

AI and Schematron for Content Verification and Correction



Octavian Nadolu, Syncro Soft

octavian_nadolu@oxygenxml.com

@OctavianNadolu

© 2023 Syncro Soft SRL. All rights reserved

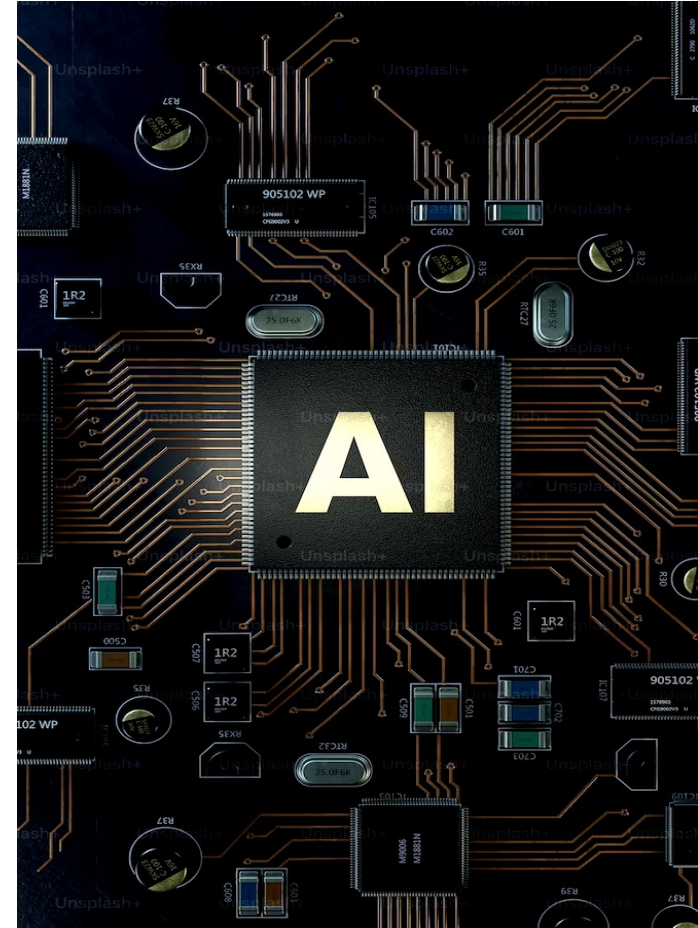
Agenda

- Artificial Intelligence (AI)
- OpenAI/Generative Pre-trained Transformer
- Schematron and Schematron Quick Fixes (SQF)
- Implementing AI in Schematron and SQF
- Examples of AI-driven Schematron and SQF Solutions
- Benefits of Using AI in Schematron and SQF
- AI in Development



What is Artificial Intelligence?

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



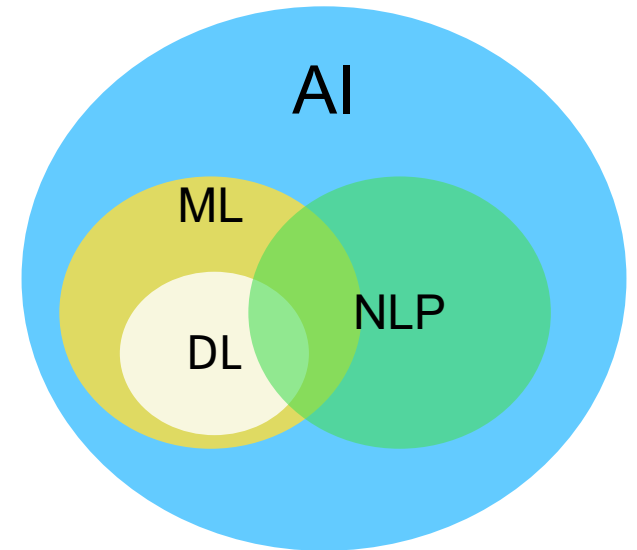
Artificial Intelligence History

- 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



Generative Pre-trained Transformer(GPT)

- Network models that uses the transformer architecture
- A type of LLM (Large Language Model)
- Pre-trained refers to the model being trained on a large corpus of data
- An application is ChatGPT developed by OpenAI

Transformers

- A transformer is a deep learning model architecture used for processing data
- The transformer architecture is based on the idea of self-attention
- Introduced in a research paper titled "Attention Is All You Need" in 2017

Embeddings

- A mathematical representations of words, sentences, or documents in a continuous vector space
- Encode similar words with similar embeddings
- Embeddings have become a fundamental component of many NLP tasks

Question: Do you use or intend to use AI as an assistant for content generation?

- ❑ Yes. I use AI
- ❑ Yes. I intend to use AI
- ❑ No. I do not use AI



OpenAI

OpenAI is an open-source research organization that works to advance artificial intelligence (AI)



OpenAI Application

OpenAI has trained language models that are very good at understanding and generating text

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

What is Schematron?

A language for making assertions in documents



Schematron and AI

Automatically verify documents using an AI model

Examples of verification with AI:

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



Schematron Quick Fix (SQF)

- SQF is an extension of the ISO Schematron
- User-defined fixes for Schematron assert/report

SQF



SQF and AI

- Correct problems in documents using AI
- Examples:
 - Rephrase to use active voice
 - Rephrase to have 20 words
 - Rephrase paragraph to answer to the following question
 - Correct spelling and grammar



Implementation of AI in Schematron

- Using XSLT functions

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

- Using extension functions

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

Oxygen AI Positron Assistant Add-on

- Uses the Oxygen AI Positron service, built on top of OpenAI GPT
- Provides built-in extension functions and actions
- The service is free to use for up to 250 requests per user/month
- Compatible with Oxygen XML Editor / Author / Developer v25.0+



https://blog.oxygenxml.com/topics/ai_positron.html

Question: Do you use OpenAI/ChatGPT or other AI services?

- Yes. I use OpenAI/ChatGPT
- Yes. I use other AI services
- No. I do not use AI



Examples of Schematron Rules



Check if the text is consistent

- Example of a rule that checks if the text is easy to read and understand

The text is not easy to read and understand

¶ The Oxygen XML Editor traverses trans-operating system boundaries, encompassing Windows as well as macOS. The installation process proffers a myriad of divergent methodologies and alternatives, thereby facilitating the execution of Oxygen XML Editor on your system or server. Furthermore, Linux also serves as a compatible platform for the utilization of this software. ¶



Check if the text is consistent



- Rule that verifies if the text is easy to read and understand

```
<sch:rule context="p">  
  <sch:assert test="ai:verify-content('Is the text easy to read and understand?', .)">  
    The text in not easy to read and understand</sch:assert>  
</sch:rule>
```




Correct text consistency

- Example fix that makes the text easy to read and understand

Correct the consistency of the text

 The Oxygen XML Editor traverses trans-operating system boundaries, encompassing Windows as well as macOS. The installation process proffers a myriad of divergent methodologies and alternatives, thereby facilitating the execution of Oxygen XML Editor on your system or server. Furthermore, Linux also serves as a compatible platform for the utilization of this software. 



 Oxygen XML Editor can be used on various operating systems, including Windows, macOS, and Linux. The installation process provides different methods and options to make it easy to install on your system or server. 

Correct the text consistency

- SQF fix that corrects the text to be easy to read and understand

```
<sqf:fix id="rephrase">
  <sqf:description>
    <sqf:title>Correct the consistency of the text</sqf:title>
  </sqf:description>
  <sqf:replace match="text()" select="ai:transform-content(
    'Correct the text to be easy to read and understand', .)"/>
</sqf:fix>
```

Check the text voice

- 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>
```

Answer to question

- Example of a rule that checks if the text answers a specified question

The test does not answer to the question "What is OpenAI?"

OpenAI has developed algorithms that use reinforcement learning and deep learning to solve problems in robotics, natural language processing, and computer vision.



Check the question

- Rule that verifies if the text does answer a specific question

```
<sch:rule context="p[@id='openai']">  
  <sch:assert test="ai:verify-content('Does it answers to the question: What is OpenAI?', .)"/>  
  The test does not answer to the question "What is OpenAPI?" </sch:assert>  
</sch:rule>
```

Reformulate text

- Example fix that reformulates the text to answer a specific question

Reformulate the text to answer to the question:
What is OpenAI?

OpenAI has developed algorithms that use reinforcement learning and deep learning to solve problems in robotics, natural language processing, and computer vision.



OpenAI is a company that creates algorithms using reinforcement learning and deep learning to solve problems in robotics, natural language processing, and computer vision.

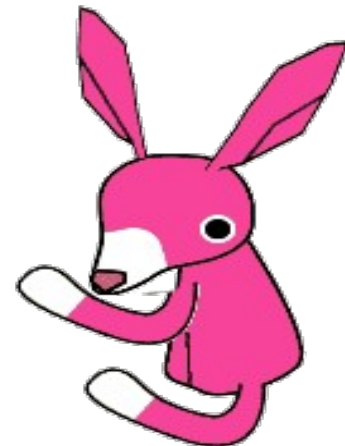
Reformulate text

- SQF fix that reformulates the text to answer the question “What is OpenAI?”

```
<sqf:fix id="rephrase">
  <sqf:description>
    <sqf:title>Reformulate the text to answer to the question:
      What is OpenAI?</sqf:title>
  </sqf:description>
  <sqf:replace match="text()" select="ai:transform-content(
    'Reformulate the text to answer to the question: What is OpenAI?', .)"/>
</sqf:fix>
```

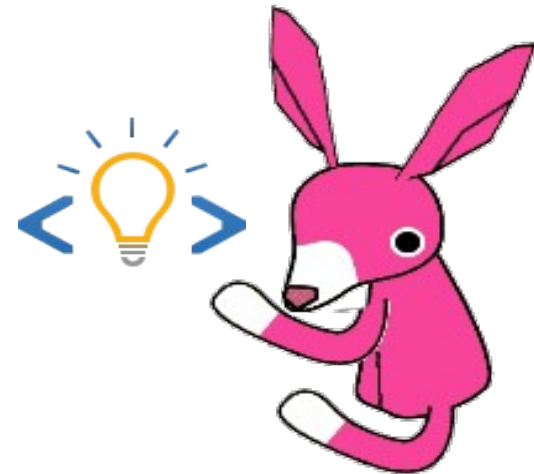
AI and Schematron

- Advantages of using AI with Schematron
 - Verify your documents using AI power
 - Define the instructions to be sent to the AI engine
 - Control the content to be verified
 - Control the content that is sent
 - Automation of the process
- 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



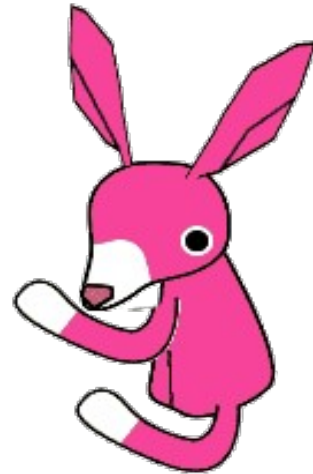
AI and Schematron Quick Fixes

- Advantages of using AI with SQF
 - Use the power of AI to correct the content
 - Control the content that is sent
 - Control the content to be modified
 - Automation of the process
- Challenges
 - The response from the AI server is not instant
 - Responses can sometimes be inaccurate



AI and Schematron Quick Fixes

- Use AI just to correct the problems
 - Reduce the cost
 - You can use the validation as you type
 - The user decides when to call the AI



Check the number of words

- Example of a rule that checks the number of words from the description

The description must contain less than 50 words.

`shortdesc` **Short Description:** Oxygen XML Developer is the industry-leading XML development tool that includes everything you need for designing XML schemas and transformation pipelines. It focuses on XML source editing, schema design, and the XSLT and XQuery support is enhanced with powerful debuggers and performance profilers. Oxygen XML Developer provides a simple and intelligent interface that makes XML development easy and effective. `shortdesc`



Check the number of words

- Rule that verifies the number of words from the shortdesc element

```
<sch:rule context="shortdesc">  
  <sch:report test="count(tokenize(.,'\s+')) > 50">  
    The description must contain less than 50 words.</sch:report>  
</sch:rule>
```


Correct text to contain less words

- Example fix that reformulates the phrase to contains less than 50 words

Reformulate phrase to contain less than 50 words

`shortdesc` **Short Description:** Oxygen XML Developer is the industry-leading XML development tool that includes everything you need for designing XML schemas and transformation pipelines. It focuses on XML source editing, schema design, and the XSLT and XQuery support is enhanced with powerful debuggers and performance profilers. Oxygen XML Developer provides a simple and intelligent interface that makes XML development easy and effective. `shortdesc`



`shortdesc` **Short Description:** Oxygen XML Developer is a comprehensive tool for XML development. It has features for designing XML schemas and transformation pipelines, and provides enhanced support for XSLT and XQuery with debuggers and performance profilers. Its user-friendly interface makes XML development simple and efficient. `shortdesc`

Correct text to contain less words

- SQF fix that reformulates the phrase to have less than 50 words

```
<sqf:fix id="rephrase">
  <sqf:description>
    <sqf:title>Reformulate phrase to contain less than 50 words</sqf:title>
  </sqf:description>
  <sqf:replace match="text()" select="ai:transform-content(
    'Reformulate phrase to contain less than 50 words', .)"/>
</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>



Check if text should be a list

- Rule that verifies the text from a paragraph should be converted to a list

```
<sch:rule context="p">  
  <sch:report test="contains(., '- ')">  
    The text should be converted to a list</sch:report>  
</sch:rule>
```

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>



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

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>
```

User Entry

The instruction to correct the problem is specified by the user



Check technical terms

- Example of a rule that checks if the technical terms are not explained adequately

The text uses WIFI term that is not explained adequately

p The data packets are sent through the router using the WIFI. p



Correct terms

- Example fix that allows the user to specify how to reformulate the phrase

Specify how to reformulate the phrase

p The data packets are sent through the router using the WIFI. p



How to correct:

Reformulate phrase and replace the ambiguous terms with a more accurate one

p The network packets are transmitted through the router using the wireless network. p

Correct terms

- SQF fix that allows the user to specify the prompt that will be sent to the AI

```
<sqf:fix id="reformulateUser">
  <sqf:description>
    <sqf:title>Specify how to reformulate the phrase</sqf:title>
  </sqf:description>
  <sqf:user-entry name="userInput" default=""
    Reformulate phrase and replace the ambiguous terms with a more accurate one">
    <sqf:description><sqf:title>How to correct:</sqf:title>sqf:description>
  </sqf:user-entry>
  <sqf:replace match="text()" select="ai:transform-content($userInput, .)"/>
</sqf:fix>
```

AI Extension Functions

- The extension functions `ai:transform-content(instruction, content)` and `ai:verify-content(instruction, content)` can be used for:
 - Validation with Schematron
 - Quick-fix execution
 - XML refactor action
 - Custom framework action

Generate AI Fix

- Generate the fix from the Schematron message
- Uses the Schematron rule context as the content to correct

```
<sch:rule context="shortdesc">  
  <sch:report test="count(tokenize(.,'\s+')) > 50">  
    The description must contain less than 50 words.</sch:report>  
</sch:rule>
```

Generate AI Fix

- **System prompt:** Act as a developer. Perform the following task:
The description must contain less than 50 words
- **User prompt:** The content from the “shortdesc” element
- **Operation:** Replace the text from “shortdesc” with the one from AI

Automatic Fix

- Advantages:
 - No need to create fixes in Schematron
 - A fix is generated automatically from each Schematron message
 - Fixes can also be generated for other error messages
- Challenges
 - The message need to be as concise as possible so that the AI can use
 - Sometimes you need to change the context for the fix or perform different operations

Use AI for Development

- Use AI in the development process, for example to create Schematron, XSLT, XSD, or JSON Schema documents
- Use AI Positron Assistant “[Follow Instruction](#)” action to generate content
- Generate documentation for Schematron, XSD, XSLT



Create Schematron using AI

- An assert that verifies the number of words to be 10

AI

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

- An assert that verifies if there is an email in text

AI

```
<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>
```

Generate Schematron documentation

- An assert that verifies if documentation is present

```
<sch:rule context="sch:rule | sch:assert">
  <sch:assert test="preceding-sibling::node()[not(self::text())[1][self::comment()]">
    The <sch:name/> does not have a documentation </sch:assert>
  </sch:rule>
```

- A fix that generates the documentation for the Schematron element

```
<sqf:add position="before">
  <xsl:comment select="ai:transform-content(
    'Act as a Schematron developer. Describe the following Schematron code.',
    fn:serialize(.))"/>
</sqf:add>
```


Create XSLT using AI

- A function that gets a string and length and returns a substring, breaking at word boundary

AI

A blue arrow pointing downwards from the text "AI" to the XSLT code block below.

```
<xsl:function name="my:substring-at-word-boundary">
  <xsl:param name="string" as="xs:string"/>
  <xsl:param name="length" as="xs:integer"/>
  <xsl:variable name="substring" select="substring($string, 1, $length)"/>
  <xsl:choose>
    <xsl:when test="contains(substring($string, $length + 1), ' ')">
      <xsl:value-of select="$substring"/>
      <xsl:text> </xsl:text>
      <xsl:sequence
        select="my:substring-at-word-boundary(substring-after($string, ' '), $length)"/>
    </xsl:when>
    <xsl:otherwise>
      <xsl:value-of select="$substring"/>
    </xsl:otherwise>
  </xsl:choose>
</xsl:function>
```

Generate XSLT documentation

- An assert that verifies if documentation is present

```
<sch:rule context="xsl:template | xsl:function">
  <sch:assert test="preceding-sibling::node()[not(self::text())[1][self::comment()]">
    The <sch:name/> does not have a documentation</sch:assert>
</sch:rule>
```

- A fix that generates the documentation for the XSLT element

```
<sqf:add position="before">
  <xsl:comment select="ai:transform-content(
    'Act as an XSLT developer. Describe the following XSLT code.',
    fn:serialize(.))"/>
</sqf:add>
```

Create XSD using AI

- A complex type that defines a tournament



```
<xsd:complexType name="Tournament">  
  <xsd:sequence>  
    <xsd:element name="Name" type="xsd:string"/>  
    <xsd:element name="Location" type="xsd:string"/>  
    <xsd:element name="StartDate" type="xsd:date"/>  
    <xsd:element name="EndDate" type="xsd:date"/>  
    <xsd:element name="PrizeMoney" type="xsd:decimal"/>  
    <xsd:element name="Winner" type="xsd:string"/>  
  </xsd:sequence>  
</xsd:complexType>
```

Generate XSD documentation

- An assert that verifies if documentation is present

```
<sch:rule context="xsd:element | xsd:complexType">
  <sch:assert test="preceding-sibling::node()[not(self::text())[1][self::comment()]">
    The <sch:name/> does not have a documentation</sch:assert>
</sch:rule>
```

- A fix that generates the documentation for the XSD element

```
<sqf:add position="before">
  <xsl:comment select="ai:transform-content(
    'Act as an XSD developer. Describe the following XSD code.',
    fn:serialize(.))"/>
</sqf:add>
```

Create JSON Schema using AI

- A type that defines a purchase order

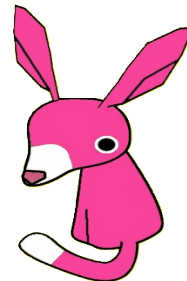
A blue downward-pointing arrow with the letters "AI" in black text inside it, indicating the process of using AI to generate the schema.

AI

```
{
  "type": "object",
  "properties": {
    "orderNumber": {"type": "integer"},
    "customerName": {"type": "string"},
    "items": {
      "type": "array",
      "items": {
        "type": "object",
        "properties": {
          "productId": {"type": "integer"},
          "quantity": {"type": "integer"},
          "price": {"type": "number"}
        }
      }
    }
  }
}
...
```

Conclusion

- Schematron can use the power of AI to verify content
- Correct content using SQF and AI
- Generate fixes automatically using AI
- Develop Schematron, XSLT, XSD, or JSON Schema with AI
- AI constantly improving and growing



Future Plans

- Add support in Web Author
- Implement new extension functions
- Add cache for GPT requests
- Improve AI support for development



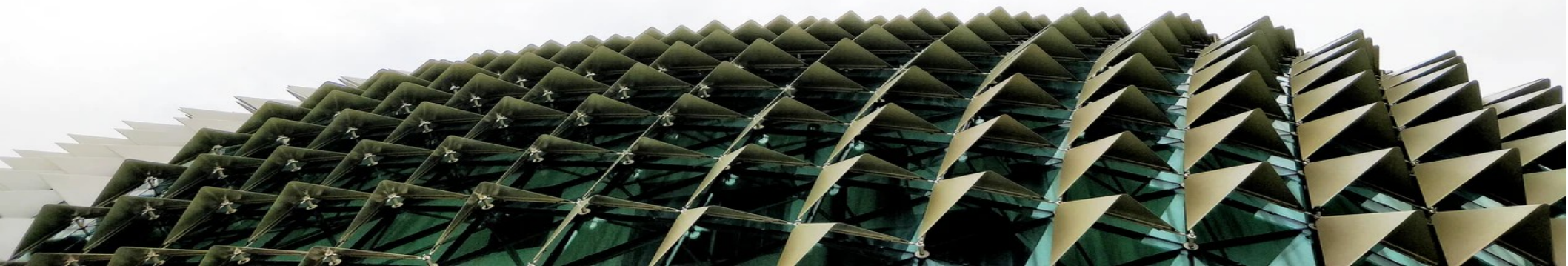
Question: What features are the most important for you?

- ❑ Add support in Web Author
- ❑ Implement new extension functions
- ❑ Add cache for GPT requests
- ❑ Improve AI support for development
- ❑ Other (feedback is welcome)



Resources

- <https://openai.com/>
- <https://platform.openai.com/docs/guides/chat>
- <http://schematron.com/>
- <http://schematron-quickfix.github.io/sqf>





Questions?

Octavian Nadolu
Project Manager at Syncro Soft

octavian.nadolu@oxygenxml.com

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

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