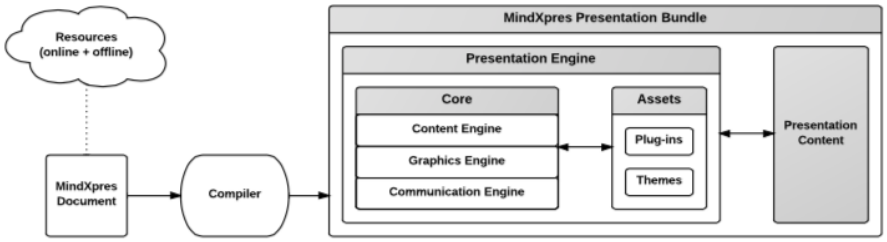


Fig. 2.2: Structure plug-in examples

content engine invokes the corresponding plug-ins in order to visualise the content.

- *Containers* are responsible for grouping and organising components of a specific type. An example of such a container is a slide with each slide containing different content but also some reoccurring elements. Every slide of a presentation may for example contain elements such as a title, a slide number and the author's name, which can be abstracted in a higher level container. Another example is an image container that visualises its content as a horizontally scrollable list of images. Note that we are not restricted to the slide format and content can be laid out in alternative ways.

- *Structures* are high-level structures and layouts for components and containers. For example, content can be scattered in a graph-like structure or it can be clearly grouped in sections like in a book. Both are radically different ways of visualising and navigating content but by abstracting them as plug-ins, the user can easily switch between different presentation styles as the ones shown in Figure 2.2. Structures differ from containers by the fact that they do not impose restrictions on the media types of their child elements and may also influence the default navigational path through the content.

2.5.5 Implementation

HTML5 and its related web technologies were chosen as the backbone for the MindXpres presentation platform. Other options such as JavaFX, Flash or game engines have also been investigated, but HTML5 seemed to be the best choice. The widely accepted HTML5 standard makes MindXpres presentations highly portable and runnable on any device with a recent web browser, including smartphones and tablets. Furthermore, HTML5 provides rich visualisation functionality out of the box and the combination with Cas-

cading Style Sheets (CSS) and third-party JavaScript libraries forms a potent visualisation platform.

Document Format and Authoring Language

The MindXpres document format which allows us to easily express a presentation's content, structure and references is based on the eXtensible Markup Language (XML). A simple example of a presentation defined in our XML-based authoring language is shown in Listing 2.3. The set of valid tags and their structure, except the `presentation` root tag, is defined by the available plug-ins.

```
<presentation>
  <slide title="Vannevar Bush">
    <bulletlist>
      <item>March 11, 1890 - June 28, 1974</item>
      <item>American Engineer, founder of Raytheon</item>
    </bulletlist>
    <image source="bush.jpg"/>
  </slide>
</presentation>
```

Lst. 2.3: Authoring a simple MindXpres presentation

Compiler

The compiler has been realised as a Node.js application. This not only allows the compiler to be used via a web interface or as a web service, but projects such as node-webkit also enable the compiler to run as a local offline desktop application. The choice of using server-side JavaScript was influenced by the fact that Node.js is capable of bridging web and desktop technologies. On the one hand, the framework makes it easy to interact with other web services and to work with HTML, JSON, XML and JavaScript visualisation libraries at compile time. On the other hand, the framework can also perform tasks which are usually not suited for web technologies, including video conversion, legacy document format access, file system access or TCP/IP connectivity.

In order to validate a MindXpres document in the XML format described above, there is an XML Schema which is augmented with additional constraints provided by the plug-ins. After validation, the document is parsed and discovered tags might trigger preprocessor actions by the plug-ins such as the extraction of data from referenced legacy document formats (e.g. Microsoft PowerPoint or Excel) or the conversion of an unsupported video format. The tag is then converted to HTML5 by simply encoding the information in the attributes of a `div` element. The HTML5 standard allows custom

attributes if they start with a `data-` prefix. Listing 2.4 shows parts of the transformed XML document shown in Listing 2.3. Note that the transformation does not include visualisation-specific information but merely results in a valid HTML5 document which is bundled into a self-contained package together with the presentation engine.

```
1 <div data-type="presentation">
2   <div data-type="slide" data-title="Vannevar Bush">
3     <div data-type="bulletlist">
4       ...
```

Lst. 2.4: Transformed HTML5 presentation content

Presentation Engine

The presentation engine's task is to turn the compiled HTML content into a visually appealing and interactive presentation. As highlighted in Figure 2.1, the presentation engine consists of several smaller components which help plug-ins to implement powerful features with minimal effort. The combination of these components enables the rapid prototyping and evaluation of innovative visualisation ideas. A resulting MindXpres presentation combining various structure, container and component plug-ins is shown in Figure 2.5.

Fig. 2.5: A MindXpres presentation

Content Engine When a presentation is loaded, the content engine is the first component that is activated. It processes the content of the HTML presentation by making use of the well-known jQuery JavaScript library.

Whenever a `div` element is discovered, the `data-type` attribute is read and the corresponding plug-ins are notified in order to visualise the content.

Graphics Engine The graphics engine provides support for interesting new visualisation and navigation styles. Next to some basic helper functions, it offers efficient panning, scaling and rotation via CCS3 transformations and supports zoomable user interfaces as well as the more traditional navigation approaches.

Communication Engine The communication engine implements abstractions that allow plug-ins to retrieve external content at run time. It further provides the architectural foundation to form networks between different MindXpres instances or to integrate third-party hardware (Roels and Signer, 2014). For our MindXpres prototype, we used a small Intel Next Unit of Computing Kit (NUC) with high-end WiFi and Bluetooth modules to act as a central access point and provide the underlying network support. MindXpres instances use WebSockets to communicate with other MindXpres instances via the access point. The access point further acts as a container for data adapters which translate input from third-party input and output devices into a generic representation that can be used by the MindXpres instances in the network. In order to go beyond simple broadcast-based communication, we have implemented a routing mechanism based on the publish-subscribe pattern where plug-ins can subscribe to specific events or publish information. The communication engine provides the basis for audience response systems (Roels and Signer, 2014) or even full classroom communication systems where functionality is only limited by the creativity of plug-in developers.

Plug-ins Plug-ins are implemented as JavaScript bundles which consist of a folder containing JavaScript files and other resources such as CSS files, images or other JavaScript libraries. As a first convention, a plug-in should provide a manifest file with a predefined name. The manifest provides metadata such as the plug-in name and version but also a list of tags to be used in a presentation. The plug-in claims unique ownership for these tags and is in charge for their visualisation if they are encountered by the content engine. As a second convention, a plug-in must provide at least one JavaScript file implementing certain methods, one of them being the `init()` method which is called when the plug-in is loaded by the presentation engine. It is up to the plug-in to load additional JavaScript or CSS via the provided dependency injection functionality. A second method to be implemented is the `visualise()` method which is invoked with a pointer to the corresponding

DOM node as a parameter when the content engine encounters a tag to be visualised. A plug-in is free to modify the DOM tree and may also register callbacks to handle future interaction with the content.

Themes We currently use CSS to provide a basic templating system. These themes offer styling either on a global or on a plug-in level. However, we see this as a temporary solution as it is not well-suited for alternative compiler outputs (e.g. PDF) and a more generic templating scheme is planned for the future.

2.5.6 Use Cases

In order to validate the architectural and technological choices, we demonstrate the extensibility and feasibility of MindXpres as a rapid prototyping platform by presenting a number of content- and navigation-specific plug-ins that have been developed so far. Additional plug-ins for audience-driven functionality such as real-time polls, screen mirroring and navigational takeover can be found in (Roels and Signer, 2014).

Structured Overview Plug-in

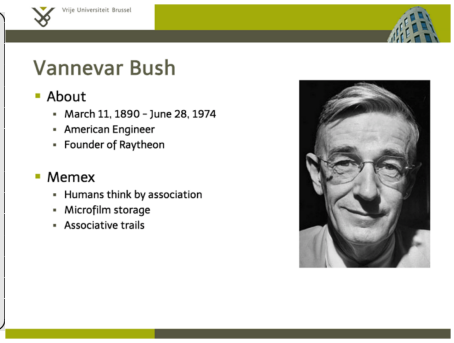
In Section 2.5 we have explained how structure plug-ins may change the way presentations are visualised and navigated. In order to illustrate this, we have implemented a structure plug-in called *structured layout* which combines a zoomable user interface with the ability to group content into sections. The resulting visualisation of the *structured layout* plug-in is shown in Figure 2.5. Whenever a new section is reached, the view is zoomed out to provide an overview of the content within the section and communicate a sense of progress.

Slide Plug-in

In order to also support the traditional slide concept, we created a slide-like container plug-in. While the benefits and issues of using slides with a fixed size are debatable, we implemented this plug-in as a proof of the framework's versatility. The main function of the slide plug-in is to provide a rectangular styleable component container with a title and some other information. Containers may also offer functionality to layout their content. In this case, the slide plug-in offers a quick and easy layout mechanism which allows the presenter to partition the slide into rows and columns. Content is then assigned to these slots in the order that it is discovered. The use of

the slide plug-in together with the resulting visualisation is exemplified in Listing 2.6. It also shows the use of the image plug-in (a component plug-in) which enables a simple form of cross-media transclusion. A visualised external image can be cropped and filters (e.g. colour correction) may be applied without duplicating or modifying the original source.

```
<slide layout="\60\40" title="Vannevar Bush">
 <bulletlist>
  <item>About
   <item>March 11, 1890 - June 28, 1974</item>
   <item>American Engineer</item>
   <item>Founder of Raytheon</item>
  </item>
  <item>...
  </item>
 </bulletlist>
 <image source="http://example.com/bush.jpg">
  <crop bounds="10%, 5%, -10%, -20%" />
 </image>
</slide>
```



Lst. 2.6: Slide plug-in

Enhanced Video Plug-in

When videos are used in educational settings, we often need more functionality than what is offered by the average video player (Reuss et al., 2008). MindXpres provides the enhanced video plug-in shown in Listing 2.7 with the possibility to overlay a video with text or arbitrary shapes. This overlay functionality can be used as a basic captioning system as well as to highlight items of interest during playback.

Furthermore, we added the option to trigger certain events at specified times. One can define that a video should automatically pause at a certain point, highlight an object and continue playing after a specified amount of time. Additional features include the bookmarking of certain positions in a video for direct access or the possibility to display multiple videos in a synchronised manner. Our enhanced video plug-in injects the default HTML5 video player and overlays it with a transparent `div` element for augmentation. Currently we make use of the HTML5 video API to synchronise the creation and removal of overlays but a SMIL-based implementation might be used in the future.

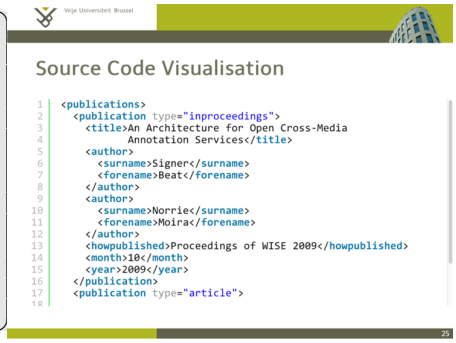
```
<video source="vid.mp4">
 <caption start="0:00" duration="1500ms">
  Lecture 3 - Butterfly Species
 </caption>
 <pause start="0:43" duration="5s">
  <caption>
   The peacock butterfly (aglais io) ...
  </caption>
  <highlight x="30%" y="9%"
             width="35%" height="40%" />
 </pause>
</video>
```

Lst. 2.7: Enhanced video plug-in

Source Code Visualisation Plug-in

Earlier, we mentioned the difficulty of visualising complex resources such as source code. Our MindXpres source code plug-in exports a `code` tag which allows the presenter to paste their code into a presentation and have MindXpres visualise it nicely by making use of syntax highlighting via the SyntaxHighlighter[3] JavaScript library. Whenever the content engine encounters a `code` tag, it invokes the code plug-in to beautify the code and automatically adds vertical scrollbars for larger pieces of source code as shown in Listing 2.8.

```
<code>
 <publications>
  <publication type="inproceedings">
   <title>An Architecture for Open Cross-Media
          Annotation Services</title>
   <author>
     <surname>Signer</surname>
     <forename>Beat</forename>
   </author>
   <author>
     <surname>Norrie</surname>
     <forename>Moira</forename>
   ...
</code>
```



Lst. 2.8: Source code visualisation

2.5.7 *Discussion and Future Work*

MindXpres currently supports transclusion and cross-media content reuse on the plug-in level. For instance, the image or video plug-in can visualise (and enhance) external resources, a dictionary plug-in might retrieve definitions

[3] http://alexgorbatchev.com/SyntaxHighlighter/

on demand via a web service or we might create a plug-in that allows us to import content (e.g. Microsoft PowerPoint slides) from legacy documents at compile time. Nevertheless, we are currently investigating the introduction of generic reuse tags in our document format which would allow the presenter to transclude arbitrary parts of other MindXpres presentations. While our focus has been on the cross-media aspect of resources that can be used in a presentation, we might also investigate the cross-media publishing aspect via alternative compiler output formats.

We are aware that the current authoring of MindXpres presentations has some usability issues. The average presenter cannot be expected to construct an XML document or any CSS themes. In order to tackle this issue and further evaluate MindXpres in real-life settings, we are currently developing a graphical MindXpres authoring tool. We further intend to provide a central plug-in repository which would make it easy for novice users to find, install and use new plug-ins via the authoring tool. In the long run, we intend to revise the use of monolithic documents and move towards repositories of semantically linked information based on the RSL hypermedia metamodel (Signer and Norrie, 2007b). This would not only promote content reuse and sharing, but also create opportunities for context-aware as well as semi-automatic presentation authoring where relevant content is recommended by the authoring tool.

2.5.8 Conclusion

TODO do we need this?

We have presented MindXpres, an HTML5 and web technology-based presentation platform addressing the lacking extensibility of existing slideware solutions. The extensibility of the MindXpres platform heavily relies on a plug-in mechanism and we have outlined how the tool can be extended on the content, visualisation as well as on the interaction level. By providing different forms of cross-media content transclusion, our solution further avoids the redundant storage and replication of slides. The presented MindXpres document format in combination with a compiler and the corresponding plug-ins offers the opportunity to have compile- or run-time cross-media content transclusion from third-party resources. At the same time, the flexible and extensible document model in combination with a zoomable user interface allows us to escape the boundaries of traditional slide formats. The presented MindXpres solution represents a promising platform for the unification of existing presentation concepts as well as for the rapid prototyping and investigation of new ideas for next generation cross-media presentation solutions.

TODO ... or this?

MindXpres tries to be an alternative unlike other alternatives, stepping away from the classic slide format and introducing a plugin system to allow literally anything you can think of. The obvious choices are visualisations, animations, layout and the embedding of various media such as video, but it can also provide interaction with the audience, allow different presenters to take control from their own device, arbitrarily and ad-hoc change the way data is presented allowing to play into the audience's reactions unhindered by design decisions made beforehand, and so many other novel ways of presenting that captivate the audience instead of lulling them to sleep.

MindXpres was first created by my advisor, Reinout Roels, in 2012 as part of his Master thesis, and has since been built upon by himself and other students.

3. PROBLEM STATEMENT

3.1 Terminology

3.2 Stuff we want to solve

TODO change this section's title

4. APPROACH

4.1 Compilation process

The first part of the approach is fairly straightforward in its basic explanation: we had to convert Microsoft PowerPoint presentations into MindXpres presentations. This involves finding out how Microsoft PowerPoint presentations are structured, getting the parts wee need out of that structure, and then putting those parts together in de MindXpres structure. Since the author of this thesis has a small background in compilers (Vandermeersch and Timbermont, 2009) it did not take long to see the resemblance of this process to that of a compiler.

A compiler takes source code and transforms it into a working program with the semantics described by that source code. The compilation process consists of several steps. First the source code is tokenized, which means the symbols in the code are identified one by one and classified in certain categories.

Then the tokens are processed by a parser into an intermediary form called a parse tree. A parser looks for certain predefined patterns in the source code. These patterns are part of the source code's language syntax. As such, these two steps analyse and validate the source code's syntax. If part of the code does not match any pattern, the parser and the compilation process stop and the user gets a message saying the code's syntax is invalid.

When a parse tree is constructed, the compilation process can alter it, to improve it. Certain patterns in the parse tree may be replaceable by different patterns with the same outcome, but with more optimal execution. This part of the compilation process is optional, and is called compiler optimization. Optimizations can consist of many things, depending on the language. For example, some languages guarantee tail call optimization, where infinite loops can be constructed by letting a function call itself as its last statement without causing a stack overflow. This is something the compiler (or interpreter) can optimize during this part of the compilation process.

After this, the parse tree can be written out to produce the desired output. Every node in the tree has a well-defined equivalent in the target language's syntax. The target language can be Assembly, which consists of the

exact instructions a CPU needs to carry out a program, or it can be another programming language. Many compilers of higher-level languages translate their language into C, for several reasons: the C compilers that translate C into Assembly have been optimized so much that it is easier to rely on them than to put an enormous amount of effort into optimizing another language; C compilers exist for most — if not all — CPU architectures, which means translating a language into C makes it compatible with all those architectures, while it would cost a lot more effort to write different compilers for every architecture you would want to make your language available on.

The conversion tool that is the purpose of this thesis, can be described in a similar succession of steps. First, we take a Microsoft PowerPoint presentation and tokenize and parse it into an intermediary structure that allows us to perform other operations, or 'optimizations', on it. The intermediary form consists of a 'parse tree' containing the components of the original presentation — a component tree, if you will.

With this structure, we can construct a MindXpres presentation containing the same components in the same place, essentially creating a 'program' with the same semantic meaning as the original 'source'.

4.2 Compiler optimizations

Since the conversion process resembles that of a compiler, it seemed logical at first to make automatic layout a part of that process, as some kind of 'compiler optimization'.

At first, we tried to traverse the component tree, giving its objects new coordinates and sizes so that they would fit together on every slide as well as possible. This seemed an easy solution, but the results were sub-optimal. On top of that, we soon realised that we were in essence creating another template out of which a presentation would be made, which was exactly the opposite of what we were trying to do. As such, we abandoned this approach.

We then switched to a different method: defining constraints for every component, in the form of margins, maximum sizes and other limits, and then calculating a way to satisfy all constraints while fitting content together on each slide. While this is clearly a better method, it turned out the compiler optimization phase was not the best place in the process to take care of this.

In the end, we decided to take a different approach, relying on the layout engine of MindXpres itself and enhancing that engine to create the automatic layout we wanteD.

4.3 Using MindXpres

MindXpres already takes care of layout for you, since that is one of its primary goals. The way it does this currently is however heavily based on templates, while we wanted a layout engine that could take any content and put in in an appropriate layout without any directions from the user. As such, we had to enhance MindXpres's layout engine to use constraints, based on the size of the content, and try to find an optimal position for every component it is given.

5. IMPLEMENTATION

To implement the *ppt2mxp* conversion tool that is the subject of this thesis, we chose the Java programming language (Gosling and McGilton, 1995), version 8. Although the author has significant experience with lots of other, more interesting, more compelling, more fun languages, several reasons pushed us towards Java, the least of them being its ease of use. Of course, Java *is* easy to use — it would not have become as popular as it is nowadays if it wasn't. It has a fairly clear and logical syntax, a consistent structure, and an extensive standard library. At conception in 1995, its performance was abysmal, but through the years it has steadily improved and somewhere between Java 5 and 6 it became an industry standard.

Quite a number of IDEs have been created to further improve developers' experience working with Java. Netbeans, Eclipse and IntelliJ come to mine, although there are many others, and of course you can still write Java using a standard (or advanced) text editor such as Notepad or VIM. While the author usually prefers the latter for any kind of text editing — this very document was written entirely using VIM — the weapon of choice when it comes to Java is currently IntelliJ. The way IntelliJ practically writes more than half of the code automatically for you is something no other IDE has been able to match. Naturally, this is the author's personal opinion and should not be seen as fact, but if you're looking for a new Java IDE, it's definitely worth checking out. The prospect of using IntelliJ for this thesis has definitely contributed to the decision of using Java. It should be noted that, had another Java IDE been required, this thesis might never have seen the light of day.

The vast and extensive amount of libraries available for Java was obviously one of the more important reasons to make this choice. The existence of the Apache POI library (see section 5.1) was a huge help in reaching our goal; without it, we would have had to figure out the very obfuscated .ppt file format structure, which undoubtedly would have taken up more time than was available to us. Other libraries like Spring, which allows the programmer to use and reuse components without writing complex systems to instantiate them, further increased our resolve to make Java our primary technology choice.

However, Java is not the only technology used here. MindXpres is written entirely in HTML5, so any tool that somehow relates to MindXpres sooner or later needs to use HTML5 as well. The widely accepted HTML5 standard makes MindXpres presentations highly portable and runnable on any device with a recent web browser, including smartphones and tablets (Roels, 2014).

In the following sections we discuss how the various technologies were used to create the *ppt2mxp* tool.

5.1 Taking Microsoft PowerPoint apart

We found Apache POI library very helpful in this part of the implementation. The POI Library — formerly "Poor Obfuscation Implementation" (Sundaram, 2004) — is a Java library that provides an API to access Microsoft document formats. The most mature (and most popular) part of it is HSSF, which stands for Horrible SpreadSheet Format, and which is used by Java developers worldwide to access Microsoft Excel spreadsheet data.

For our purposes, we relied on HSLF ("Horrible SLideshow Format"), which gave us access to a Microsoft PowerPoint presentation's contents in many ways. We could access all images at once, or every bit of text from the whole presentation, but the most interesting to us was the ability to access contents on a per-slide basis.

This allowed us to loop over the presentation's slides, converting them one by one, by placing the contents of each slide in a MindXpres slide equivalent.

5.2 Generating MindXpres

5.2.1 Plain HTML5

Since the MindXpres compiler was not functional during most of this thesis' implementation, we decided to generate an html file much like the MindXpres compiler would, including the MindXpres JavaScript library and plugins. This required us to first find out how MindXpres works on the inside, which proved to be a steep learning curve but gave us more insight into the software than we would've gotten if we only had to generate MindXpres XML and leave the rest to the compiler.

5.2.2 MindXpres XML

5.3 Creating layouts

5.3.1 Using constraints

5.3.2 Other ways

6. CONCLUSIONS AND FUTURE WORK

6.1 Contribution

6.2 Future Work

BIBLIOGRAPHY

R. Roels, *MindXpres: An Extensible Content-driven Cross-Media Presentation Platform*, 2014.

E. R. Tufte, *The Cognitive Style of PowerPoint: Pitching Out Corrupts Within*, 2003, graphics Press.

D. K. Farkas, *Toward a Better Understanding of PowerPoint Deck Design*, 2006, information Design Journal + Document Design 14(2).

T. H. Nelson, *The Heart of Connection: Hypermedia Unified by Transclusion*, 1995, communications of the ACM 38(8).

A. Holzinger, M. D. Kickmeier-Rust, and D. Albert, *Dynamic Media in Computer Science Education; Content Complexity and Learning Performance: Is Less More?*, 2008, educational Technology & Society 11(1).

C. Adams, *PowerPoint, Habits of Mind, and Classroom Culture*, 2006, journal of Curriculum Studies 38(4).

A. Gross and J. Harmon, *The Structure of PowerPoint Presentations: The Art of Grasping Things Whole*, 2009, iEEE Transactions on Professional Communication 52(2).

L. Good and B. Bederson, *Zoomable User Interfaces as a Medium for Slide Show Presentations*, 2002, information Visualization 1(1).

L. Lichtschlag, T. Karrer, and J. Borchers, *Fly: A Tool to Author Planar Presentations*, 2009, in: Proc. of CHI 2009, Boston, USA.

J. Lanir, K. Booth, and A. Tang, *MultiPresenter: A Presentation System for (Very) Large Display Surfaces*, 2008, in: Proc. of MM 2008, Vancouver, Canada.

J. Chen, W. Chu, J. Kuo, C. Weng, and J. Wu, *Tiling Slideshow*, 2006, in: Proc. of Multimedia 2006, Santa Barbara, USA.

B. Signer and M. Norrie, *PaperPoint: A Paper-based Presentation and Interactive Paper Prototyping Tool*, 2007, in: Proc. of TEI 2007, Baton Rouge, USA.

L. Nelson, S. Ichimura, E. Pedersen, and L. Adams, *Palette: A Paper Interface for Giving Presentations*, 1999, in: Proc. of CHI 1999, Pittsburgh, USA.

R. Spicer, Y. Lin, A. Kelliher, and H. Sundaram, *NextSlidePlease: Authoring and Delivering Agile Multimedia Presentations*, 2012, aCM TOMCCAP 8(4).

D. Edge, J. Savage, and K. Yatani, *HyperSlides: Dynamic Presentation Prototyping*, 2013, in: Proc. of the CHI 2013, Paris, France.

P. Garcia, *Retooling PowerPoint for Hypermedia Authoring*, 2004, in: Proc. of SITE 2004, Atlanta, USA.

C. Sü and B. Freitag, *MML-TheLearningMaterialMarkupLanguageFramework*, 2002, in: Proc. of ICL 2002, Villach, Austria.

J. Fisler and S. Bleisch, *eLML, the eLesson Markup Language: Developing sustainable e-Learning Content Using an Open Source XML Framework*, 2006, in: Proc. of WEBIST 2006, Setubal, Portugal.

J. Can os, M. Marante, and M. Llavador, *SliDL: A Slide Digital Library Supporting Content Reuse in Presentations*, 2010, in: Proc. of ECDL 2010, Glasgow, UK.

K. Verbert, X. Ochoa, and E. Duval, *The ALOCOM Framework: Towards Scalable Content Reuse*, 2008, journal of Digital Information 9(1).

D. Raggett, *Slidy - A Web Based Alternative to Microsoft PowerPoint*, 2006, in: Proc. of XTech 2006, Amsterdam, The Netherlands.

M. Bush and J. Mott, *The Transformation of Learning with Technology: Learner-Centricity, Content and Tool Malleability, and Network Effects*, 2009, educational Technology Magazine 49(2).

D. Austin, *Beginnings of PowerPoint: A Personal Technical Story*, 2009.

B. Parks, *Death to PowerPoint*, 2012.

H. Haller and A. Abecker, *iMapping: A Zooming User Interface Approach for Personal and Semantic Knowledge Management*, 2010, in: Proc. of Hypertext 2010, Toronto, Canada.

R. Dufresne, W. Gerace, W. Leonard, J. Mestre, and L. Wenk, *Classtalk: A Classroom Communication System for Active Learning*, 1996, journal of Computing in Higher Education 7(2).

B. Signer and M. Norrie, *Interactive Paper: Past, Present and Future*, 2010, in: Proc. of PaperComp 2010, Copenhagen, Denmark.

E. Reuss, B. Signer, and M. Norrie, *PowerPoint Multimedia Presentations in Computer Science Education: What do Users Need? In: Proc*, 2008, of USAB 2008, Graz, Austria.

R. Roels and B. Signer, *A Unified Communication Platform for Enriching and Enhancing Presentations with Active Learning Components*, 2014, in: Proc. of ICALT 2014, Athens, Greece.

B. Signer and M. Norrie, *As We May Link: A General Metamodel for Hypermedia Systems*, 2007, in: Proc. of ER 2007, Auckland, New Zealand.

J. Vandermeersch and S. Timbermont, *Compiling Mutually Tail-Recursive Functions Into Iterative Loops*, 2009.

J. Gosling and H. McGilton, *The Java Language Environment*, 1995.

E. Sundaram, *Excelling in Excel with Java*, 2004.