LibreOffice
The Document Foundation

News from the ODF Toolkit

FOSDEM 2022

Svante Schubert

Hi I am Svante Schubert
I am co-maintainer of the ODF Toolkit (with Michael Stahl).
Michael and I are also both editors in the OASIS ODF technical committee (TC) and I am also co-chair of the ODF TC.

ODF Toolkit

Highlights

- Two Releases
  - 0.9.0 - JDK 8 - Simple API (last)
  - 0.10.0 - JDK 11 - ODF Change API
- Finalising **ODF Toolkit 1.0.0**
- Upcoming:

    In-Document Search API
    by NGI Zero (NLnet) fund

Highlights:

Two releases in November last year...

Soon 1.0.0:
- ODF 1.3 support is missing..
- Some Open question on „Java Module“

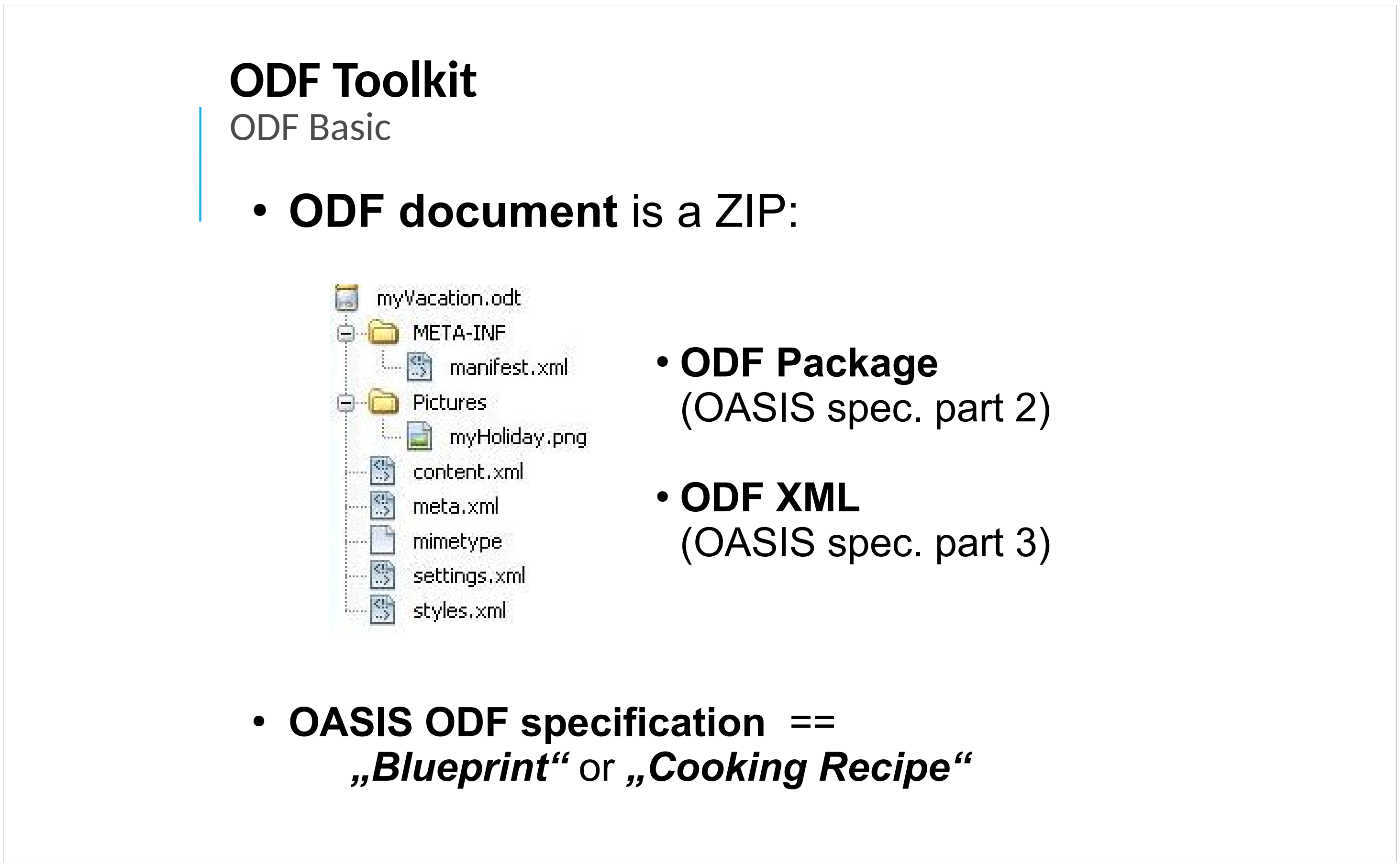
I got funding to work full time on Toolkit!

ODF Toolkit? What for?

800 pounds sculpture (was stolen) from Healdsburg, Calif.

Not a **silver bullet**, just a **BIG hammer** :-)
It’s a toolkit, therefore a **set of tools**…

Obviously this tools meant for **ODF,**
I might add developers as it is a Software Library mostly written in Java



ODF Toolkit

ODF Basic

- **ODF document** is a ZIP:

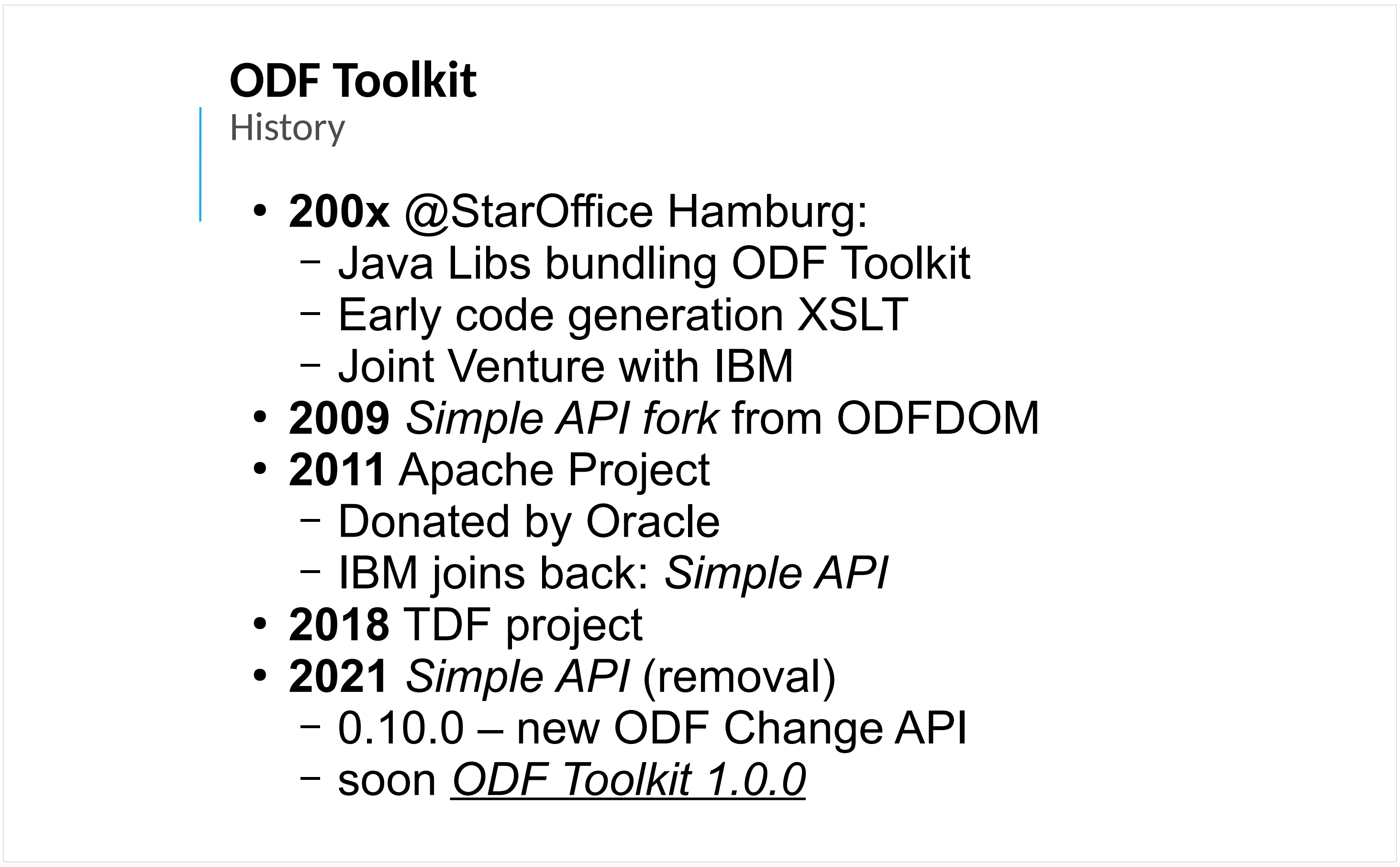
```
myVacation.odt
├─ META-INF
│   └─ manifest.xml
├─ Pictures
│   └─ myHoliday.png
├─ content.xml
├─ meta.xml
├─ mimetype
├─ settings.xml
└─ styles.xml
```

- **ODF Package**
  (OASIS spec. part 2)
- **ODF XML**
  (OASIS spec. part 3)

- **OASIS ODF specification** ==
  „Blueprint" or ***„Cooking Recipe"***

- ODF Package is defined by:
  OASIS specification part 3 (ODF 1.2) – Part 2/1.3
  XML Manifest (content table),
  encryption,
  signature
- EPUB Format used ODF 1.1 unfortunately became
  incompatible to ODF 1.2 (miscommunication?)

- ODF XML is defined by:
  OASIS specification part 1 (ODF 1.2) – Part 3/1.3
  XML (content,styles,meta) files



ODF Toolkit

History

- **200x** @StarOffice Hamburg:
  - Java Libs bundling ODF Toolkit
  - Early code generation XSLT
  - Joint Venture with IBM
- **2009** *Simple API fork* from ODFDOM
- **2011** Apache Project
  - Donated by Oracle
  - IBM joins back: *Simple API*
- **2018** TDF project
- **2021** *Simple API* (removal)
  - 0.10.0 – new ODF Change API
  - soon *ODF Toolkit 1.0.0*

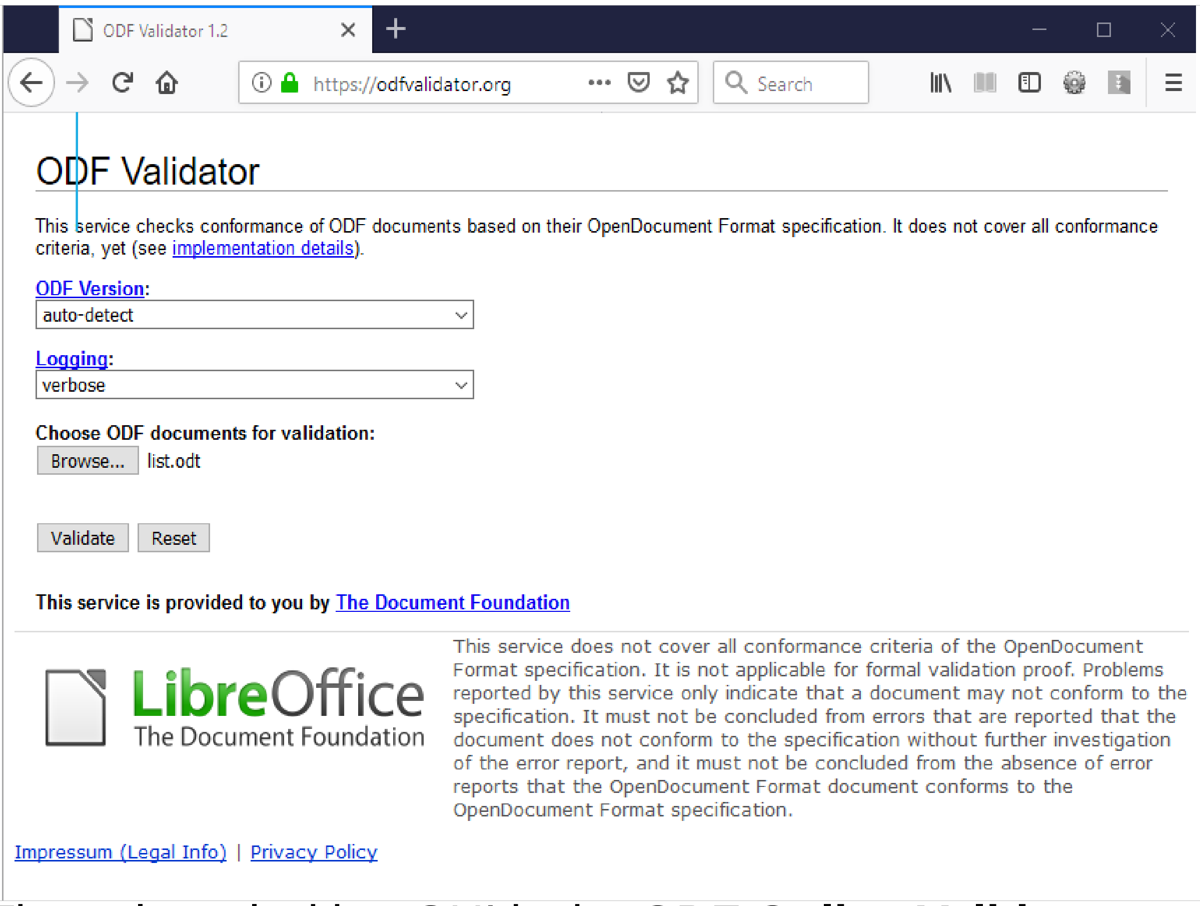
- Michael and I were both till 2011 developers in Hamburg

While he was working on the Writer, I was focused on a Java backend of web office (this Java Backend)

2012 ODFOM was forked for an open-source WebOffice from Open-Xchanged adding the ODF Change API

2017 I got a PrototypeFund:

- as part of it: merge back the OX fork



ODF Validator

This service checks conformance of ODF documents based on their OpenDocument Format specification. It does not cover all conformance criteria, yet (see implementation details).

ODF Version:
auto-detect

Logging:
verbose

Choose ODF documents for validation:
Browse... list.odt

Validate Reset

This service is provided to you by The Document Foundation

LibreOffice The Document Foundation

This service does not cover all conformance criteria of the OpenDocument Format specification. It is not applicable for formal validation proof. Problems reported by this service only indicate that a document may not conform to the specification. It must not be concluded from errors that are reported that the document does not conform to the specification without further investigation of the error report, and it must not be concluded from the absence of error reports that the OpenDocument Format document conforms to the OpenDocument Format specification.

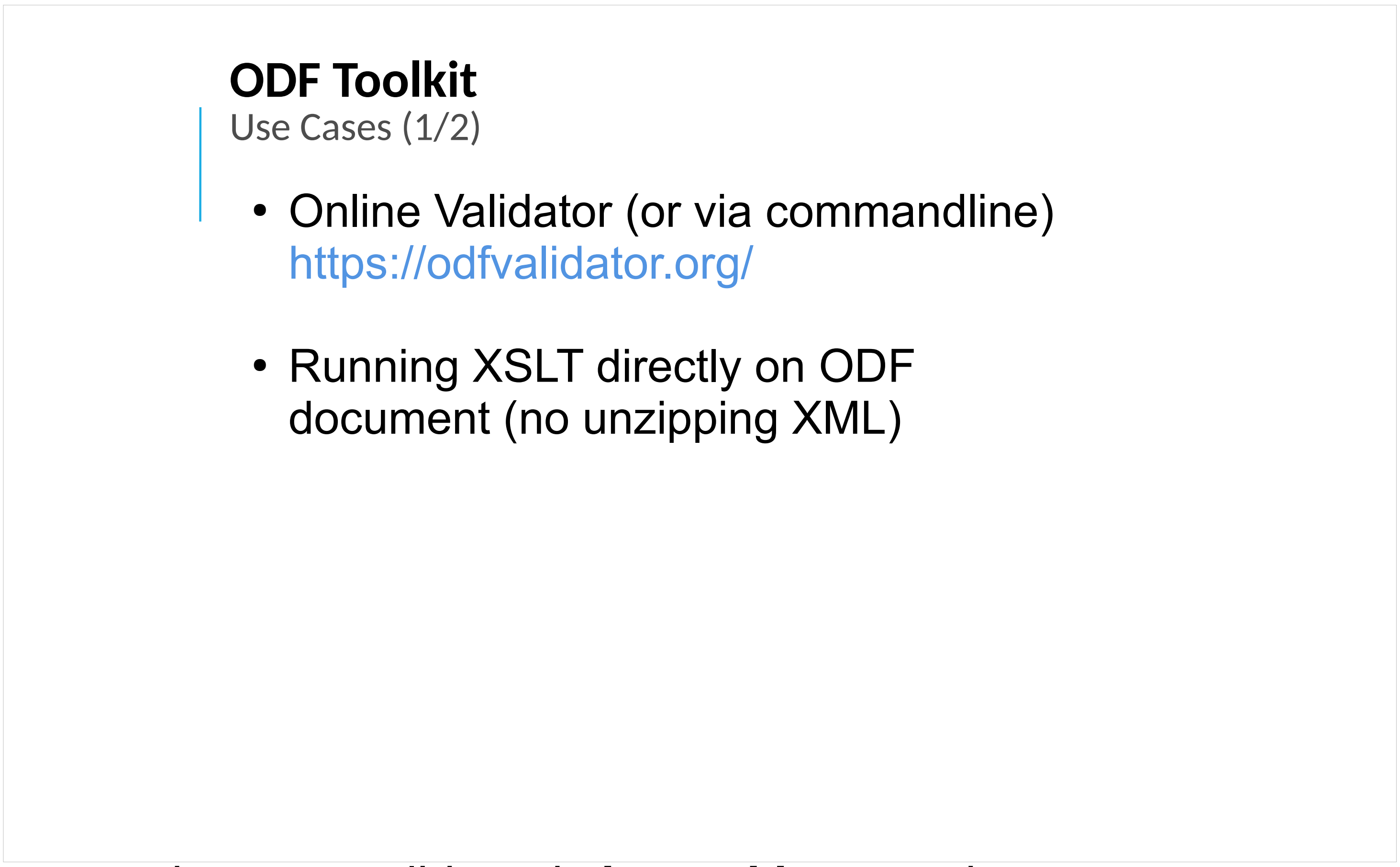
Impressum (Legal Info) | Privacy Policy

The only tool with a GUI is the **ODF Online Validator**..

Hosted by at **odfvalidator.org**

Coming with GUI and validator bundled as a **JEE WAR** (Web Archive) easily to be added to a a Web Application Server, such as Apache Tomcat

2011 small funding from Nlnet: provide GUI & WAR



ODF Toolkit

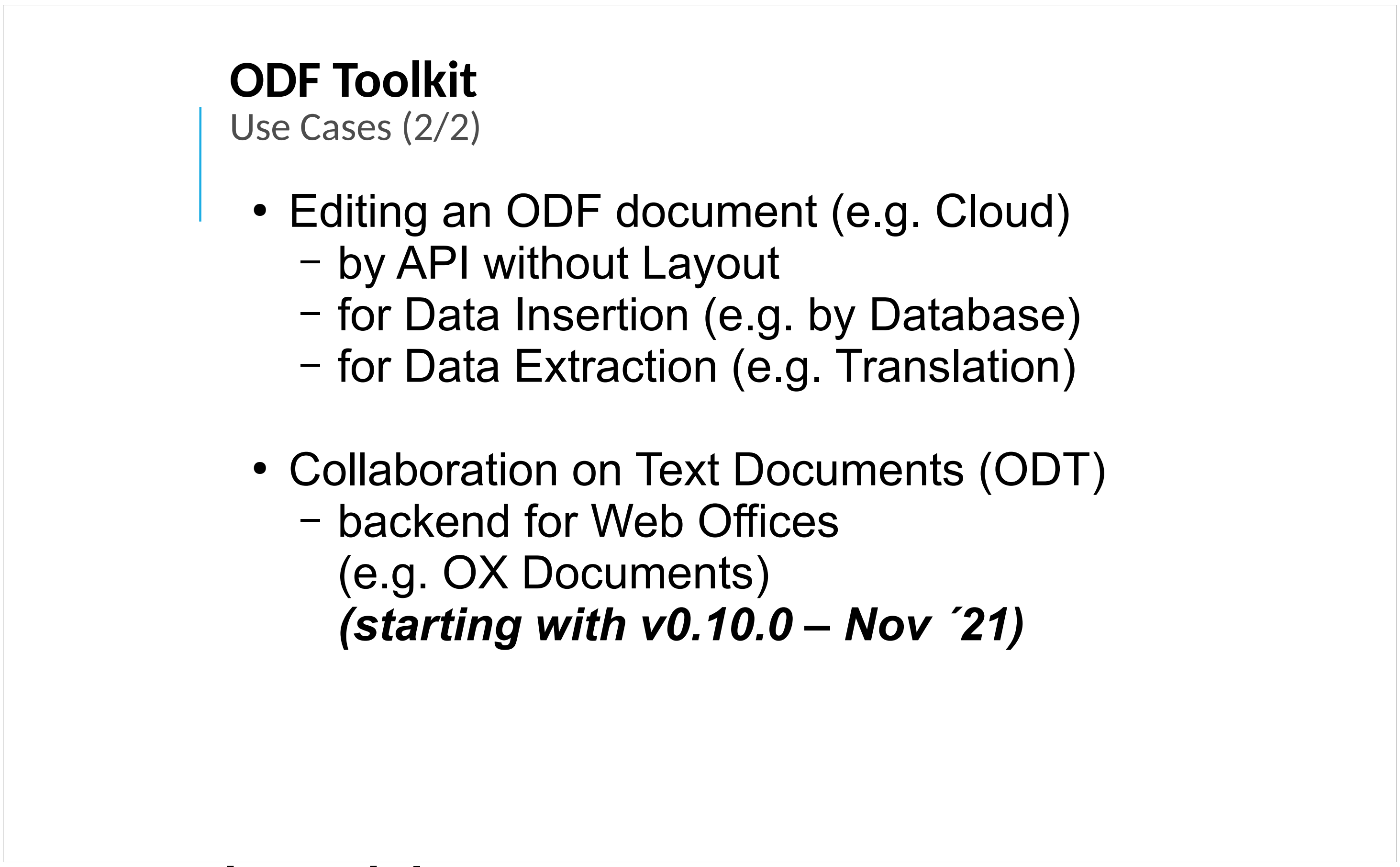
Use Cases (1/2)

- Online Validator (or via commandline)
  https://odfvalidator.org/

- Running XSLT directly on ODF document (no unzipping XML)

- The ODF validator is **hosted by TDF** the Document Foundation!

- Who knows XSLT? Running on the **zipped XML within the ODF** without the need to unzip the files.



ODF Toolkit

Use Cases (2/2)

- Editing an ODF document (e.g. Cloud)
  - by API without Layout
  - for Data Insertion (e.g. by Database)
  - for Data Extraction (e.g. Translation)

- Collaboration on Text Documents (ODT)
  - backend for Web Offices
    (e.g. OX Documents)
    (starting with v0.10.0 – Nov ´21)

- **Main Module ODFDOM:**

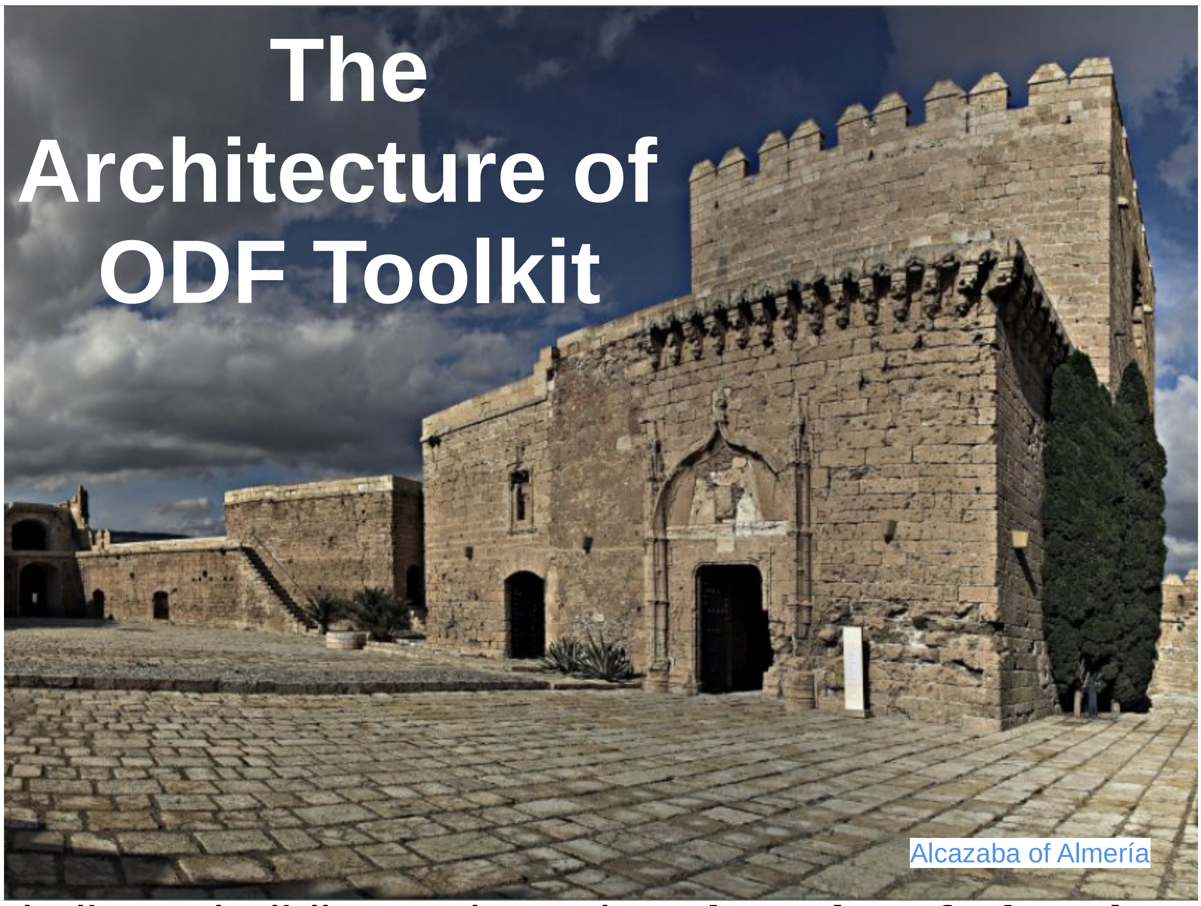
ODF manipulation by API on server

A) **New Data** in templates

Create ODF from Database:

1. Get visible text – 2. translate – 3. exchange existing text!

- COLLABORATION with ODFDOM (soon with 1.0.0)



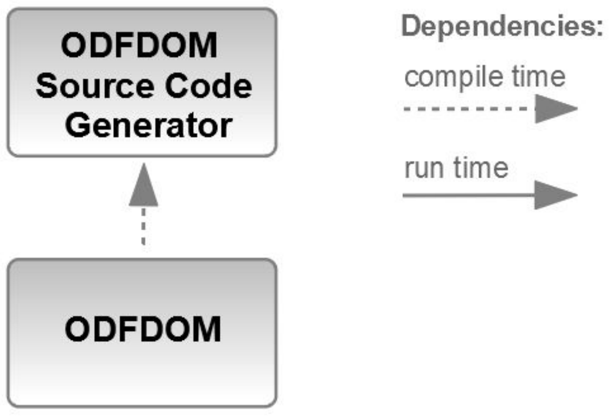
The Architecture of ODF Toolkit

Similar to buildings – here the **Alcazaba of Almeria – architecture matter** for software.

Starting with the **modules** of the **ODF Toolkit**

ODF Toolkit

Architecture

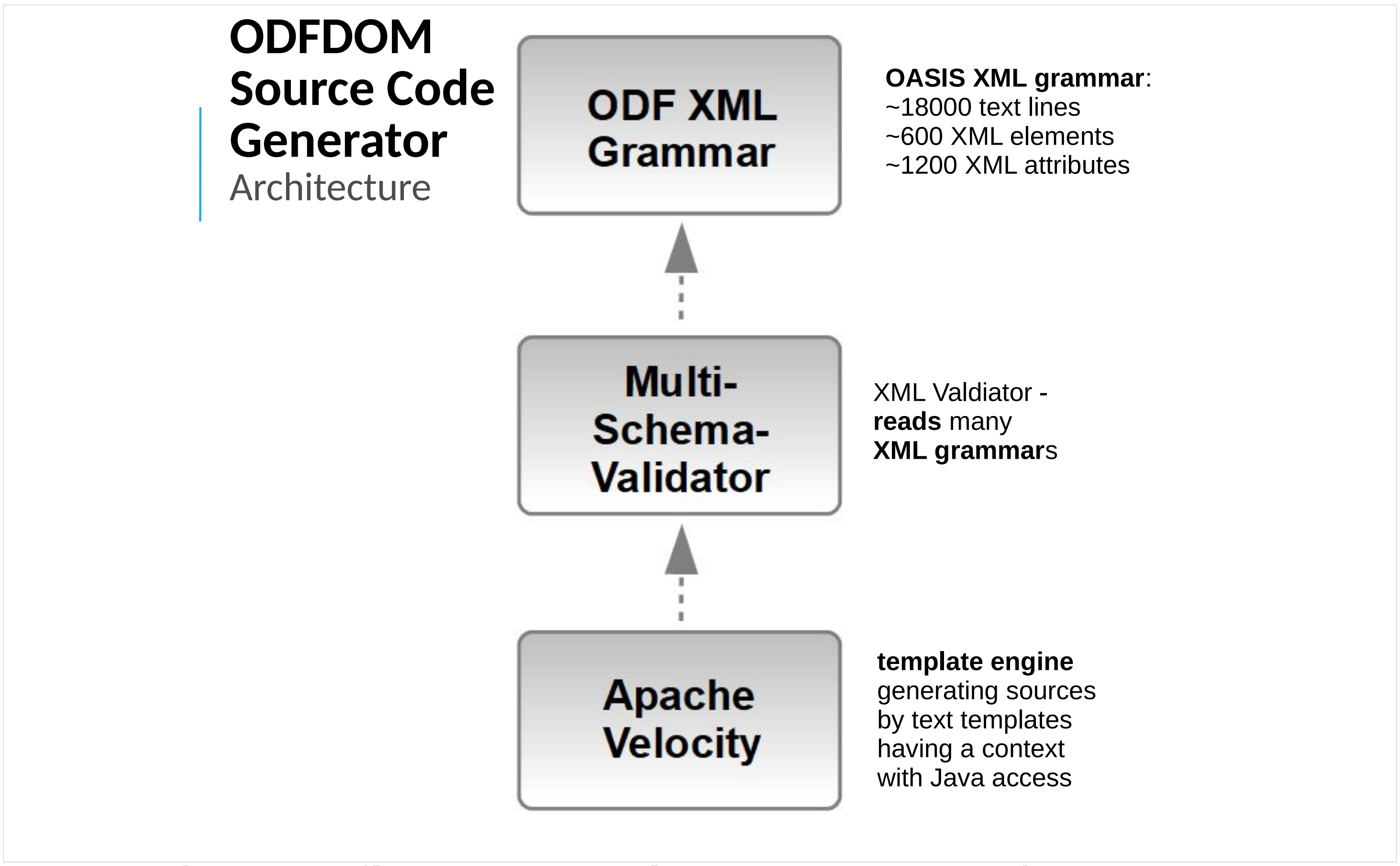


Similar to **JAXB for W3C schema** (grammar).
(Java XML Binding of JEE, works only for W3C schema)

The **ODF grammar/schema** is used to **generate Java sources**.

Every ODF element and attribute is created as a **typed Java DOM class** in ODFDOM to ease developer to create valid ODF.

In the future, likely a second approach in **RUST**?
Allowing multi-threading and better memory handling.



ODFDOM Source Code Generator
Architecture

ODF XML Grammar

OASIS XML grammar:
~18000 text lines
~600 XML elements
~1200 XML attributes

Multi-Schema-Validator

XML Valdiator -
reads many
XML grammars

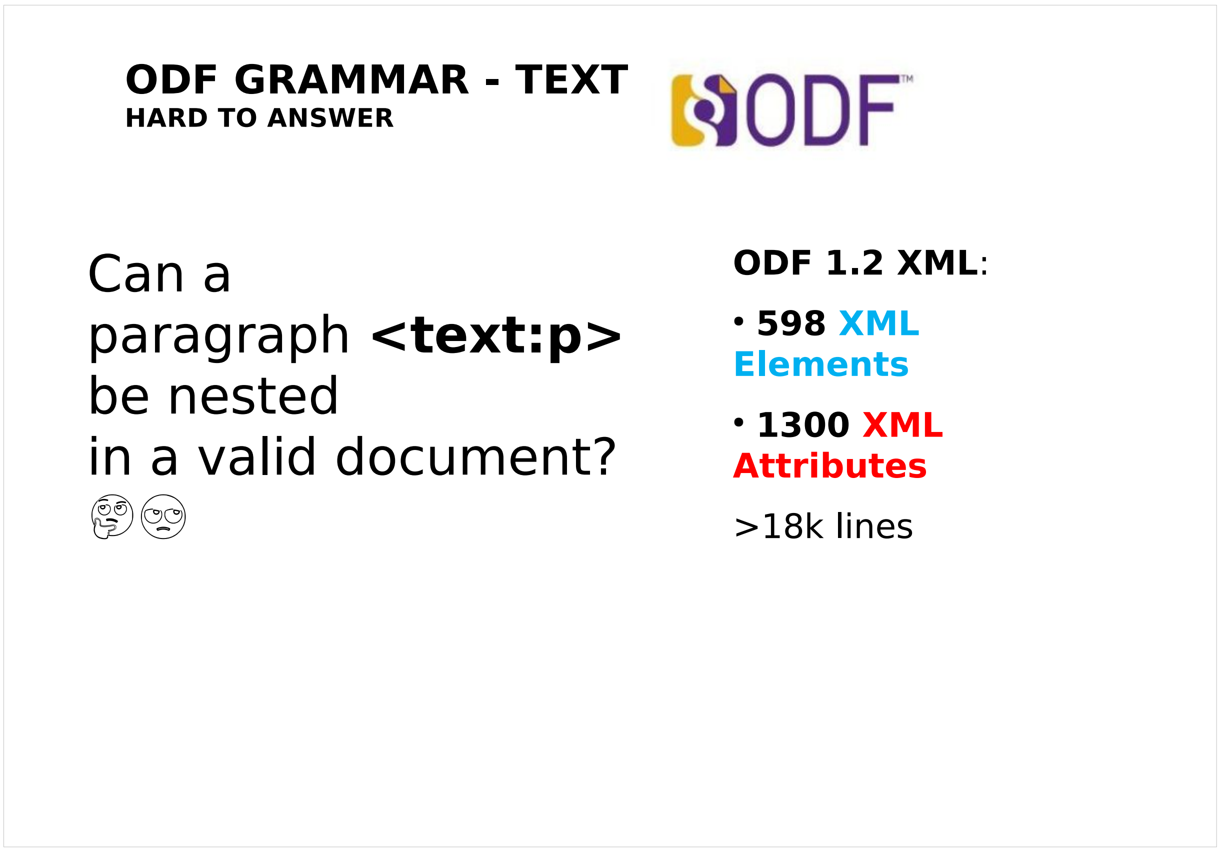
Apache Velocity

template engine
generating sources
by text templates
having a context
with Java access

In the very **first approach XSLT** was used to create from the ODF XML grammar the Java sources.

We **split the complexity** and
reused existing opensource software:
a) Multi-Schema-Validator (MVS)
to read the XML grammar
b) Apache Velocity Engine as Template engine

With ODF Toolkit 0.10.0 introduced common tree data structure to allow other tooling on the XML grammar



ODF GRAMMAR - TEXT

HARD TO ANSWER

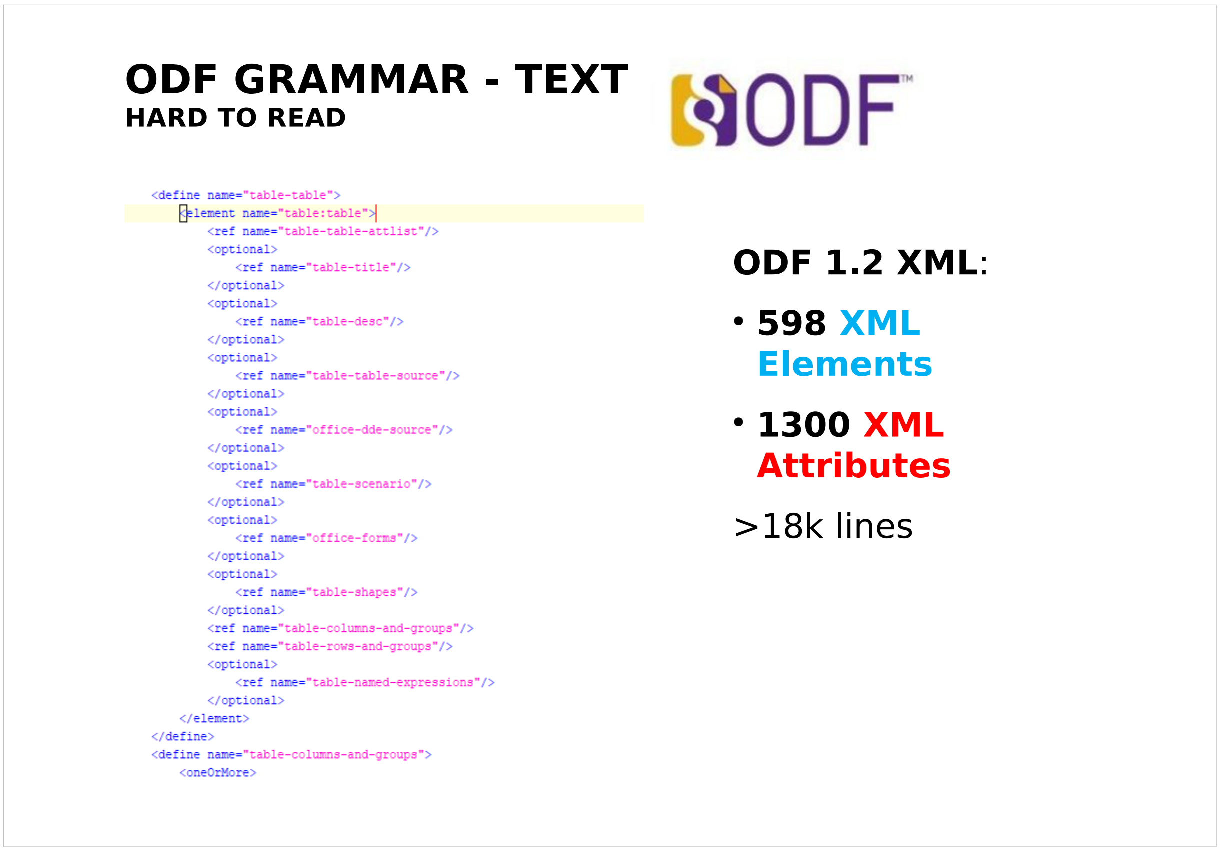
Can a paragraph **<text:p>** be nested in a valid document? 🤔🙄

ODF 1.2 XML:

- **598 XML Elements**
- **1300 XML Attributes**

>18k lines

- Grammar hard to understand
- Only basic set of information to define changes
- hard to standarizse changes
- Analysis difficult..



ODF GRAMMAR - TEXT

HARD TO READ

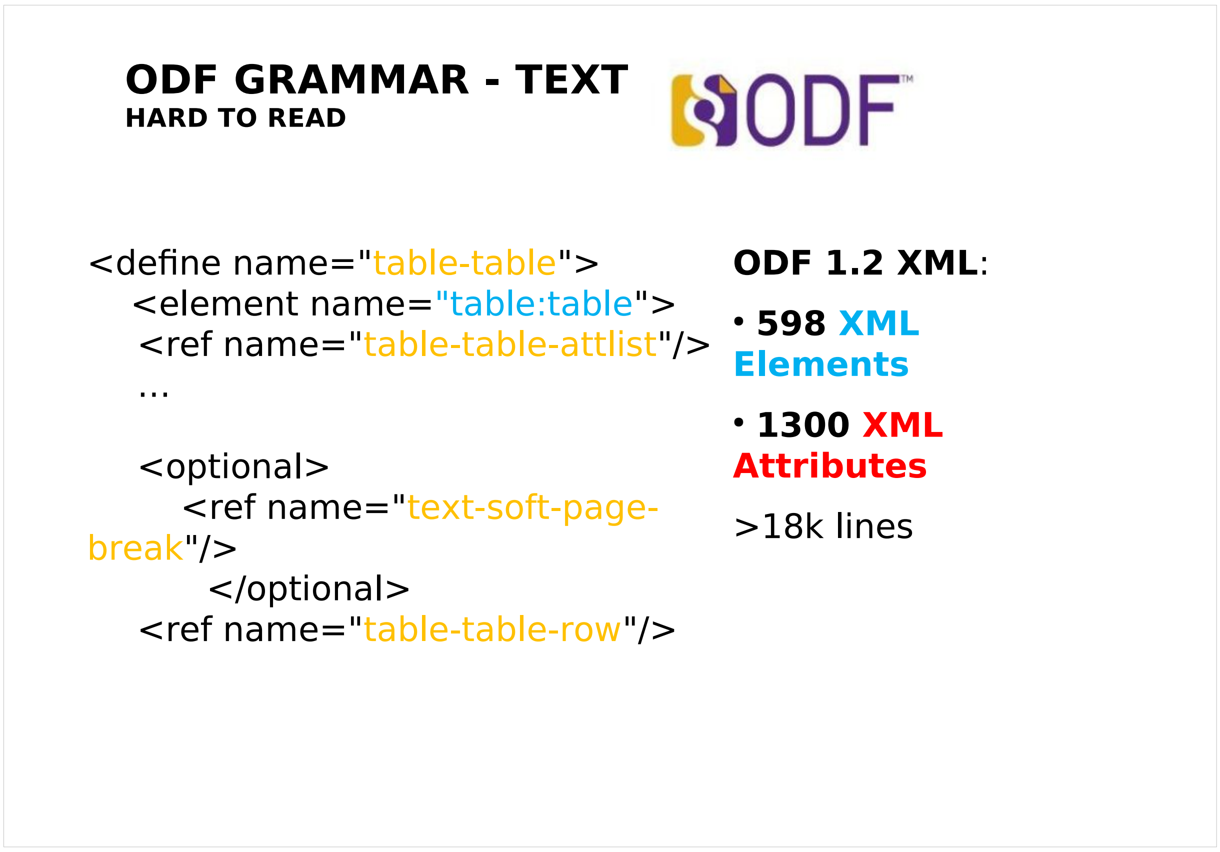
```
<define name="table-table">
    <element name="table:table">
        <ref name="table-table-attlist"/>
        <optional>
            <ref name="table-title"/>
        </optional>
        <optional>
            <ref name="table-desc"/>
        </optional>
        <optional>
            <ref name="table-table-source"/>
        </optional>
        <optional>
            <ref name="office-dde-source"/>
        </optional>
        <optional>
            <ref name="table-scenario"/>
        </optional>
        <optional>
            <ref name="office-forms"/>
        </optional>
        <optional>
            <ref name="table-shapes"/>
        </optional>
        <ref name="table-columns-and-groups"/>
        <ref name="table-rows-and-groups"/>
        <optional>
            <ref name="table-named-expressions"/>
        </optional>
    </element>
</define>
<define name="table-columns-and-groups">
    <oneOrMore>
```

ODF 1.2 XML:

- **598 XML Elements**
- **1300 XML Attributes**

>18k lines

- Grammar hard to understand
- Only basic set of information to define changes
- hard to standarizse changes
- Analysis difficult..



ODF GRAMMAR - TEXT

HARD TO READ

```
<define name="table-table">
    <element name="table:table">
     <ref name="table-table-attlist"/>
     ...

     <optional>
         <ref name="text-soft-page-
break"/>
           </optional>
     <ref name="table-table-row"/>
```

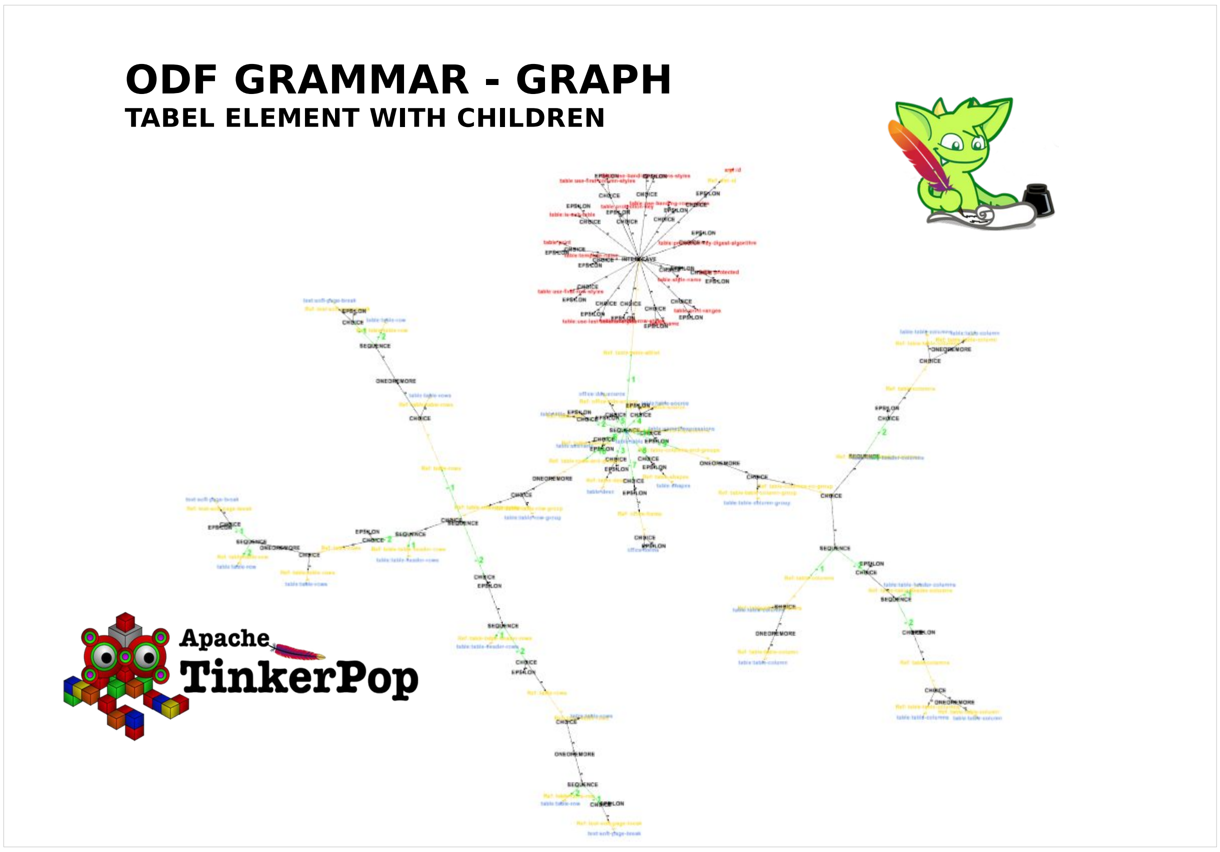
ODF 1.2 XML:

- **598 XML Elements**
- **1300 XML Attributes**

>18k lines

- Grammar hard to understand
- Only basic set of information to define changes
- hard to standarizse changes

- Analysis difficult..



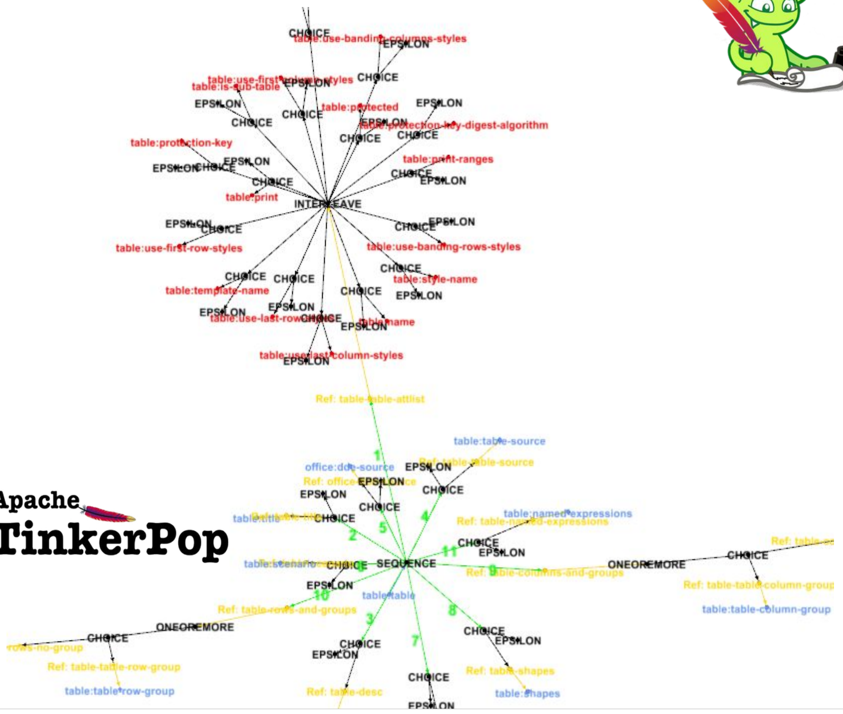
ODF GRAMMAR - GRAPH

TABEL ELEMENT WITH CHILDREN

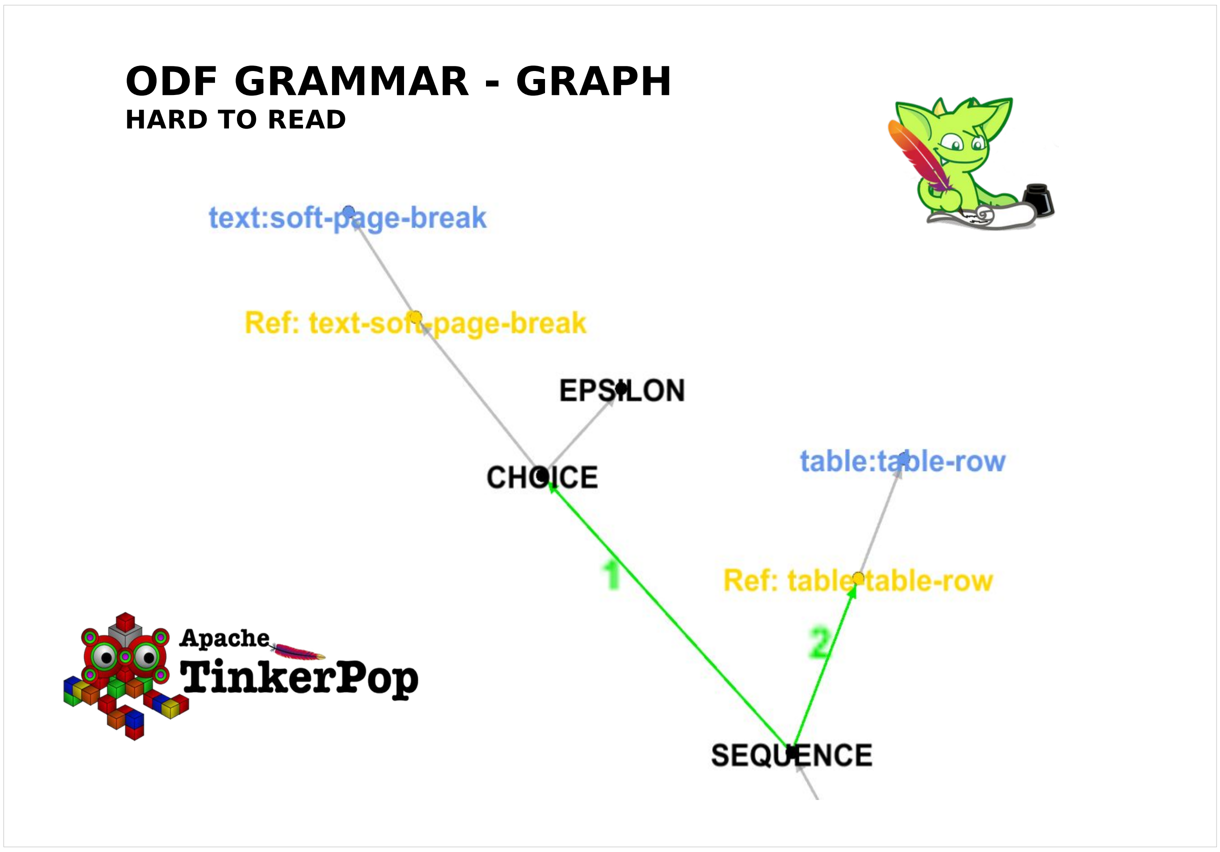
- https://groups.google.com/forum/#!searchin/gremlin-users/svante%7Csort:date/gremlin-users/P8MdzzlFtng/vYqYlukJAgAJ

ODF GRAMMAR - GRAPH

TABEL ELEMENT WITH CHILDREN



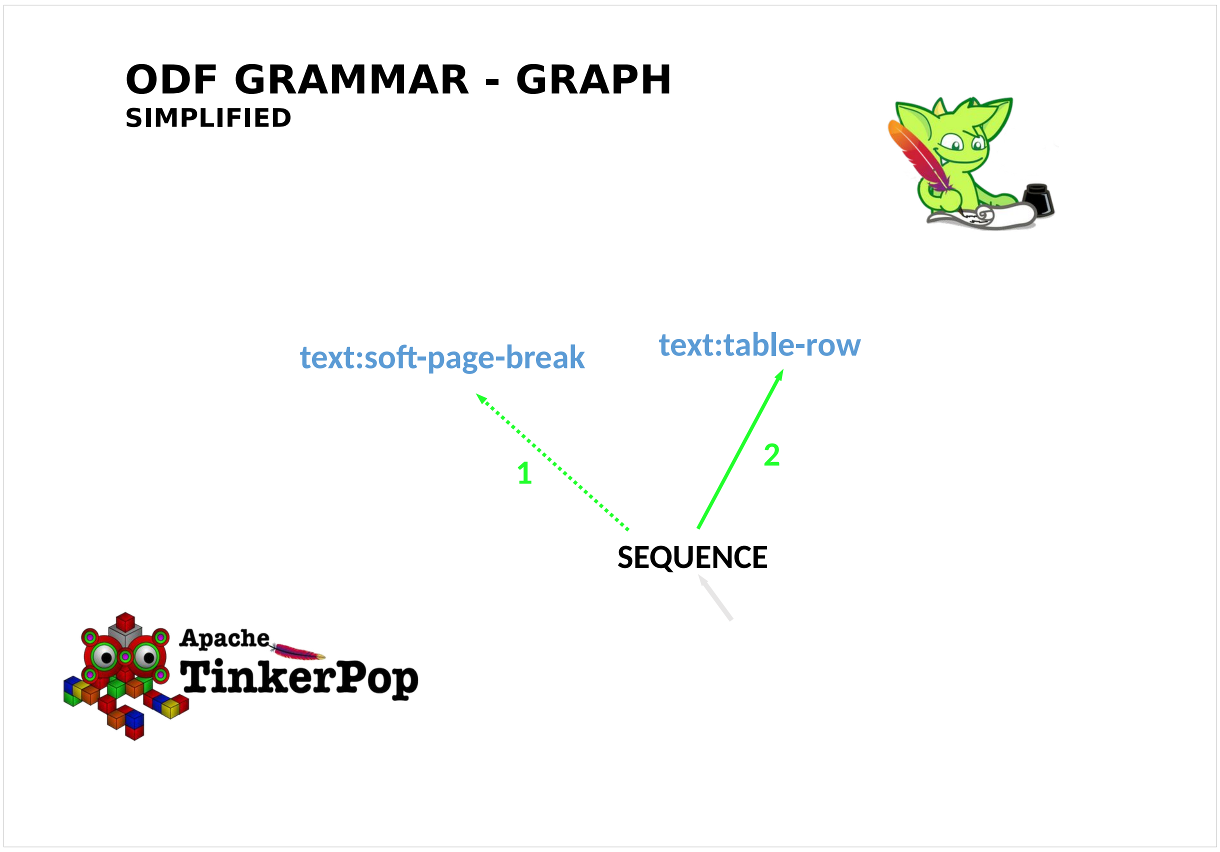
- https://groups.google.com/forum/#!searchin/gremlin-users/svante%7Csort:date/gremlin-users/P8MdzzlFtng/vYqYlukJAgAJ



ODF GRAMMAR - GRAPH

HARD TO READ

- https://groups.google.com/forum/#!searchin/gremlin-users/svante%7Csort:date/gremlin-users/P8MdzzlFtng/vYqYlukJAgAJ

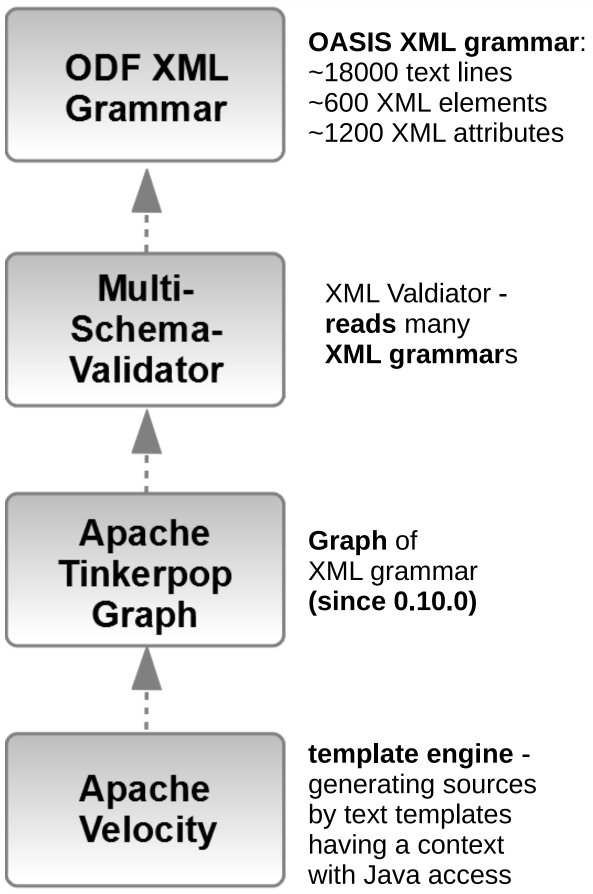


ODF GRAMMAR - GRAPH

SIMPLIFIED

Gremlin is worth the name, incredible time spending learning Gremlin Graph language to execute this simplification step..

ODFDOM Source Code Generator
Architecture


ODF XML Grammar
OASIS XML grammar: ~18000 text lines ~600 XML elements ~1200 XML attributes
Multi-Schema-Validator
XML Valdiator - reads many XML grammars
Apache Tinkerpop Graph
Graph of XML grammar (since 0.10.0)
Apache Velocity
template engine - generating sources by text templates having a context with Java access

In the very **first approach XSLT** was used to create from the ODF XML grammar the Java sources.

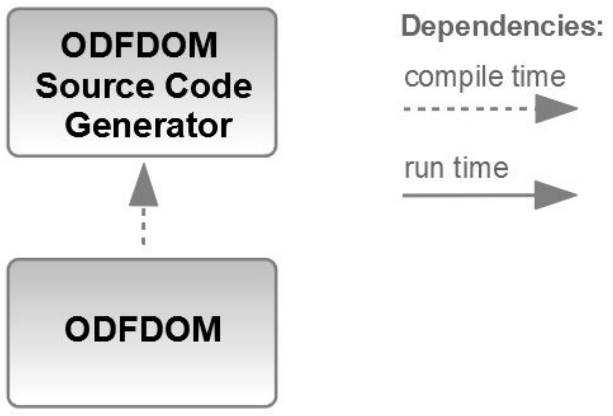
We **split the complexity** and
reused existing opensource software:
a) Multi-Schema-Validator (MVS) to read the XML grammar

b) NOW with a GRAPH representation of the ODF grammar

c) Apache Velocity Engine as Template engine

With ODF Toolkit 0.10.0 introduced common tree data structure to allow other tooling on the XML grammar

ODF Toolkit
Architecture



Similar to **JAXB for W3C schema** (grammar).
(Java XML Binding of JEE, works only for W3C schema)

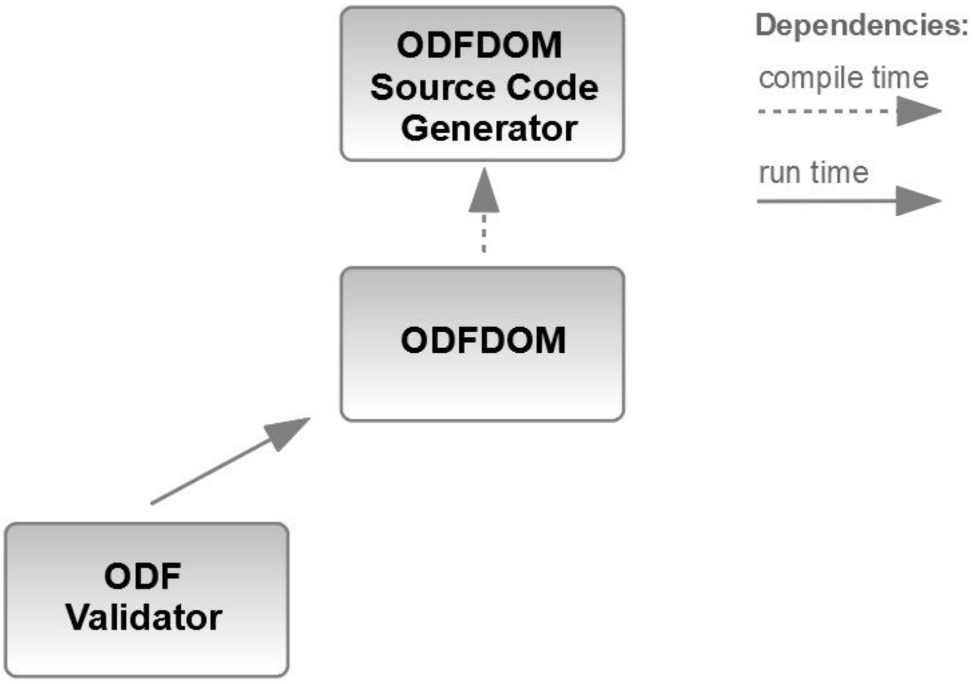
The **ODF grammar/schema** is used to **generate Java sources**.

Every ODF element and attribute is created as a **typed Java DOM class** in ODFDOM to ease developer to create valid ODF.

In the future, likely a second approach in **RUST**?
Allowing multi-threading and better memory handling.

ODF Toolkit

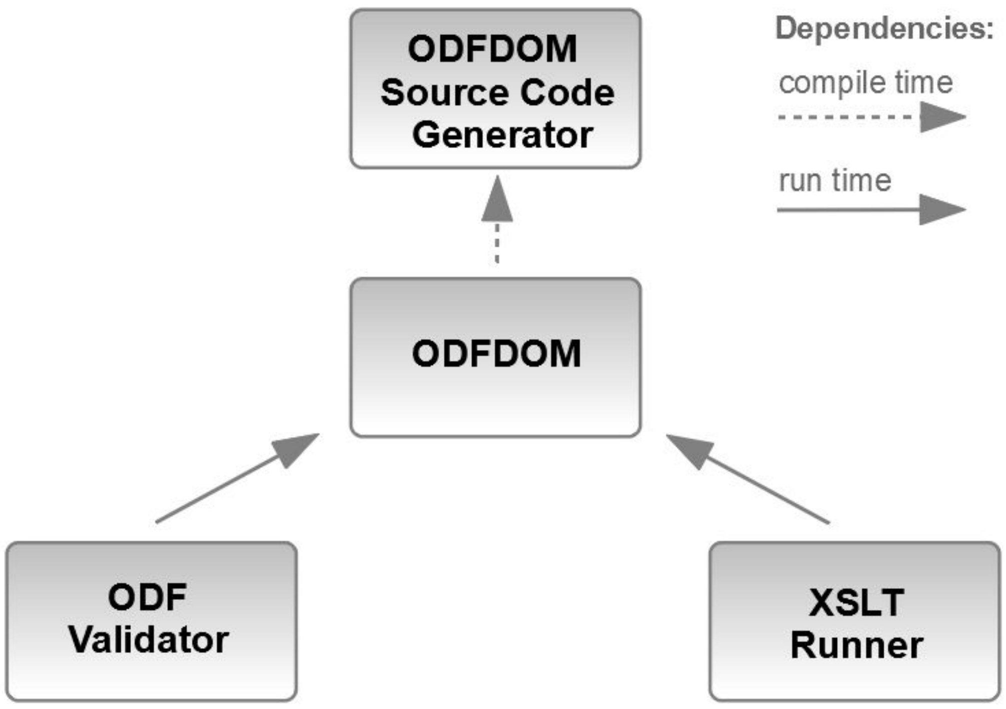
Architecture



ODF Validator relies on ODFDOM

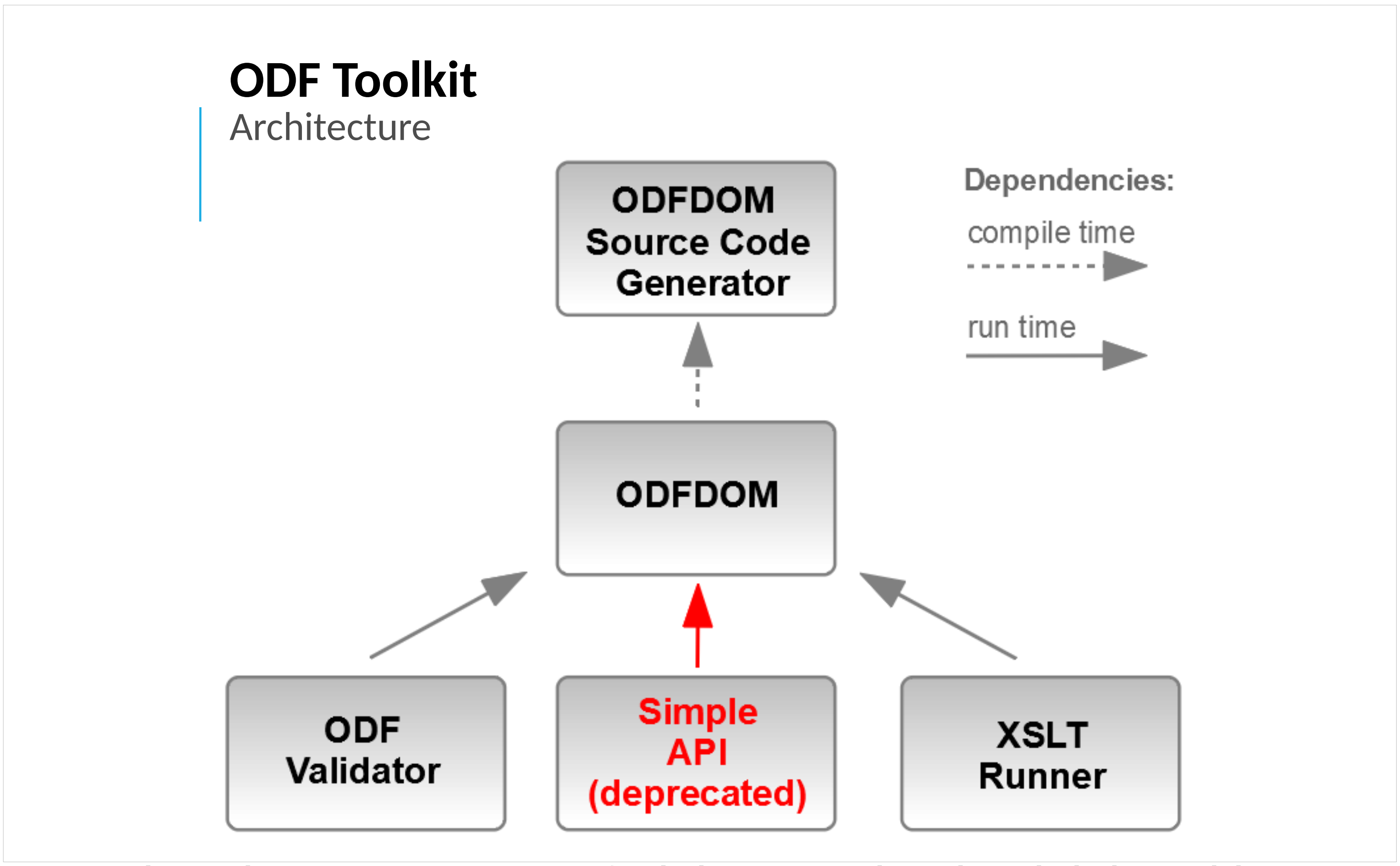
The ODF document is loaded by ODFDOM and error messages are gathered during loading from ODFDOM.

ODF Toolkit
Architecture



XSLT Runner relies on ODFDOM

The ODF document is loaded by ODFDOM and the XML streams are provided to the XSLT processor, references within the XML resolved into the ZIP.



ODF Toolkit

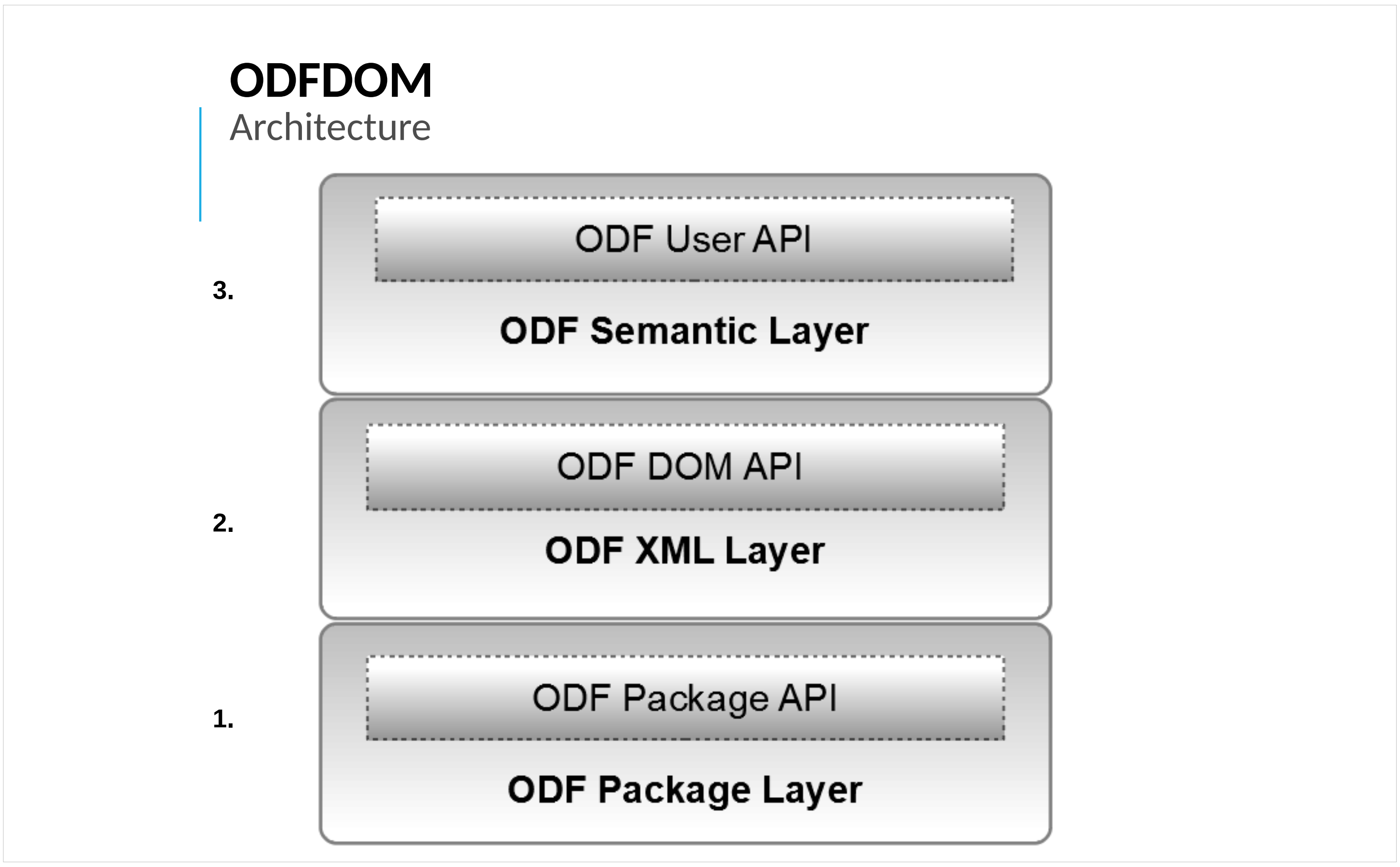
Architecture

The Simple API was a fork by IBM, but back joint with the ODF Toolkit at Apache.

The Simple API correct idea was to provide a high level user API not on the XML, but on the uers‘s semantics.

But it includes too much duplicated code from ODFDOM and was therefore deprecated in 0.9.0 and will not be part of 1.0.0

In addition, much more should be generated in ODFDOM instead written manually.

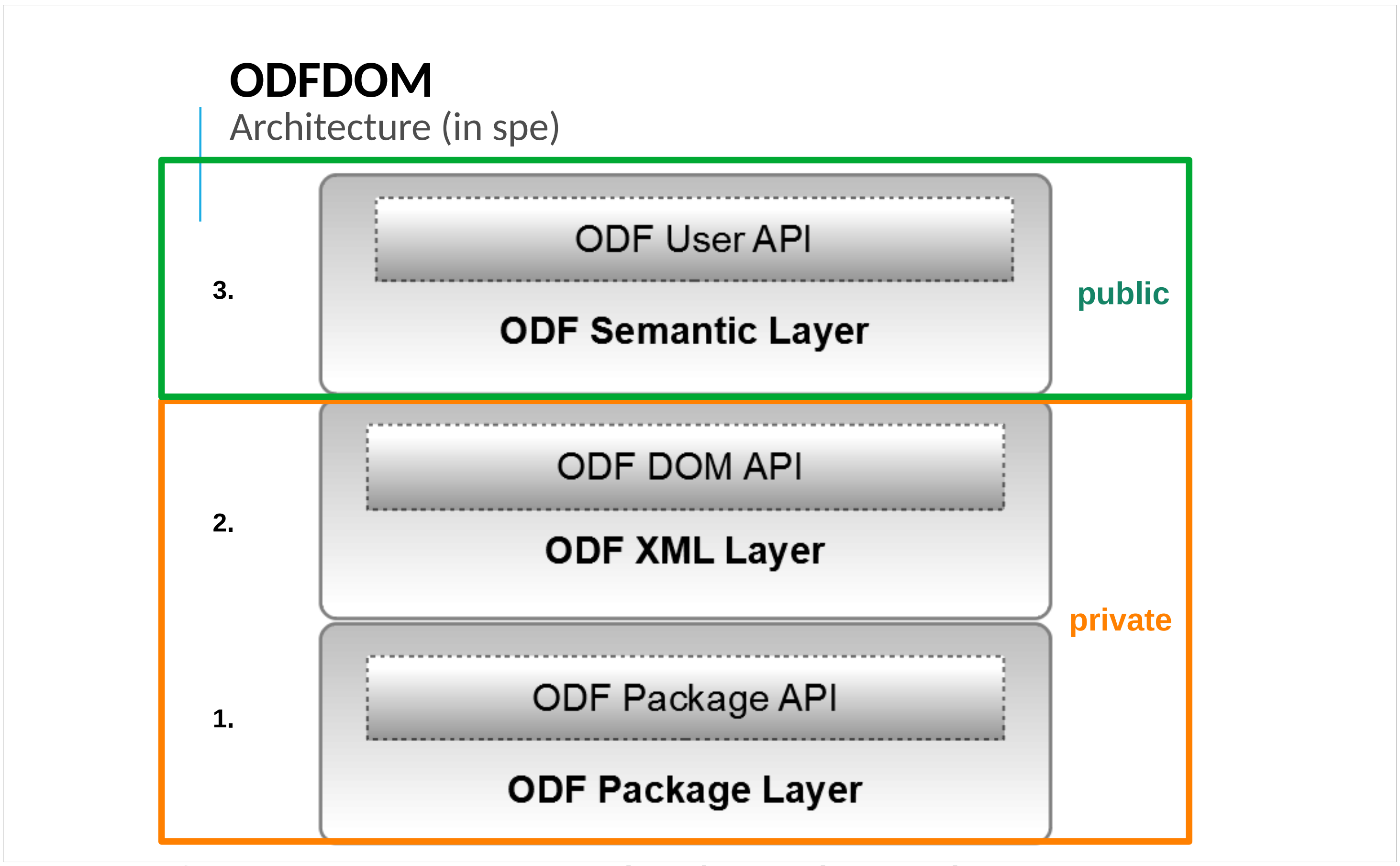


ODFDOM
Architecture

1. The lowest first API is the **Package API**.
Taking care of unzipping, add/read/delete ZIP‘s content and updating content table.
[**NOTE:** PKG API NOT OWN JAR DUE TO HARD CODED DOC LOADER DETECTION]
Providing:
a) **Document support** (being a directory with MIME TYPE in ODF)
b) **XML support** (access and caching the DOM)

2. DOM API is generated from the ODF XML Schema, should contain all implementation details and providing the corsett of XML validness#

3. User API currently hand written (in future mostly generated) should provide all functionality from easy user semantic perspective by accessing the underlying layers.



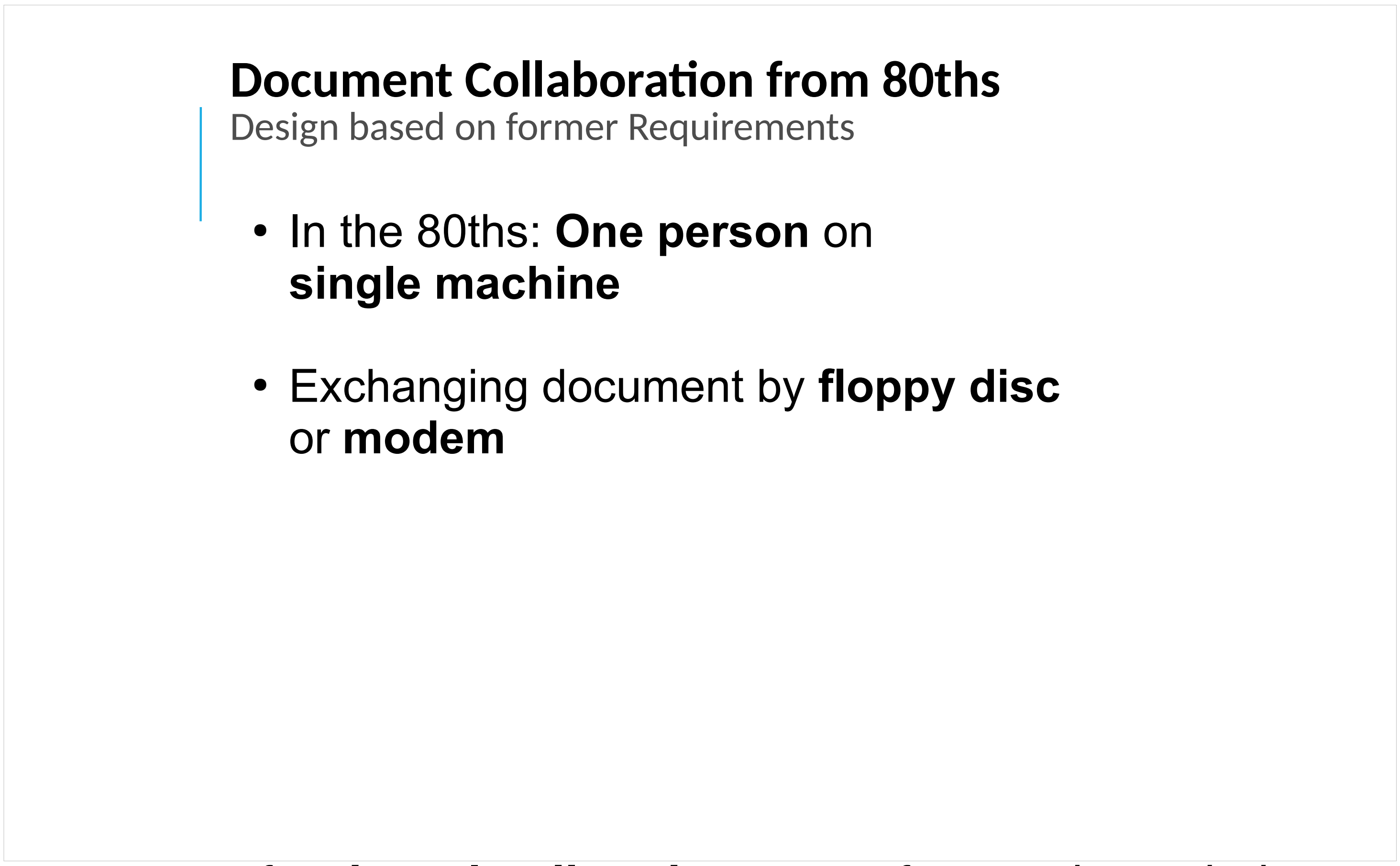
Unfortunately ODF applications do not have an interoperable RunTime Environment, like browsers having the DOM Model, which is the reason that JavaScript works across different browsers and ACID tests work (loading document shows level of support).

LibreOffice knows nothing about XML after loading.

There is a much better **interoperability over semantic**.

Every ODF applications allows add/modify/delete the same semantic entities (like paragraph, character, table, image, etc.) and their properties (bold, color, border width, etc.), we are working out the semantic model and its initial API, the **User API**.

Only **User API should be public** (or the other APIs public, but only an exchangeable implementation detail).

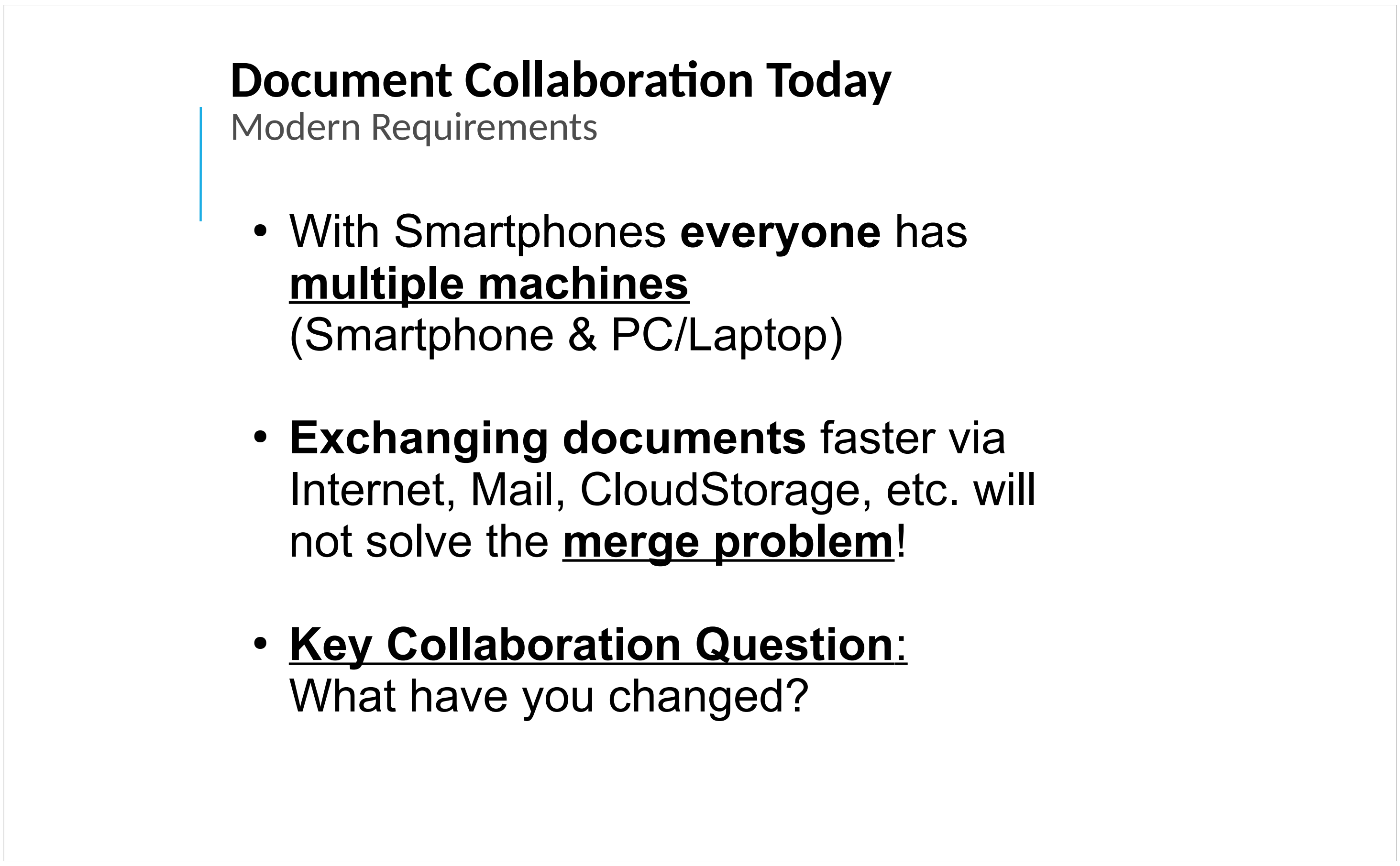


Document Collaboration from 80ths

Design based on former Requirements

- In the 80ths: **One person** on **single machine**
- Exchanging document by **floppy disc** or **modem**

Most of **today's leading document** formats have their **roots in the 80ths**. Their design was build upon requirements of these days: to represent the document state for one single machine or to exchange a document by floppy disc or modem. Often designed for a single purpose far more narrow than their current usage. New features were often accomplished by workarounds. For example, change-tracking of any office format does not track a defined interoperable change. Only the earlier state of the changed area is stored, to be swapped back in case of rejection.



Document Collaboration Today

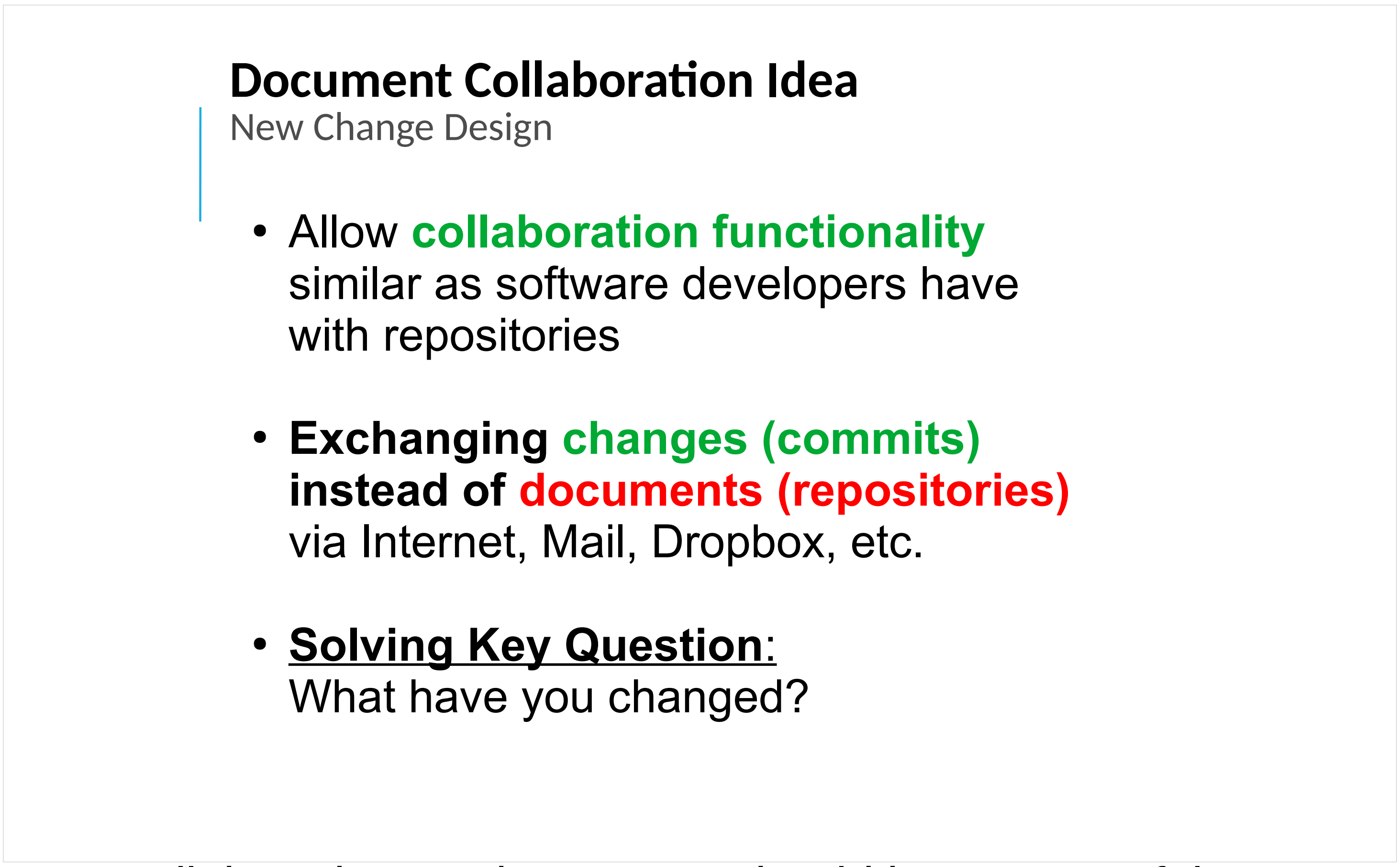
Modern Requirements

- With Smartphones **everyone** has **multiple machines**
  (Smartphone & PC/Laptop)

- **Exchanging documents** faster via Internet, Mail, CloudStorage, etc. will not solve the **merge problem**!

- **Key Collaboration Question**:
  What have you changed?

Nowadays, with the rise of mobile devices, online collaboration is ubiquitous and creates challenges when dealing with documents designed for an environment from the 80ths.



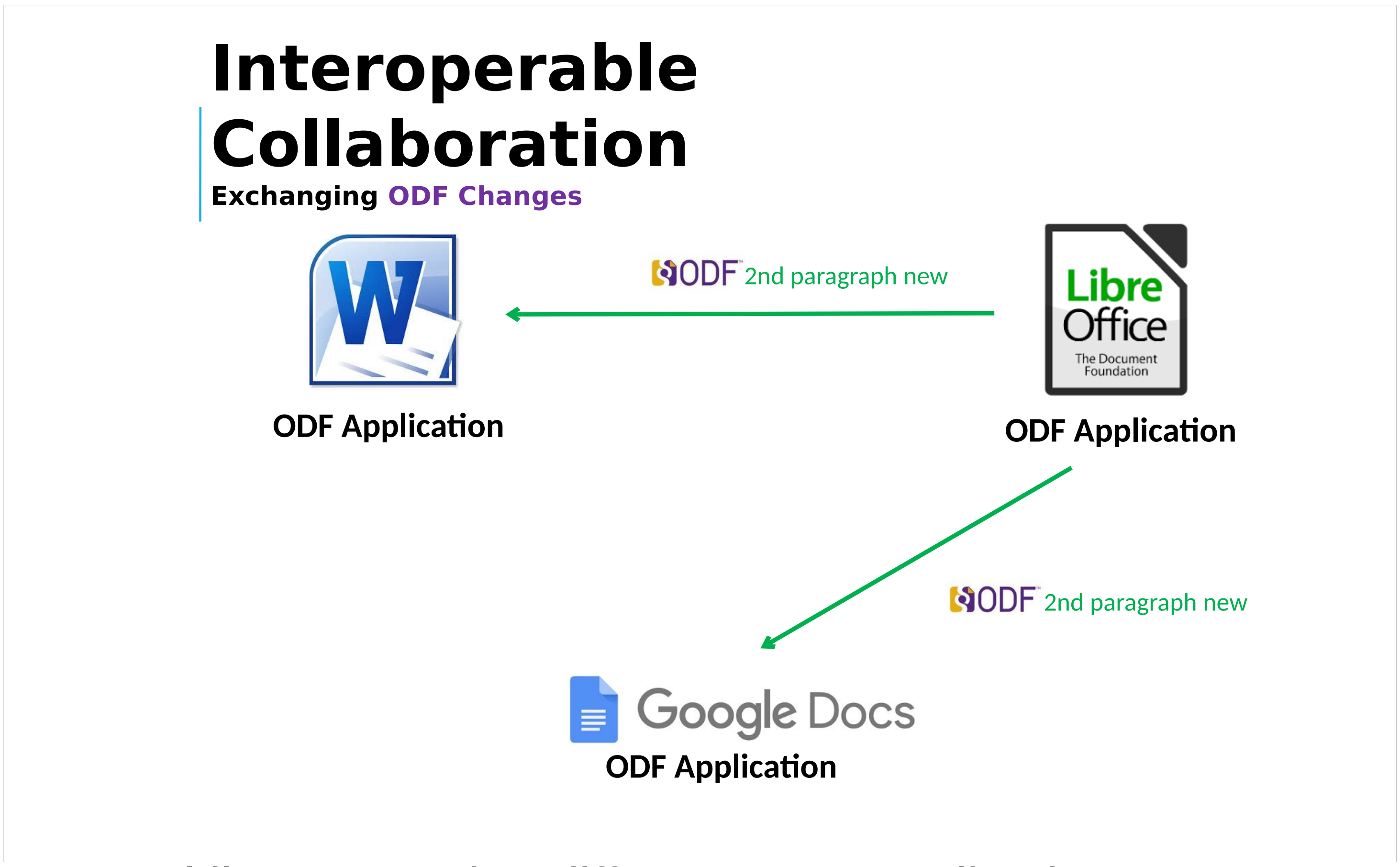
Document Collaboration Idea

New Change Design

- Allow **collaboration functionality** similar as software developers have with repositories
- **Exchanging changes (commits) instead of documents (repositories)** via Internet, Mail, Dropbox, etc.
- **Solving Key Question**: What have you changed?

Collaboration on documents should be as powerful as for software developers working on repositories (like GIT) .

Exchanging documents between editors for collaboration instead of exchanging the editors‘ changes is as efficient as software developers zipping their source code repositories and exchanging those. To be able to merge the changes of other editors into a single representive document, the changes of each editor have to be known. Today, an interoperable exchange of user changes between applications is unfortunately impossbile as world wide only full documents are being specified (e.g. HTML, ODF, OOXML, Docbook, etc.).



Interoperable Collaboration

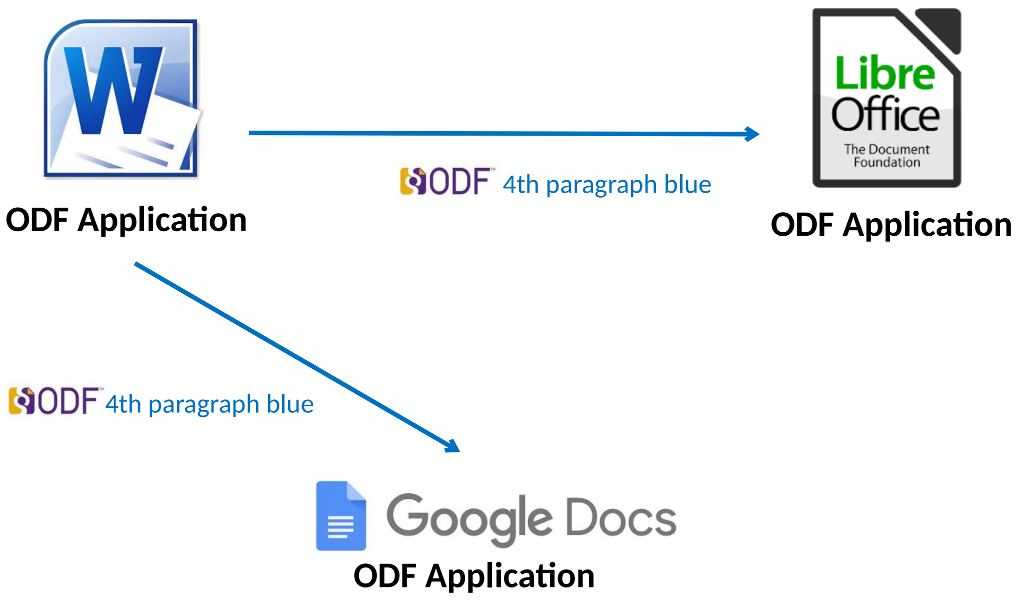
Exchanging ODF Changes

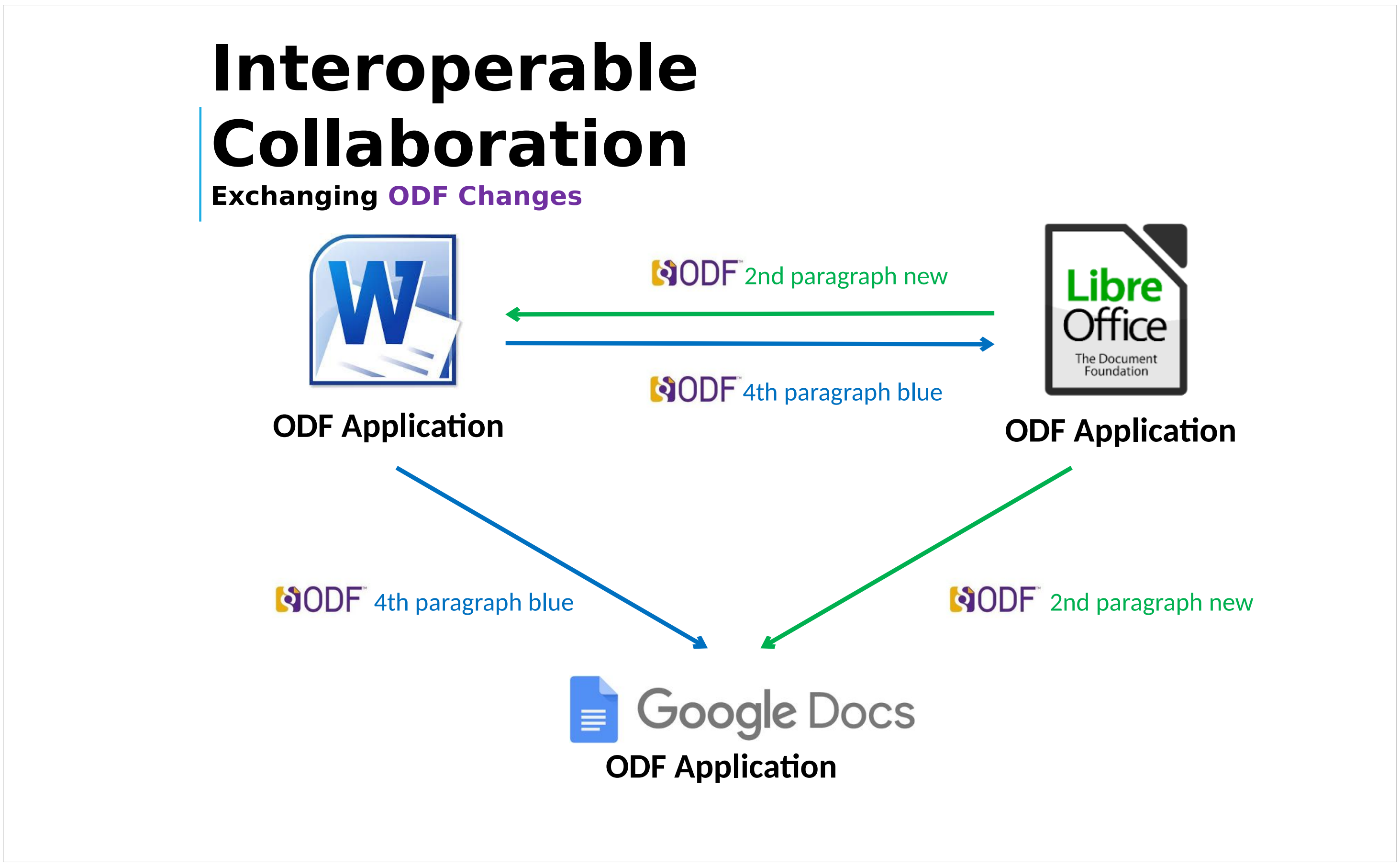
Multile users using different ODF applications exchanging no longer docs, but their changes!

Here they are already exchanging standardized OASIS ODF changes similar to semantic user changes.

Interoperable Collaboration

Exchanging ODF Changes

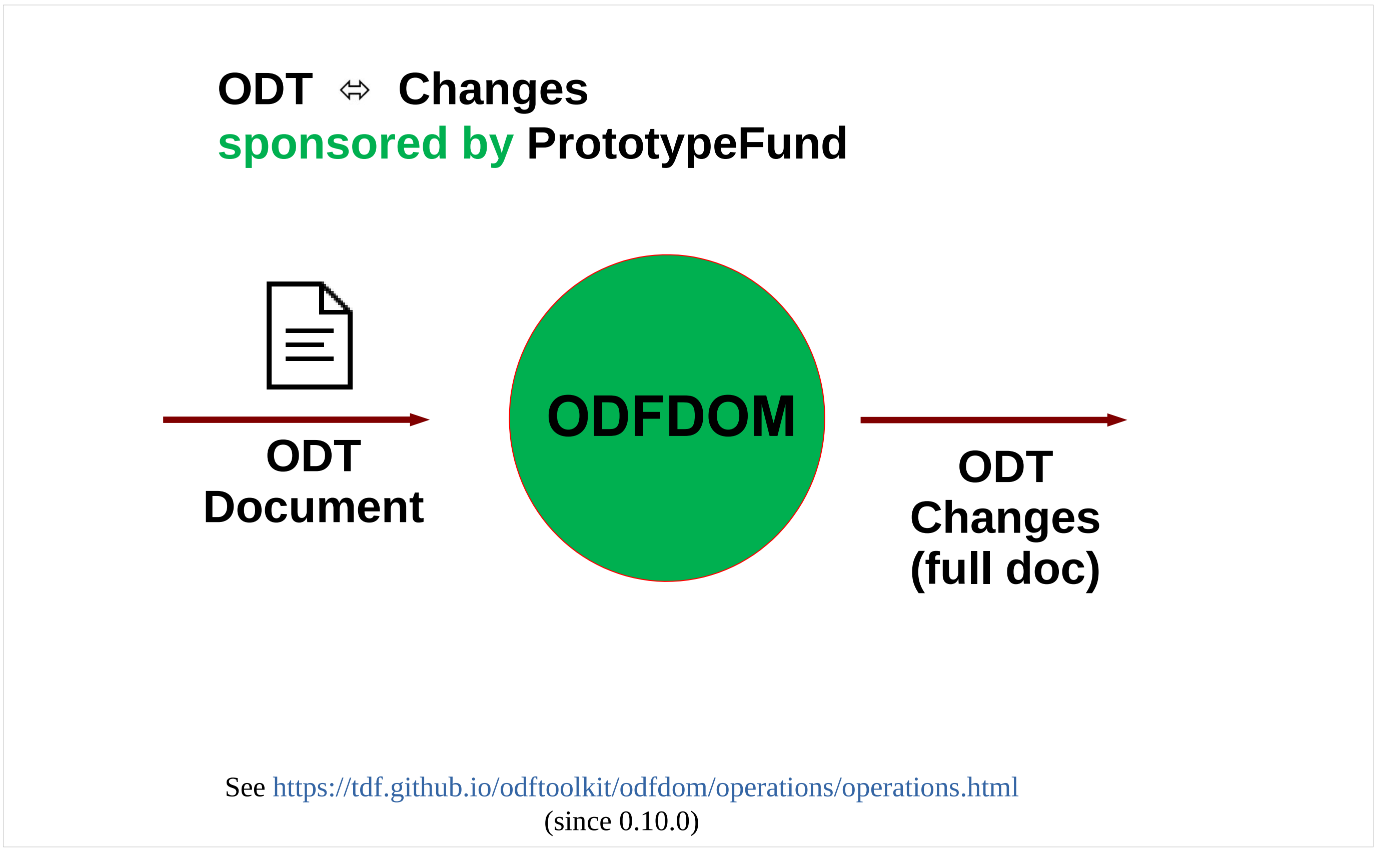




Interoperable Collaboration

Exchanging **ODF Changes**

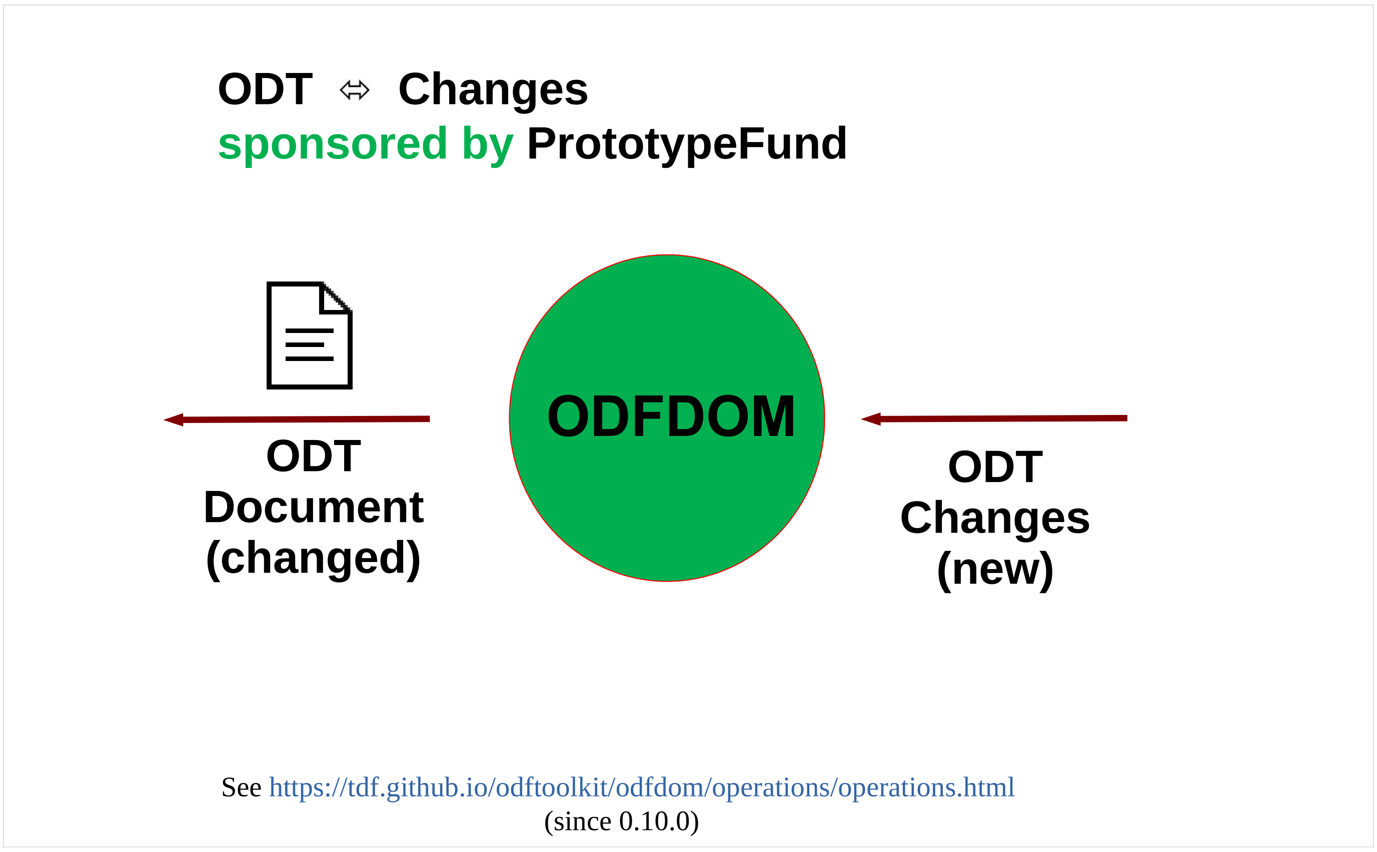
ODF CHANGES next EVOLUTINOARY STEP



ODT ⬄ Changes
sponsored by PrototypeFund

See https://tdf.github.io/odftoolkit/odfdom/operations/operations.html
(since 0.10.0)

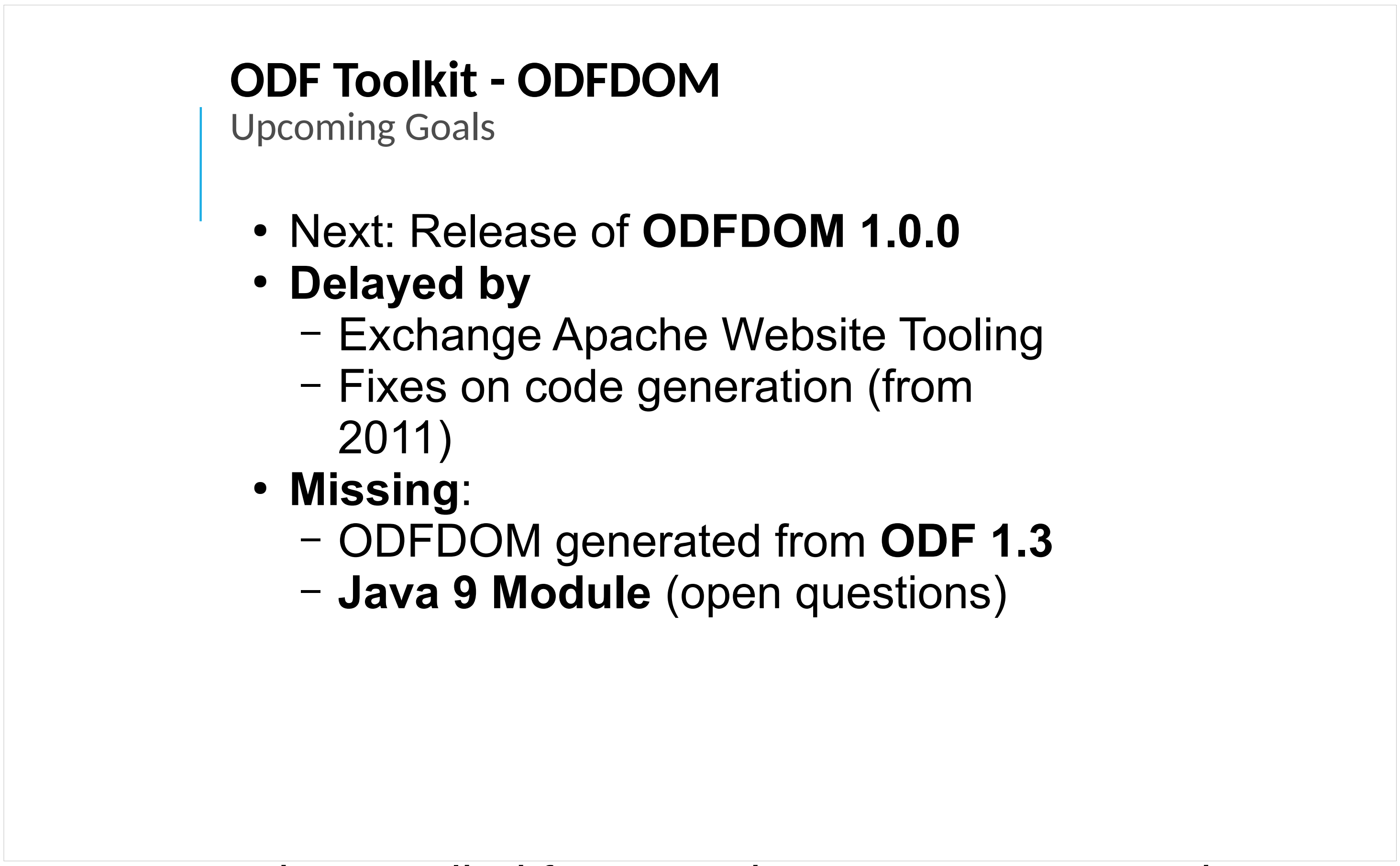
Founded by the German PrototypeFund soon in ODF Toolkit 1.0.0 an ODT is transformed to its equivalent list of user changes (JSON format)



ODT ⇔ Changes
sponsored by PrototypeFund

See https://tdf.github.io/odftoolkit/odfdom/operations/operations.html
(since 0.10.0)

New user changes (JSON format) applied to the document can be merged into the document by ODFDOM (ver. 1.0.0) and saved back as altered ODT.

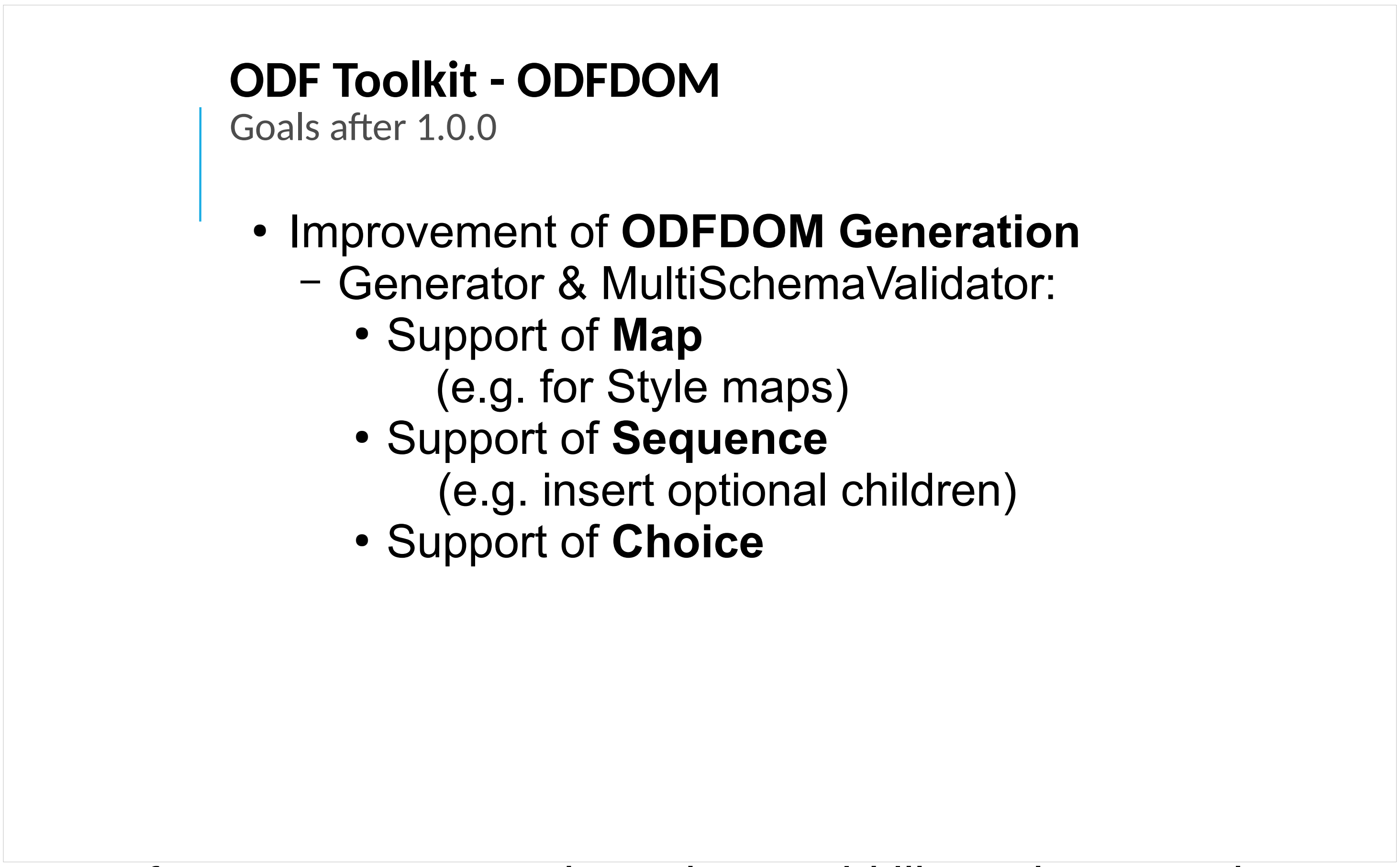


ODF Toolkit - ODFDOM

Upcoming Goals

- Next: Release of **ODFDOM 1.0.0**
- **Delayed by**
  - Exchange Apache Website Tooling
  - Fixes on code generation (from 2011)
- **Missing**:
  - ODFDOM generated from **ODF 1.3**
  - **Java 9 Module** (open questions)

Svante has applied for next winter‘s PrototypeFund



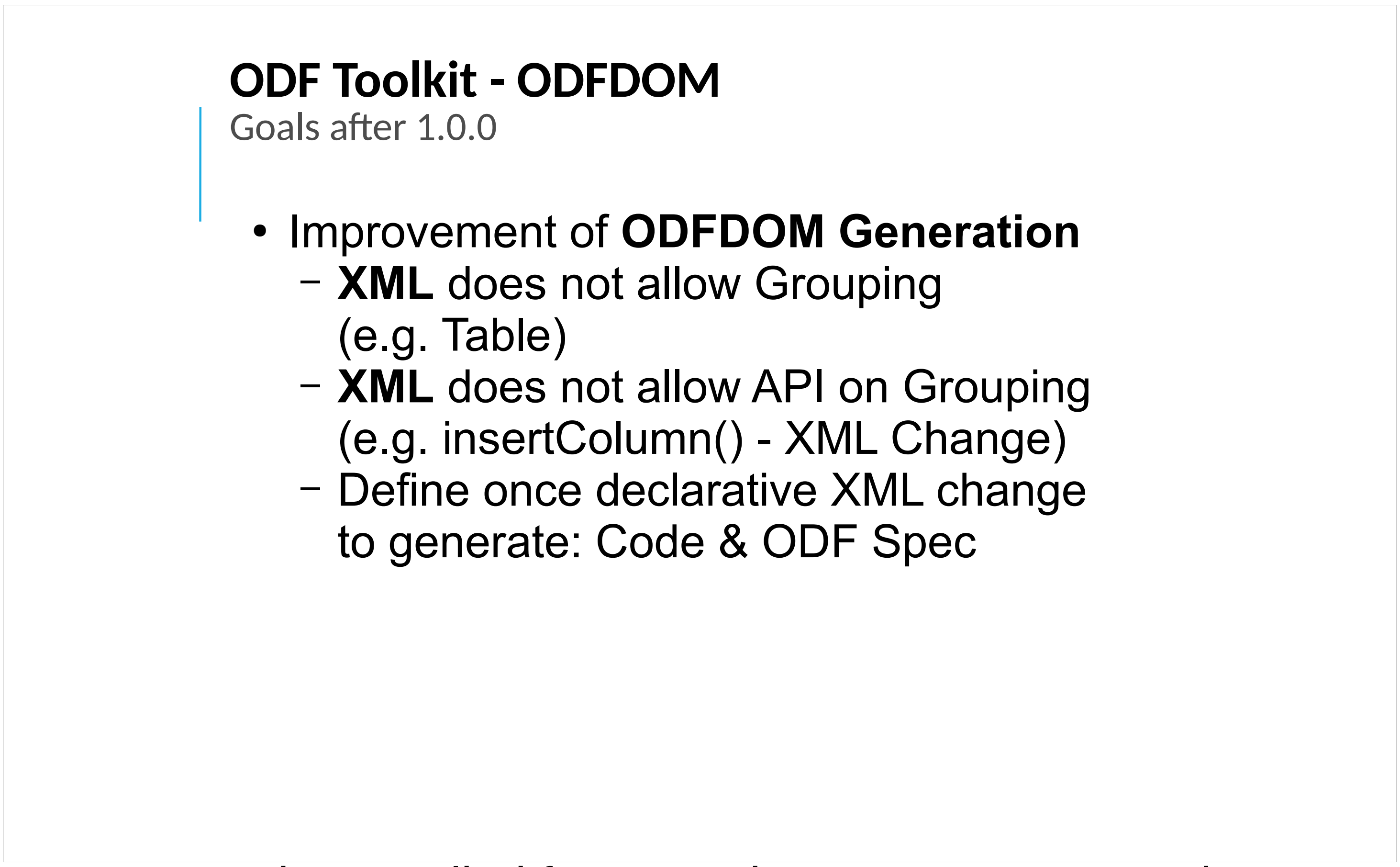
ODF Toolkit - ODFDOM

Goals after 1.0.0

- Improvement of **ODFDOM Generation**
  - Generator & MultiSchemaValidator:
    - Support of **Map**
      (e.g. for Style maps)
    - Support of **Sequence**
      (e.g. insert optional children)
    - Support of **Choice**

After 1.0.0 on a next branch I would like to improve the ODFDOM generation

Getting more out-of-the-box
from the ODF RNG Grammar
Less manual programming



ODF Toolkit - ODFDOM

Goals after 1.0.0

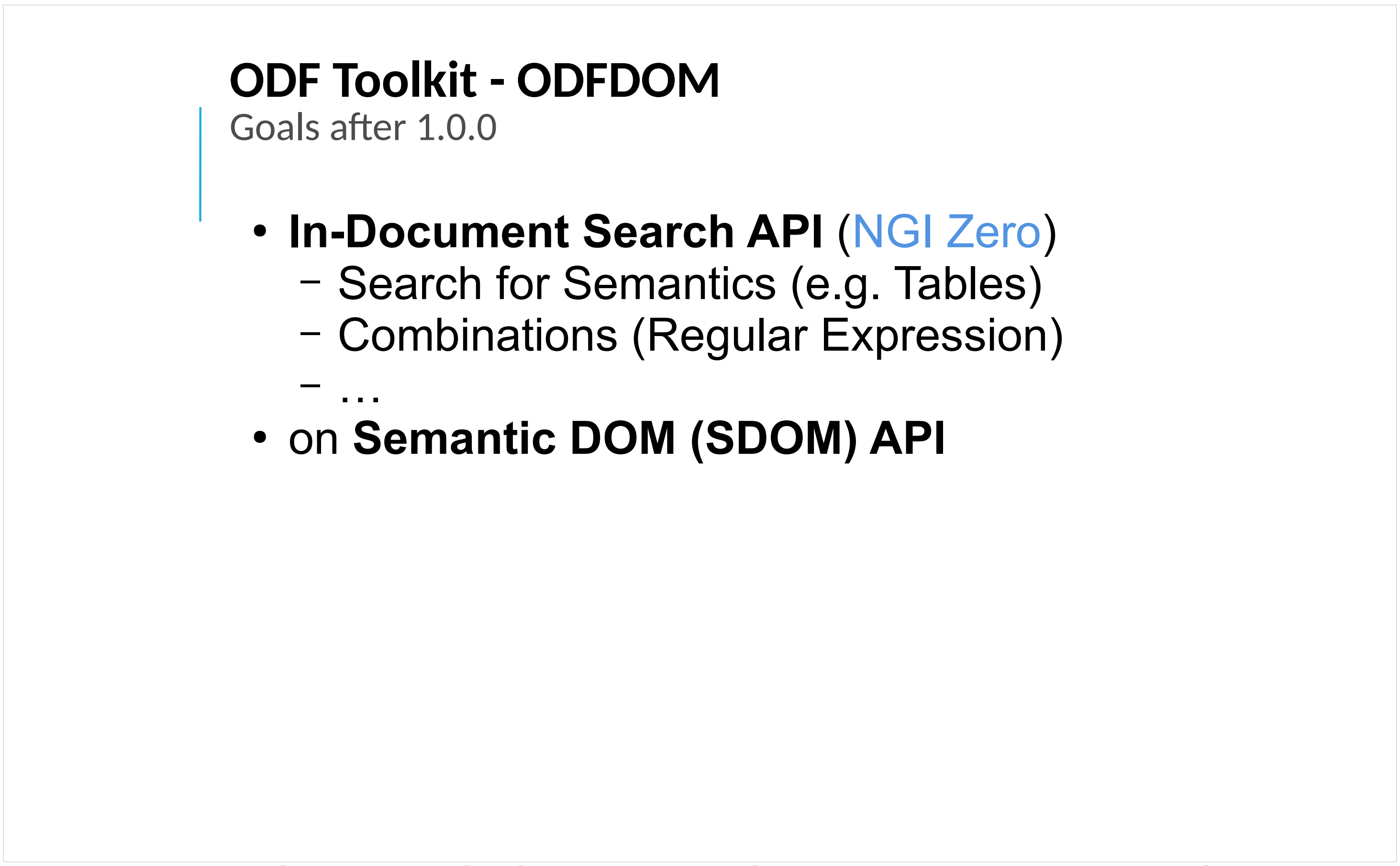
- Improvement of **ODFDOM Generation**
  - **XML** does not allow Grouping (e.g. Table)
  - **XML** does not allow API on Grouping (e.g. insertColumn() - XML Change)
  - Define once declarative XML change to generate: Code & ODF Spec

Svante has applied for next winter‘s PrototypeFund project to connect various front-end editors to make this idea useable by end-users.

The Emacs would load the paragraphs (as lines) and text of full featured ODT documents, editing it and saving Emacs user changes back without destroying the document.

The HTML editor CKEditor 5, which uses changes as its interior core, will be able to load, edit and save ODT files similar to Emacs without destroying unkown functionality.

LibreOffice is planned to extended by the lead of Thorsten Behrens to have a prototype for changes.



ODF Toolkit - ODFDOM

Goals after 1.0.0

- **In-Document Search API** (NGI Zero)
  - Search for Semantics (e.g. Tables)
  - Combinations (Regular Expression)
  - …
- on **Semantic DOM (SDOM) API**

Svante has applied for next winter‘s PrototypeFund project to connect various front-end editors to make this idea useable by end-users.

The Emacs would load the paragraphs (as lines) and text of full featured ODT documents, editing it and saving Emacs user changes back without destroying the document.

The HTML editor CKEditor 5, which uses changes as its interior core, will be able to load, edit and save ODT files similar to Emacs without destroying unkown functionality.

LibreOffice is planned to extended by the lead of Thorsten Behrens to have a prototype for changes.

ODF Toolkit

Resources

- **Website:**
  https://odftoolkit.org/
  https://tdf.github.io/odftoolkit/docs/ (latest)
- **Sources**:
  https://github.com/tdf/odftoolkit
- **Online Validator (hosted by TDF)**
  https://odfvalidator.org/

- **ODF Specification**
  http://docs.oasis-open.org/office/OpenDocument/v1.3/os/
- **ODF Specification Tooling**
  https://github.com/oasis-tcs/odf-tc/