

# AI for XML Development

## Advantages and Challenges

Octavian Nadolu, Syncro Soft

octavian\_nadolu@oxygenxml.com

@OctavianNadolu

© 2024 Syncro Soft SRL. All rights reserved

## About me

---

Project Manager at Syncro Soft

[octavian.nadolu@oxygenxml.com](mailto:octavian.nadolu@oxygenxml.com)

- Over 20 years of experience in XML technology
- Contributed to various XML-related open source projects
- Speaker at multiple conferences
- Co-Editor of Schematron QuickFix specification developed by a W3C community group



schematron  
Structured  
editing  
XML  
review  
XQuery  
Publish  
PDF  
IDREFS  
WebDAV  
DTD DocBook  
oxygen  
authoring  
XML Editor  
Single  
Source  
Datab  
XHTML  
Cha  
Col  
We  
JS  
KML  
XSLT  
SVN  
JSON  
SVG  
XSD SCHXSD  
XPRRNCFO  
frameworks  
Profiling  
WSDL  
styles  
visual  
WebHelp  
DITA  
TEI  
XSL  
PHP  
Ant  
Js

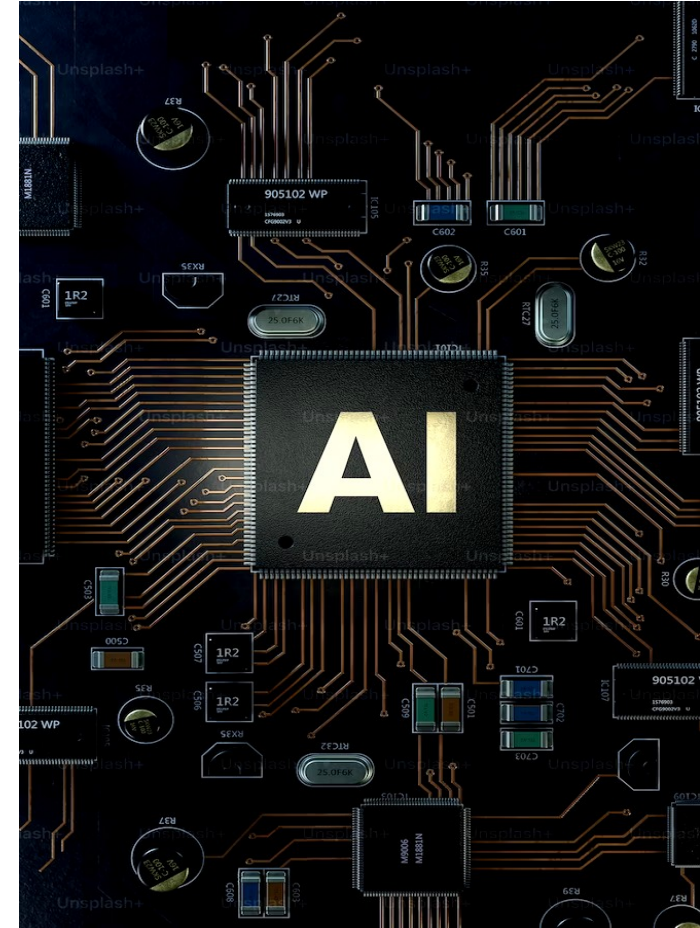
# Agenda

- Artificial Intelligence (AI)
- How AI Understands XML
- Using AI to Generate XML Content
- Using AI to Add XML Markup
- AI in Developing XSLT, XSD, and Schematron
- Transform XML using AI and XPath Functions
- Advantages and Challenges



# What is Artificial Intelligence?

Artificial Intelligence (AI) is a branch of computer science dealing with the simulation of intelligent behavior in computers.



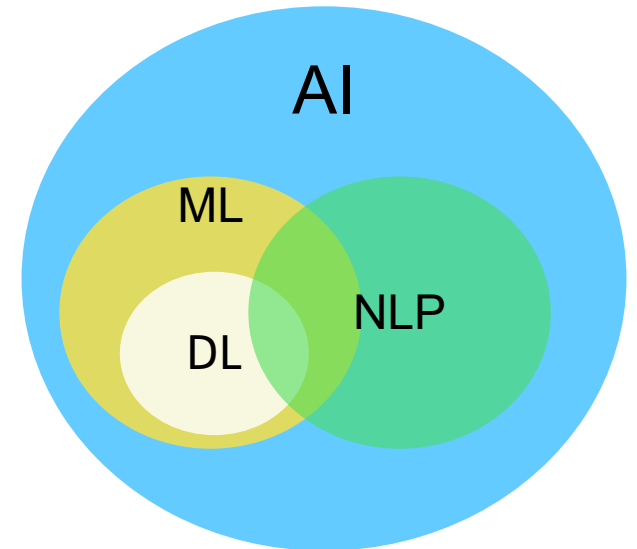
# Artificial Intelligence History

- AI history dates back to ancient times
- In 1950s AI began to take shape - researchers began to use computers to try and simulate human intelligence
- AI program by mathematician Alan Turing
- Expert system by Edward Feigenbaum in the 1970s and the emergence of neural networks in the 1980s
- Today, AI is used to automate processes, improve efficiency, and solve complex problems



# AI in Natural Language Processing

- Natural language processing (NLP) is a subfield of AI that focuses on enabling machines to understand and generate human language
- Machine learning (ML) involves training algorithms to learn patterns in data
- Deep learning (DL) is a type of machine learning that uses neural networks



# Transformers

- A **transformer** is a deep learning model (neural network)
- Transforms **input sequences** into **output sequences**
- The transformer architecture is based on the idea of **self-attention**
- Introduced in a research paper titled "*Attention Is All You Need*" in 2017

# Large Language Models

- Uses **transformer** architecture
- Trained on a **large corpus of data**

## Examples:

- ChatGPT (GPT-4 model) - *developed by OpenAI*
- Bard - *developed by Google*
- Bing - *developed by Microsoft*
- Llama Chat - *developed by Meta*
- *Claude, Mistral, Hugging Face, Amazon Bedrock*



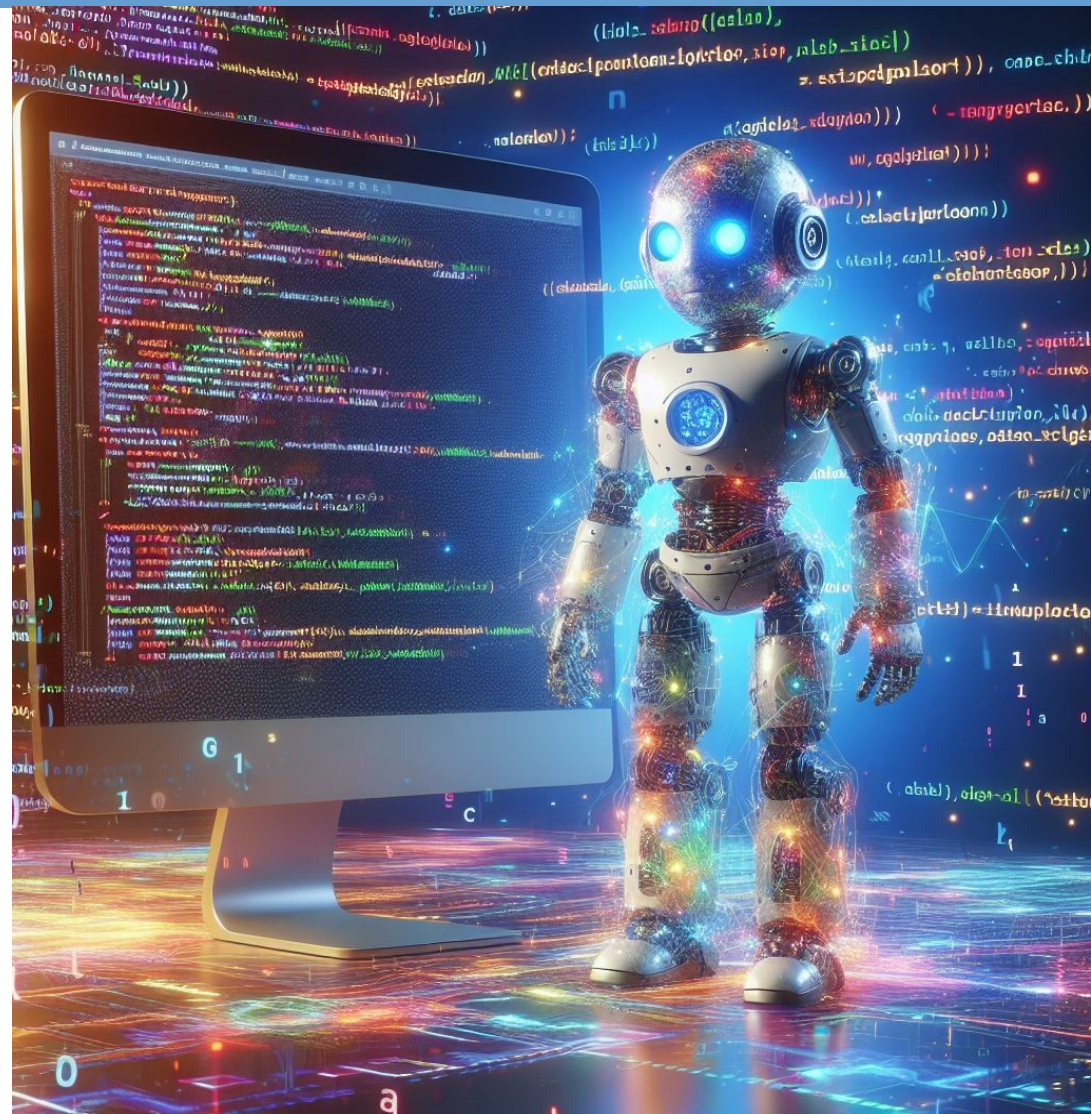
# AI Application for Text Processing

Trained language models that are very good at understanding and generating text

- Text Summarization
- Natural Language Processing
- Text Generation
- Machine Translation
- Text Classification

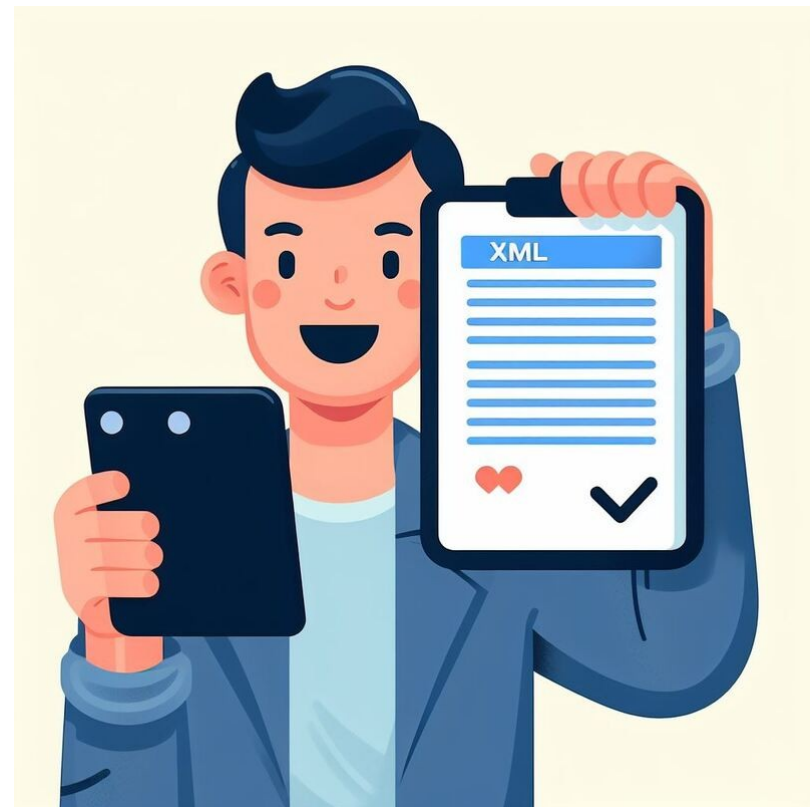
# AI and XML

- XML in a format that is both human-readable and machine-readable
- AI can understand XML markup
- AI can generate XML markup
- AI can add XML markup to the content



# Using AI to Generate XML Content

- Use AI to generate XML markup
- Create an XML document from a description
- Create XML content based on a schema
- Generate Dita, DocBook, TEI documents



# Generate XML Content using AI

- Example of prompt that creates a Topic

Generate a new DITA XML topic based on following description:  
Technical documentation for a car



```
<topic id="topic_car_technical_documentation">
  <title>Technical Documentation for Car</title>
  <body>
    <section>
      <title>Introduction</title>
      <p>This document provides comprehensive technical information about the car,
        including its specifications, features, and maintenance guidelines.</p>
    </section>
    ...
  </body>
</topic>
```

# Generate XML Content using AI

- Example of prompt creates a table

Act as an XML developer, Respond with no additional information,  
Create a table from the following content:

- Go to Definition - Shows the definition for...
- Open Schema - Opens the selected schema..
- Edit Attributes - Allows you to edit the ...



```
<table>  
  <tr><th>Feature</th><th>Description</th></tr>  
  <tr><td>Go to Definition</td><td>Shows the definition for...</td></tr>  
  <tr><td>Open Schema</td><td>Opens the selected schema...</td></tr>  
  <tr><td>Edit Attributes</td><td>Allows you to edit the...</td></tr>  
</table>
```

# Using AI to Add XML Markup

- You can use AI to add markup to your content
- XML markup can be added depending on the context
- Correct documents to respect a specific schema



# Add XML Markup using AI

- Example of prompt that generates an unordered list from a set of phrases

Act as an XML developer. Respond with no additional information,  
Create a list from the phrases from the paragraph

```
<p> - Is active/passive voice used in the description?  
- Is this written according to the styleguide?  
- Does the to pic answers to this "question"?  
- Is spelling and grammar correct? </p>
```



```
<p><ul>  
  <li>Is active/passive voice used in the description?</li>  
  <li>Is this written according to the styleguide?</li>  
  <li>Does the to pic answers to this "question"? </li>  
  <li>Is spelling and grammar correct?</li>  
</ul></p>
```

# Add XML Markup using AI

Act as an XML developer. Respond with no additional information, Add the missing cells from the table.

entry Media entry	entry Description entry		
entry mp3 entry	entry Moving Picture entry	entry audio entry	
entry wav entry			
entry pcm entry	entry audio entry	entry Width entry	
entry embedded video entry	entry Embedded Iframe Code entry	entry iframe entry	entry Width & Height entry



entry Media entry	entry Description entry	entry entry	entry entry
entry mp3 entry	entry Moving Picture entry	entry audio entry	entry entry
entry wav entry	entry entry	entry entry	entry entry
entry pcm entry	entry audio entry	entry Width entry	entry entry
entry embedded video entry	entry Embedded Iframe Code entry	entry iframe entry	entry Width & Height entry



# AI and XML

- You can generate XML content using AI
- The prompt is very important
- Different AI models can give different results
- The result is not always the same for the same prompt

# Prompt for AI

- A prompt represents instructions and context given to a language model to perform a task

*“Act as an XML developer. Respond with no additional information. Create a table from the following content: ”*

- Techniques for creating a prompt:
  - *Adapted persona: “Act as an XML developer”*
  - Clearly defining the goal or objective: *“Create a table from the following content”*
  - Refine the result: *“Respond with no additional information.”*
  - Provide specific context (schema): *“Create a DITA table ...”, “Create a table that respects the following schema”*
  - Provide additional information on request:
    - *“Function that returns the schema of the document”* → schema
    - *“Function that returns the location in the document”* → XPath location

# AI in Developing XSLT, XSD, and Schematron

- AI can be used for developing
- You can use it for example to:
  - Document code
  - Explain code
  - Generate code
  - Find problems
  - Refactor code
  - Start a conversation for a given code

WITH AI-POWERED XSLT, XSD, AND SCHEMATRON

**Revolutionize Your  
XML Development**

# AI and XSL

- Generate the documentation for XSLT

```
<xsl:template match="timeResolution" mode="header">
  <xsl:call-template name="listHeader">
    <xsl:with-param name="name" select="'Time Resolution'"/>
  </xsl:call-template>
</xsl:template>
```



## Document Code

Create documentation for the code and include it as a comment in the document.

```
<!-- This XSLT code template matches "timeResolution" in header mode and
calls the "listHeader" template with the parameter "Time Resolution". -->
```

# AI and Schematron

- Explain the Schematron code

## Explain Code

Explain the code in the chat window

```
<sch:schema xmlns:sch="http://purl.oclc.org/dsdl/schematron">
  <sch:pattern id="NumberOfcharactersInAnAbbreviation">
    <sch:rule context="BBB">
      <sch:report test="string-length(@bbb) &lt; 2">There is not enough letters in the
        abbreviation</sch:report>
      <sch:report test="string-length(@bbb) > 3">There is too much letters in the abbreviation
        </sch:report>
    </sch:rule>
  </sch:pattern>
</sch:schema>
```



In this code snippet, a rule is defined within a pattern to validate the length of an abbreviation denoted by the attribute `@bbb` within an element `BBB`. Here's a breakdown of the code:

- The schema is defined with the namespace `http://purl.oclc.org/dsdl/schematron`.
- A pattern with the id `NumberOfcharactersInAnAbbreviation` is defined.
- Within this pattern, a rule is specified with the context set to `BBB`, which means this rule will be applied to elements named `BBB`.
- Two reports are defined within the rule:
  - The first report checks if the string length of the attribute `@bbb` is less than 2 and provides a corresponding error message.
  - The second report checks if the string length of the attribute `@bbb` is greater than 3 and provides a corresponding error message.

# Create Schematron using AI

**Generate code**

An assert that verifies the number of words to be 10



```
<sch:assert test="count(tokenize(., '\s+')) = 10">  
    There should be exactly 10.</sch:assert>
```

**Generate code**

An assert that verifies if there is an email in text



```
<sch:assert test="matches(., '\b[A-Za-z0-9._%+-]+\@[A-Za-z0-9.-]+\.[A-Z|a-z]{2,}\b')">  
    There is no email in the text</sch:assert>
```

# AI and XSD

- Generate XSD code

```
<xs:complexType name="PurchaseOrderType">
  <xs:sequence>
    <xs:element name="CustomerName" type="xs:string"/>
    <xs:element name="OrderDate" type="xs:date"/>
    <xs:element name="Items" type="ItemsType"/>
  </xs:sequence>
</xs:complexType>

<xs:complexType name="ItemsType">
  <xs:sequence>
    <xs:element name="Item" type="ItemType" maxOccurs="unbounded"/>
  </xs:sequence>
</xs:complexType>

<xs:complexType name="ItemType">
  <xs:sequence>
    <xs:element name="ProductName" type="xs:string"/>
    <xs:element name="Quantity" type="xs:integer"/>
    <xs:element name="Price" type="xs:decimal"/>
  </xs:sequence>
</xs:complexType>
```

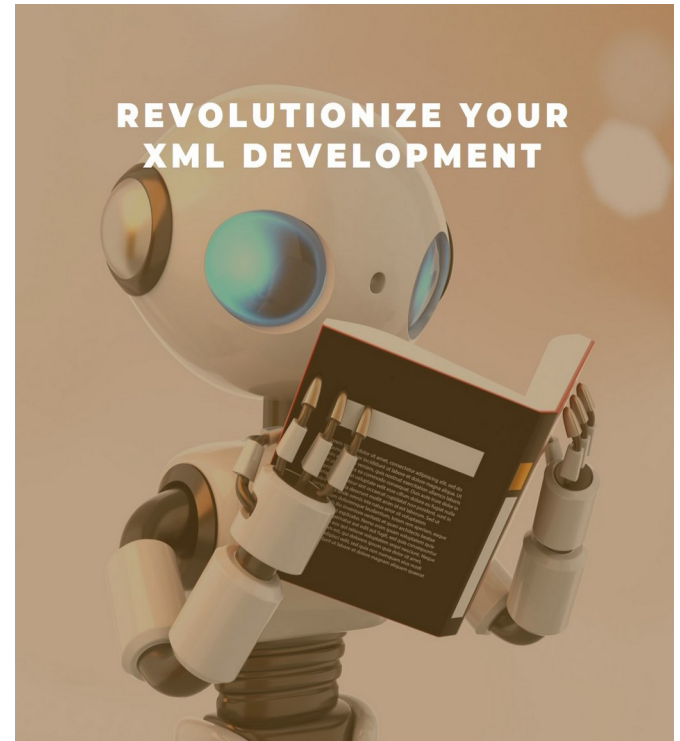
## Generate code

A complex type that defines a purchase order



# Powerful AI for Development

- Provide document context for the actions
- Provide associated schema or XML
- Provide referred components
- Reuse code when generating new content





# Automate AI Actions

- Use the AI engine API to perform complex actions
- Automate the process



# Implementation of AI API

- Using XSLT functions

```
<xsl:function name="ai:chatGPT">
  <xsl:param name="instruction"/>
  <xsl:param name="content"/>
  <xsl:variable name="url" select="https://api.chatgpt.com/v1/chatbot/question"/>
  <xsl:variable name="requestBody" select="concat('{', '&quot;text&quot;:&quot;', $instruction,
$content, '&quot;', '}')"/>
  <xsl:variable name="response" select="document(concat($url, '?apiKey=',
'&lt;your_api_key>'))//response"/>
  <xsl:sequence select="$response"/>
  .....
</xsl:function>
```

- Using extension functions

- xs:boolean **ai:verify-content**(xs:string instruction, xs:string content)
- xs:string **ai:transform-content**(xs:string instruction, xs:string content)

# Functions Built-in Prompt

- A prompt represents instructions and context given to a language model to perform a task
- Functions can provide a specific built-in prompt
  - `ai:verify-content(instruction, content)`  
*“You are a technical writer and you need to verify the following and respond with true or false:” + Is active voice used in the description? + content*
  - `ai:transform-content(instruction, content)`  
*“You are a developer and you need perform the following task:” + Rephrase to use active voice + content*

# AI Functions Usage

- Use the AI functions from XSL and XQuery
  - Transform content using AI
  - Create refactoring actions based on AI
- Use the AI functions from Schematron and SQF
  - Verify document content automatically using AI
  - Correct problems in document using AI

# Transform content using XSL and AI

- Generate the documentation for all templates and functions

```
<xsl:template match="xsl:template | xsl:function">
  <xsl:comment><xsl:value-of select="ai:transform-content(
    'As a developer, create a single phrase of documentation for the provided XSL content.', .)"/>
  </xsl:comment>

  <xsl:copy>
    <xsl:apply-templates select="node() | @*"/>
  </xsl:copy>
</xsl:template>
```

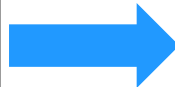
# Transform content using XSL and AI

```

<xsl:template match="/">
  <html>
    <head><title>Employees</title></head>
    <xsl:element name="table">
      <xsl:attribute name="border">1</xsl:attribute>
      <tr class="header">
        <xsl:attribute name="bgcolor">#336666</xsl:attribute>
        <xsl:attribute name="align">center</xsl:attribute>
        <td>
          <font face="Arial" size="3">
            <b>Name</b>
          </font>
        </td>
      </tr>
    </xsl:element>
  </html>
</xsl:template>

...

<xsl:template match="//p:person">
  <xsl:element name="tr">
    <xsl:attribute name="align">center</xsl:attribute>
    <xsl:element name="td">
      <xsl:attribute name="width">120</xsl:attribute>
      <font face="verdana" size="3">
        <i>
          <xsl:value-of select="p:name/p:family/text()"/>
          <xsl:text> </xsl:text>
          <xsl:value-of select="p:name/p:given/text()"/>
        </i>
      </font>
    </xsl:element>
  </tr>
</xsl:template>
  
```



```

<!--This template styles and displays an "Employees" table with
columns for "Name", "Email", and "Link".-->
<xsl:template match="/">
  ...

<!--This XSLT template transforms `p:person` elements into
centered HTML table rows (`tr`) with columns (`td`) for the
person's full name, email, and link attributes for subordinates
and manager, formatted with Verdana font.-->
<xsl:template match="//p:person">
  ...
  
```

# Transform content using XSL and AI

- `ai:transform-content(instruction, (user, agent,)* content)`

```
<!-- Step 1: rephrase in 3 possible ways -->
<sch:let name="user-3variants" value="Provide 3 variants of rephrasing the given text in strictly less than 75 words."/>
<sch:let name="assistant-3Variants" value="ai:transform-content($user-3variants, $currentShortDesc)"/>

<!-- Step 2: get the advantages and disadvantages of each variant -->
<sch:let name="user-compare3varinats" value="For each of the given text variants, enumerate 3 strong points and 3 weak points about how it is written."/>
<sch:let name="assistant-AdvAndDisadv" value="ai:transform-content($user-compare3varinats, $assistant-3Variants)"/>

<!-- Step 3: ask the AI to choose the best variant -->
<sch:let name="system-bestVariant"
  value="Based on the strong points and weak point of the way each text is written, choose the one of them that best summarizes the following text and is the best written. Respond with just the best text, nothing else."/>
<sch:let name="assistantBestVariant" value="ai:transform-content($system-bestVariant,
  $user-3variants, $assistant-3Variants, $user-compare3varinats, $assistant-AdvAndDisadv,
  $currentShortDesc)"/>
```

# AI in XSL and XQuery

- Advantages of using AI with XSL and XQuery
  - Perform refactoring actions
  - Use AI to generate the content
  - Control the content that is sent
  - Control the content that is modified
  - Automate the process
- Challenges
  - Cost for transformation
  - The response from the AI server is not instant
  - Responses can sometimes be inaccurate



# Verify content using Schematron

- Example of a rule that checks if the text uses active voice

In the description we should use active voice

`shortdesc` **Short Description:** The journey into the world of AI is continued through the exploration of its application in conjunction with Schematron and Schematron Quick Fix (SQF) for content verification and correction. In this webinar, a comprehensive overview of AI will be offered, the potential advantages it brings will be highlighted, and the challenges encountered when utilizing AI for these purposes will be illuminated. `shortdesc`



# Check the text voice

- Rule that verifies if the text voice is active

```
<sch:rule context="shortdesc">  
  <sch:assert test="ai:verify-content('Is active voice used?', .)">  
    In the description we should use active voice.</sch:assert>  
</sch:rule>
```

# Correct the text voice

- Example fix that reformulates the text to use active voice

Reformulate the text to use active voice

**Short Description:** The journey into the world of AI is continued through the exploration of its application in conjunction with Schematron and Schematron Quick Fix (SQF) for content verification and correction. In this webinar, a comprehensive overview of AI will be offered, the potential advantages it brings will be highlighted, and the challenges encountered when utilizing AI for these purposes will be illuminated.



**Short Description:** Explore the application of AI in conjunction with Schematron and SQF for content verification and correction in the webinar. Offer a comprehensive overview of AI, highlight its potential advantages, and illuminate the challenges encountered when utilizing AI for these purposes.

# Correct the text voice

- SQF fix that reformulates the text to use active voice

```
<sqf:fix id="rephrase">
  <sqf:description>
    <sqf:title>Reformulate the text to use active voice</sqf:title>
  </sqf:description>
  <sqf:replace match="text()" select="ai:transform-content('
    Reformulate to use active voice', .)"/>
</sqf:fix>
```

# Check if text should be a list

- Example of a rule that checks if the text should be converted to a list

The text should be converted to a list.

<body>

<p> - Is active/passive voice used in the description?

- Is this written according to the styleguide?

- Does the text answer to this "question"?

- Is spelling and grammar correct? </p>

</body>



# Create a list from phrases

- Example of fix that generates an unordered list from a set of phrases

Create a list from the phrases from the paragraph

<p> - Is active/passive voice used in the description?  
- Is this written according to the styleguide?  
- Does the topic answers to this "question"?  
- Is spelling and grammar correct?</p>



<ul>  
 <li>Active/passive voice used?</li>  
 <li>Written according to styleguide?</li>  
 <li>Topic answers question?</li>  
 <li>Spelling and grammar correct?</li>  
</ul>

# Create a list from phrases

- SQF fix that creates a list from a set of phrases

```
<sqf:fix id="replace">
  <sqf:description>
    <sqf:title>Create a list from the phrases from the paragraph</sqf:title>
  </sqf:description>
  <sqf:replace match="text()">
    <xsl:value-of select="ai:transform-content(
      'Create a Dita unordered list with an item from each phrase', .)"
      disable-output-escaping="yes"/>
  </sqf:replace>
</sqf:fix>
```

# AI and Schematron+SQF

- Advantages of using AI with Schematron and SQF
  - Verify and correct your documents using AI
  - Define the instructions to be sent to the AI engine
  - Control the content that is sent
  - Control the content that is modified
  - Automate the content verification and correction
- Challenges
  - High cost for validation as you type or multiple validations
  - The response from the AI server is not instant
  - Responses can sometimes be inaccurate





# Conclusion

- AI constantly growing and improving
- Use AI to generate and mark XML content
- Use AI for Development (in XSL, XSD, Schematron)
- Use XPath functions to automate the process
- Do not expect to do your job, use AI to improve your job



## Resources

- <https://github.com/f/awesome-chatgpt-prompts>
- <https://platform.openai.com/docs/guides/chat>
- Integrated Extension Functions
- <http://schematron.com/>
- <http://schematron-quickfix.github.io/sqf>





Questions?

Octavian Nadolu  
Project Manager at Syncro Soft

[octavian.nadolu@oxygenxml.com](mailto:octavian.nadolu@oxygenxml.com)

Twitter: [@OctavianNadolu](https://twitter.com/OctavianNadolu)

LinkedIn: [octaviannadolu](https://www.linkedin.com/in/octaviannadolu)