The original purpose of this project was to address an issue called -

**PDF3: Ensuring the correct tab and reading order in PDF documents for Web Content Accessibility Guidelines (WCAG).**

For details please look into articles below:

[https://www.w3.org/TR/WCAG20-TECHS/pdf#PDF3](_blank)

1.3.2 **Meaningful Sequence**: When the sequence in which content is presented affects its meaning, a correct reading sequence can be programmatically determined. (Level A)

<https://www.w3.org/TR/UNDERSTANDING-WCAG20/content-structure-separation-sequence.html>

Some internet articles use an “informal name” called “**Logical Reading Order**” (**LRO**) instead of the official name.

**Why the project is called hierarchicaldatatransformations?**

From an implementation standpoint, this project includes source code files for very complicated custom sorting of XML files.

Once I started PDF accessibility compliance project I’ve realized that this project could be extended:

1. To include other hierarchical data format transformations, such as JSON, HTML, or COBOL data.

2. It applies to many other tasks and not only to accessibility.

3. Other developers are encouraged to contribute to the project. This way we will have a common reusable code in one place. And you can quickly prototype it in other situations.

**Background details of the “Reading Order in PDF documents” issue.**

Some organizations only concern with the HTML portion of AODA compliance, but it's not enough.

**People, who can't read online reports, need to convey a meaning of PDF online report by hearing it’s content instead.**

So **for them it's critically important that the information from PDF file must be in Custom Logical Reading Order.**

PDF file could be a bank statement, or a bill from an online retailer order or any government-issued document. And what if a self-isolated person with poor vision gets this document? He can’t read it visually. A speech reader program on his/her computer could read PDF file out loud. **Really it’s a part of essential software services.**

Let’s look realistically into the near future. Coronavirus will not disappear soon.

So to prevent the virus spread, many people will rely more and more on online government, banking, retail, insurance, and financial sector activities.

Plus the ability to hear the content of a file, while doing something else, is a very convenient time saver for everyone.

**So the solution proposed in this project will have huge business value for all of us.**

**What reading order is? (VERY IMPORTANT)**

Humans read simple documents in z-order (from left to right, top to bottom fashion).

But for real-life existing online dynamically generated PDF files structure is much more complicated. Humans interpret some text as a label and some as value, when they see the report with their eyes.

Challenges:

1. There are hundreds of those label value pairs in the report.

2. PDF file itself does not have a programmatic way to specify matching label value pairs.

3. Once PDF file was generated, all of those pdf labels and values elements were inserted in a random order. Typical scenario sequence is label77, value15, many other elements, value77, ..., label 4. Speech reader will read PDF files in that random order. So it will be impossible to understand the content of the file by hearing it.

5. Logical Reading Order of specific report is not in z-order.

6. Label could be allocated on the left or the top side of the “matching” value (somewhere around).

7. Two different labels could have different spacing to their matching value. And you can’t change the report layout.

8. Label could be a few pixels to the right of the value.

**Here is how I’ve solved this problem for a real client.**

My online PDF files were generated by the J2EE application using Jasper Reports.

Initially I had 28 very large XML files with random order of XML elements inside. Some files contained 7000 rows and a few hundred of XML elements. Those template files defined the order in which pdf elements would be inserted into the generated PDF file.

**The layout of my Customized Logical Reading Order** was extremely complicated by itself.

PDF report has 8 sections in the middle and 2 sections on the left and right sides. Within each section we had z-order reading with challenges 6-8. Schematically it looked this way.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Left side section | Section 2 started here | | Section 1 | Right Side section |
| Section 2 continued | | |
| Section 3 | Section 4 | |
| Section 5 | | Section 6 |
| Section 7 | | Section 8 |

It was practically impossible to reallocate hundreds of XML elements manually.

So to accommodate my LRO I have to develop the JasperReportsTransformator program, which will generate “identical” XML files. In other words XML elements have to be reallocated within a new XML file, that now they follow LRO.

JasperReportsTransformator class does complex XML transformations of specific jasper jrxml files. It generates a new jrxml file with my custom LRO.

It also does complex validations, to make sure that the content of the new and original version of the jrxml file is the same. There are more details, ideas, and techniques documented in the source code files.

Working for this project gave me a great exposure to pdf files structure, which could apply to tasks beyond PDF accessibility.

And experience to address other PDF accessibility issues besides Logical Reading Order.

The project uses Java 8. It’s a standalone application.

**Dependencies**: **Java JDK 8+, git** must-have. **Eclipse** is recommended.

Project URL:

<https://github.com/IgorArtSoft/hierarchicaldatatransformations>

Repository owner and author of this idea: admin, Igor Artimenko.