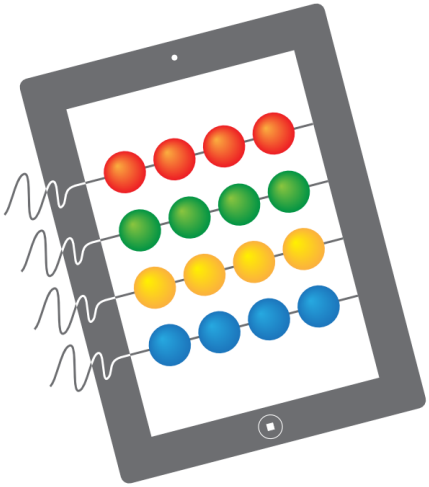
FP7 ICT STREP Project



LEARN PAD

Deliverable D4.3

Quality Assessment Mechanisms Implementation

Descriptions of the Modules for Models Verification and NL Content Quality Assessment

http://www.learnpad.eu

LINAGORA

No Magic Europe

Fachhochschule Nordwestschweiz

WIKI

Project Number	:	FP7-619583
Project Title	:	Learn PAd Model-Based Social Learning for Public Administrations

Deliverable Number	:	D4.3
Title of Deliverable	:	Quality Assessment Mechanisms Implementation
Nature of Deliverable	:	Prototype
Dissemination level	:	Public
Licence	:	Creative Commons Attribution 3.0 License
Version	:	4.0
Contractual Delivery Date	:	April 30, 2016
Actual Delivery Date	:	April 30, 2016
Contributing WP	:	WP4
Editor(s)	:	Barbara Re (UniCam), Fabrizio Fornari (UniCam), Giorgio Oronzo Spagnolo (CNR), Alessio Ferrari (CNR)
Author(s)	:	Barbara Re (UniCam), Andrea Polini (UniCam), Stefania Gnesi (CNR), Alessio Ferrari (CNR), Fabrizio Fornari(UniCam), Giorgio O. Spagnolo (CNR), Flavio Corradini (UniCam)
Reviewer(s)	:	Guglielmo De Angelis (CNR), Jean Simard (xWIKI)

Abstract

This deliverable describes the implementation of the quality assessment strategy in Learn PAd. In particular, it describes the implementation of the Model Verification component (MV), and Content Analysis component (CA). The former implements the strategy for formal verification, and understandability checking of BP models. The latter implements the strategy for quality assessment of the natural language content that describes the BP models.

Keyword List

BPMN, Business Process, Understandability, Verification, Quality, Learnability, Natural Language Processing, Natural Language Quality

Document History

Version	Changes	Author(s)
0.1	Table of Content.	Barbara Re (UniCam), Andrea Polini (UniCam), Stefania Gnesi (CNR), Alessio Ferrari (CNR), Fabrizio Fornari(UniCam), Giorgio O. Spagnolo (CNR), Flavio Corradini (UniCam)
1.0	First Draft.	Barbara Re (UniCam), Andrea Polini (UniCam), Stefania Gnesi (CNR), Alessio Ferrari (CNR), Fabrizio Fornari(UniCam), Giorgio O. Spagnolo (CNR), Flavio Corradini (UniCam)
2.0	Complete draft	Barbara Re (UniCam), Andrea Polini (UniCam), Stefania Gnesi (CNR), Alessio Ferrari (CNR), Fabrizio Fornari(UniCam), Giorgio O. Spagnolo (CNR), Flavio Corradini (UniCam)
3.0	Candidate Release	Barbara Re (UniCam), Andrea Polini (UniCam), Stefania Gnesi (CNR), Alessio Ferrari (CNR), Fabrizio Fornari(UniCam), Giorgio O. Spagnolo (CNR), Flavio Corradini (UniCam)
4.0	Final Quality Check	Antonia Bertolino (CNR)

Document Reviews

Release	Date	Ver.	Reviewers	Comments
ToC	29 Feb 2016	0.1		
Draft	25 Mar 2016	1.0		
Internal	09 Apr 2016	2.0		
Candidate Final	23 Apr 2016	3.0		

Glossary, acronyms & abbreviations

Item	Description
7PMG	Seven Process Modeling Guidelines
AGD	Average Gateway Degree
BP	Business Process
BPEL	Business Process Execution Language
BPMN	Business Process Model and Notation
CFC	Control Flow Complexity
GM	Gateway Mismatch
GH	Gateway Heterogeneity
CRG	Compliance Rule Graph
eCRG	extended Compliance Rule Graph
CTL	Computation Tree Logic
G-CTL	Graphical Computational Tree Logic
EG-CTL	Extended Graphical Computational Tree Logic
EPC	Event Process Chain
FCL	Formal Contract Language
FSP	Finite State Processes
LTL	Linear Temporal Logic
MGD	Maximum Gateway Degree
NL	Natural Language
OMG	Object Management Group
PA	Public Administration
PAIS	Process-Aware Information System
PN	Petri Net
TNG	Total Number of Gateways

Table Of Contents

List Of Figures

1 Introduction

1.1. Deliverable Purpose

This report gives a description of the Model Verification component, and of the Content Analysis component. It is provided as a complement to the software deliverable D4.3. For each component, we provide an overview of the architecture and behaviour, and we give the guidelines on how to download and run each component.

1.2. Related Deliverables

The deliverable has been organized according to the first results of the Learn PAd project as stated by the following Deliverables.

- **D4.1 Quality Assessment Strategies for BP Models**.
- **D4.2. Quality Assessment Strategies for Contents**.

1.3. Deliverable Structure

The deliverable is organized as follows.

- Chapter 2 describes the implementation of the model verification component.
- Chapter 3 describes the implementation of the content analysis component.
- Chapter 4 reports some conclusions and future development.

2 Model Verification Component

2.1. Overview

In this chapter we provide a detailed description of the Verification component architecture. This component is engaged to manage all the aspects of the Learn PAd model formal analysis. In particular, in this chapter we will describe all its sub-components, all the executions flows supported and the interfaces provided.

2.2. Architecture

As we already present in the Deliverable 4.2 in Fig. **??** a general overview of the Verification component architecture is provided. As can be seen by this overview, currently the architecture is mainly composed by 4 sub-components: the BPMN to Petri Net generator, the optimization engine, the model checker component and the metrics calculator also called understandability component.

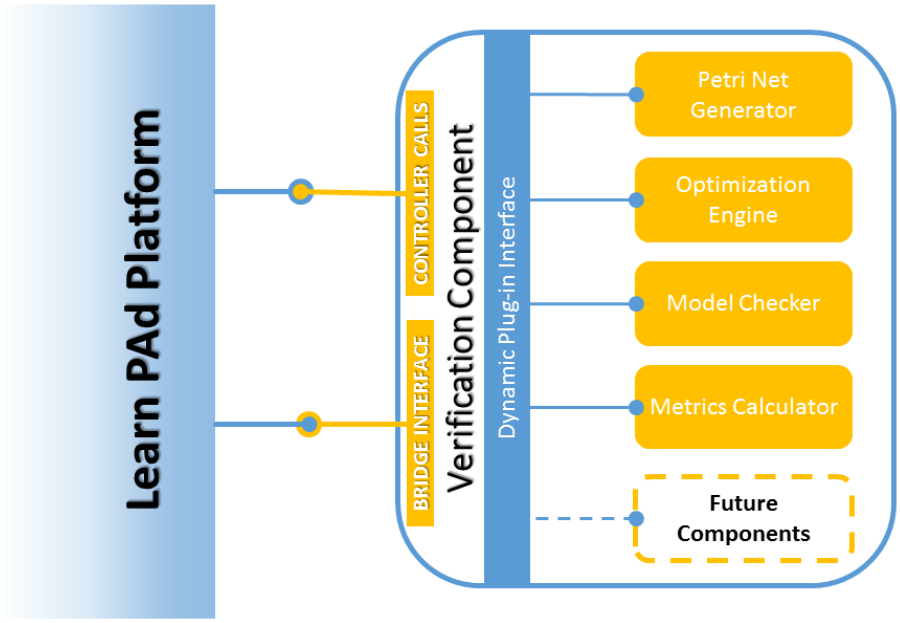


Figure 2.1: Overall architecture

With this initial structure of the component we are able to formally verify structural problems, present in the flow of the modeled Business Process and to evaluate its understandability. In order to verify other aspects described in the Learn PAd model, this architecture can be extended adding other sub-components used to generate the relative Petri Net for each specific aspect. The optimization engine and the model checker component will be reused. Thanks to the pluggable interface implemented, each component can be substituted with another providing the same functionality and new modules can be added in order to perform other kind of verification checks.

As any Learn PAd module, the Verification component will expose a set of functionalities to the platform

through the bridge interface while will call functions inside its controller in order to contact the platform. Following we will discuss in detail each component.

2.2.1. Business Process to Petri Net Generator

Thanks to this sub-component, it is possible to create a Petri Net that strictly represent the Business Process flow of the Learn PAd model. This module integrates the mapping rules previously described and is the point of access for the Verification component as it will be described in the following sections. For each BPMN element this module will apply the right mapping rule creating the corresponding Petri Net elements. The generated Petri Net elements are then connected as described by the Sequence and Message Flow of the BPMN model. The component uses a specific data structure in order to represent, memorize and interact with a Petri Net. This structure is shared between all the components in order to simplify the operations of generating, optimizing Petri Nets and exporting them in a format accepted by the Model Checker. Once a Petri Nets as been generated from the BPMN model, the optimization engine is called.

2.2.2. Optimization Engine

This module manages all the mechanisms that can be applied to a Petri Net in order to minimize it and optimize the work of the Model Checker component. All the algorithms applied in this phase have to preserve the original semantic of the Petri Net. In particular this module implements two optimization mechanisms:

- Unfolding: it will generate an unfolded net starting from the original one.
- Reduction: it will apply some simple reduction rules in order to simplify the Petri Net preserving its original semantic. In this way it is possible to alleviate state space explosion in the model checking phase. This mechanism will be a next feature so a detailed description of the reduction rules will be provided in a future version of the document.

2.2.3. Model Checker Component

The model checker component is the engine that performs the formal verification of a specific property on the model. It relies on an external tool named LOLA (LOw Level petri net Analyzer) and provides mechanisms in order to interact with the tool in terms of:

- providing the Petri Net model in the specific syntax accepted by the tool.
- providing the property in the specific syntax accepted by the tool.
- processing the tool results for internal analysis.

LOLA is released as open source under GNU License. It is well consolidated and implements state of the art techniques in order to perform property verification in a formal and efficient way, as demonstrated by its score obtained on several Model Checking contests (http://mcc.lip6.fr/).

The model checker component in particular take care of the following phases:

1) Property Generation: in this phase we generate a property strictly specific for the model to verify.
2) Petri Net Model Export: in this phase the internal Petri Net model is exported in the specific format accepted by the model checker tool. LOLA in particular uses its own format named EBNF in order to represent a Petri Net.
3) Property Export: in this phase we export the previously generated property in the syntactical format accepted by the model checker. The properties accepted by LOLA are expressed as CTL* formulas extended with some special words.

4) Verification: the module now calls the model checker tool and waits for a response. This part of the process can be really time consuming, so the component has to manage the process termination after a specific timeout in order to not saturate the machine resources. It also has to perform a multi-thread execution so it can be able to get, each time, the status of any process.

5) Result Analysis: in this phase the output of the model checker tool is captured and processed in order to return true when the property is verified, or false otherwise. This module will also analyze the counter example trace when the property is verified and reports it in a structured way.

2.2.4. Understandability Component

This module is used to calculate understandability metrics for a given Business Process. This sub-component is the only one disjoint from the previously described because don't need translation in Petri Net and model checking but will work directly with the BPMN model analyzing its elements.

The quality assessment strategy includes modeling understandability guidelines which are supported by a Java tool integrated with the Learn PAd Modeling Environment[1]. The Java tool helps the designer, to establish if a model complies with the guidelines. This component is a freely downloadable[2] plugin written in Java. The tool reads a *.bpmn* file compliant with the OMG BPMN 2.0 standard and produces a XML file which describes the guidelines that are not met and the BPMN elements violating them. This component exposes also a *RESTful* interface for being used outside the Learn PAd platform and it is also ready to be packaged as a WAR to be deployed on an Application Server like Tomcat. We also developed, for demonstrability and reusability purposes, a basic web user interface to permit the access to the guidelines verification component[3]. The tool allows to automatically verify 32 of the 50 guidelines; this 32 guidelines are the ones that have an associated threshold or refer to the presence/absence of BPMN elements and their associated labels. Each guideline applies to specific model elements. Therefore, the implemented algorithm navigates the model elements that are relevant and checks whether the elements comply to the guideline. For example, guideline "Explicit start and end events" (ID 12) applies to BPMN event elements. In this case, the algorithm navigates the whole set of events, to check that Start and End Events are included in the model, as described in the pseudocode below (Algorithm 1).

Algorithm 1: Guideline (ID 12) "Explicit start and end events"

```
forall the FlowElement fe ∈ Process P do
    if fe instance of StartEvent then
        countStartEvent++;
    else
        if fe instance of EndEvent then
            countEndEvent++;
        end
    end
end
if countStartEvent < 1 ∨ countEndEvent < 1 then
    return guidelineviolated;
end
```

To provide a slightly more complex example, we consider guideline *"Use linear sequence flow"* (ID 45). The basic idea of this guideline is the creation of sequence flows without foldings, if not really necessary. So, intuitively, if the sequence flows are connected to a gateway, then the sequence flows can include foldings, otherwise not. Algorithm 2 implements the check of the guideline ID 45. To be linear, a sequence flow shall

[1] Learn PAd modeling environment, available at: https://goo.gl/cSgCzU
[2] Guidelines verification component, source code available at: https://goo.gl/hK33Ix
[3] http://understandabilitybpmn.isti.cnr.it

be composed of two points (*maxwaypoints* = 2, in the algorithm). If a sequence flow is composed of more than two points, then the sequence flow shall be connected to a gateway. The connection to a gateway can include a maximum of two foldings, which means four points (*maxwaypoints* = 4).

Algorithm 2: Guideline (ID 45) "Use linear sequence flow"

```
maxwaypoints = 2;
forall the BPMNEdge be ∈ Process P do
    if be instance of SequenceFlow then
        List<waypoint> waypoints = be.getWaypoint();
        target = fe.getTarget();
        source = fe.getSource();
        if target ∨ source instance of Gateway then
            maxwaypoints = 4;
        end
        if waypoints.size() > maxwaypoints then
            return guidelineviolated;
        end
    end
end
```

2.3. Behavior

In this section we provide a description of the interaction flows between the Verification component and the Learn PAd platform and between the modules inside the Verification component.

Fig 2.2 provides a detailed view of the interaction between the Learn PAd Platform and the Verification Component with the interactions between all the sub-modules of the Verification Component involved in the verification of properties over a Petri Net model.

The interaction process is started by the platform that asks for the verification of a model. The type of verification to be performed is chosen among all the available verification types provided by the component. In this case the type must be one related to property verification over Petri Net models like deadlock checks. The request for verification returns a unique Id that has to be used in order to check the status of the verification. After this request the Verification component generates a new verification thread referenced by the returned unique Id. This thread obtains the model from the platform ad starts the verification generating a PN model from the BP. After that the optimization mechanisms are applied generating the unfolded net and reducing the net. When all the optimization phases are completed the model checker module generates the property to verify, it exports the property and the model in the format accepted by the model checker tool and it executes the external tool. When the tool terminates its execution, the module processes the result and returns a response including counter-example to the verification component. Received the response, the Verification component terminates the thread and it stores the result for future requests changing its status to completed.

In any moment after the verification request, the platform can ask for the verification status. In this case the Verification component will look for the specified thread and obtain a response that can be "In progress" if the thread is present or "Completed" if the thread is already terminated. When the platform obtains the status "Completed" it can ask for the result that will be provided in XML format.

Fig 2.3 provides a detailed view of the interaction between the Learn PAd Platform and the Verification Component with the interactions between the sub-module of the Verification Component involved in the verification of process understandability.

The process in this scenario is started by the platform that asks for the verification of a model. The type of verification to be performed can be chosen between all the available verification types provided by the

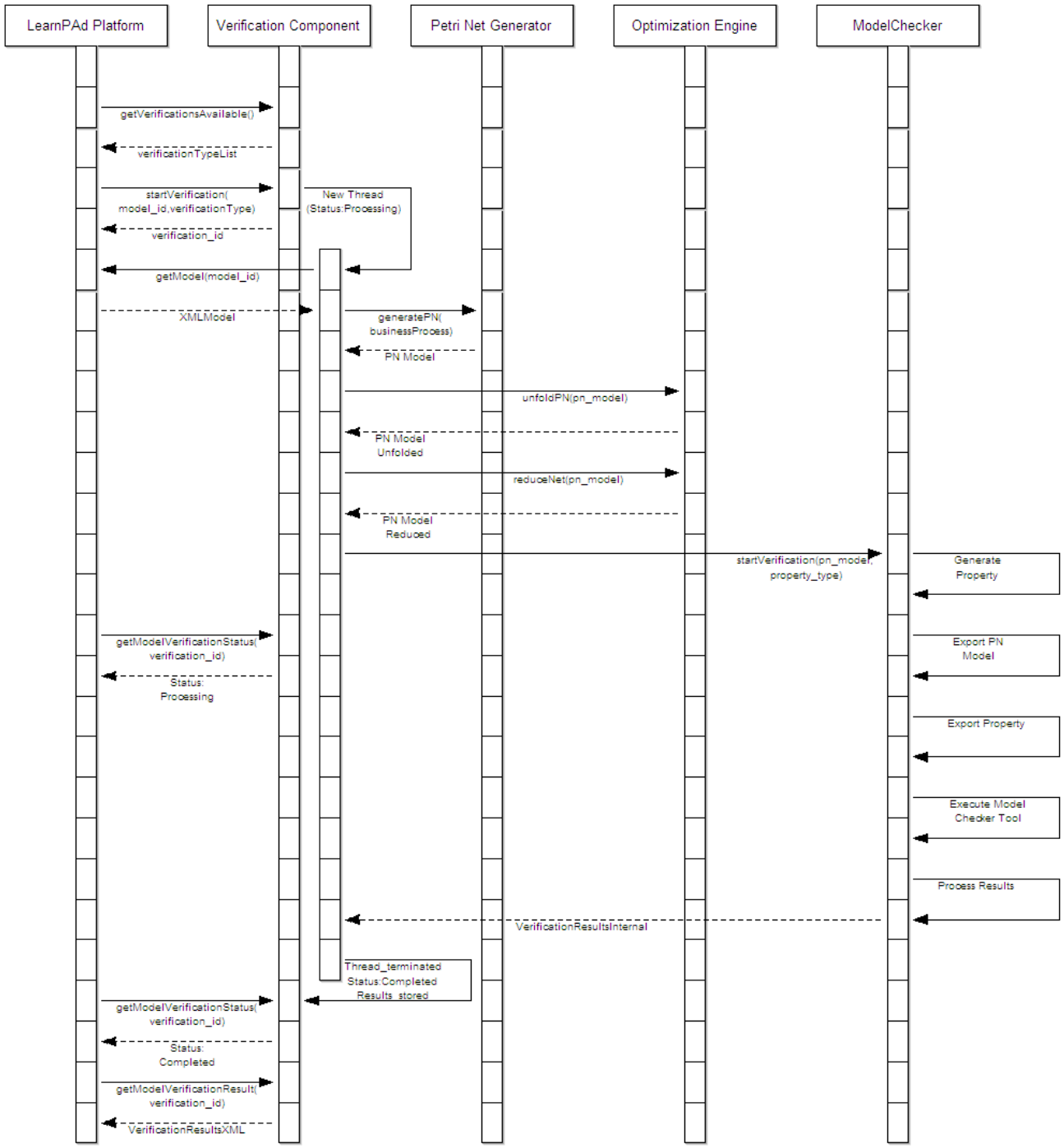


Figure 2.2: Interaction between Learn PAd Platform and Verification Component: formal verification

component. In this case the type must be one that specify the verification of understandability. The request for verification returns a unique Id that has to be used in order to check the status of the verification. After this request the Verification component generates a new verification thread referenced by the returned unique Id. This thread obtains the model from the platform ad starts the analysis calling the Metrics Calculator module. When this module finish its execution, it returns all the generated metrics to the verification component. Received the response, the Verification component terminates the thread and it stores the result for future requests changing its status to completed. In any moment after the verification request, the platform can ask for the verification status. In this case the Verification component will look for the specified Thread and obtain a response that can be "In progress" if the thread is present or "Completed" if the thread is already

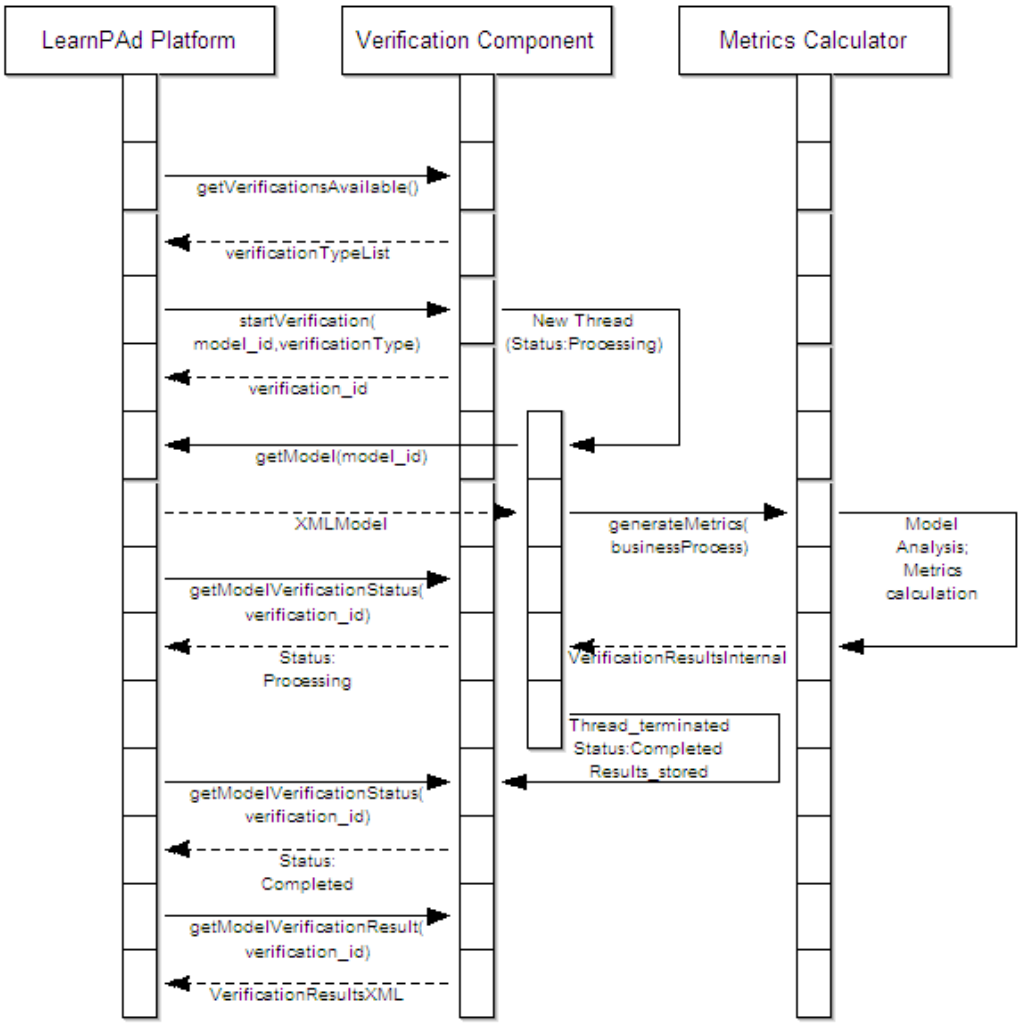


Figure 2.3: Interaction between Learn PAd Platform and Verification Component: understandability

terminated. When the platform obtains the status "Completed" it can ask for the result that will be provided in XML format.

2.4. Build and Run

The following steps must be followed in order to execute the model verification component:

1) download the model verification component using the git clone command:
   - git clone https://github.com/LearnPAd/learnpad.git
2) build and run the App using the provided scripts:
   - cd lp-model-verification
   - ./build
3) Launch the component:
   - cd out & ./start

Inside the jar (in eu/learnpad/verificationComponent/utils/) is present a file called config.txt where you can change the default address of the LearnPAd Platform and the local port used by the REST Web Server.

The service is available at "localhost:9998/rest" and all the provided methods can be used from the wadl file "localhost:9998/rest/application.wadl".

This component exposes REST API in order to perform several kind of verifications on a LearnPAd model. The different types of verifications are provided by plugins. The component automatically recognize all the plugins present inside a specific folder (the default one is ./VerificationComponentPlugins where . is the directory containing the verification component jar) and execute the right one on each request, in an asynchronous way.

Results are stored in a folder (the default one is ./VerificationComponentResults where . is the directory containing the verification component jar) and available for future analysis. All the plugins are automatically compiled and copied in the default plugin directory by maven.

3 Content Analysis Component

3.1. Overview

The Content Analysis (CA) component implements the features for natural language content analysis described in Deliverable D4.2. To this end, it implements a series of linguistic quality evaluation checks on the content that is received from the Learn PAd core platform. In this section, we give an overview of the structure and behaviour of the component, and we explain how to download, build and run the component.

3.2. Architecture

Fig. 3.1 depicts the architecture of LearnPAd. The Content Analysis (CA) component interacts with the Learn PAd core platform to (a) get the content that shall be verified, and (b) produce the results of the verification.

Fig. 3.2 reports the architecture of the CA component, highlighting the main elements and libraries that form the component, namely:

- **CA core:** this is the controller of the component, which performs the call to the other libraries;
- **Language tools:** this is the library that is called to perform correctness checks;
- **GATE:** this is the library that is called to support POS tagging, and other language pre-processing features required for part of the checks performed by CA core.
- **REST Interface:** this is the interface that is used to interact with the Learn PAd core platform.

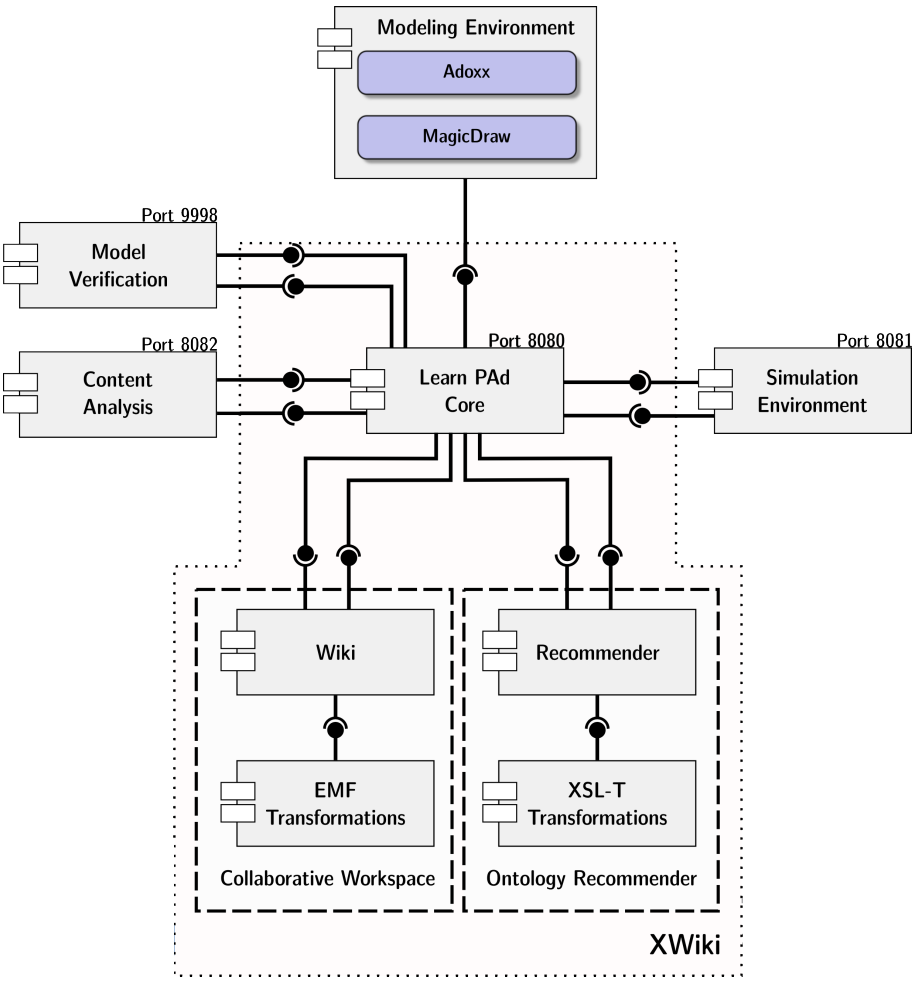


Figure 3.1: LeanPad overall architecture

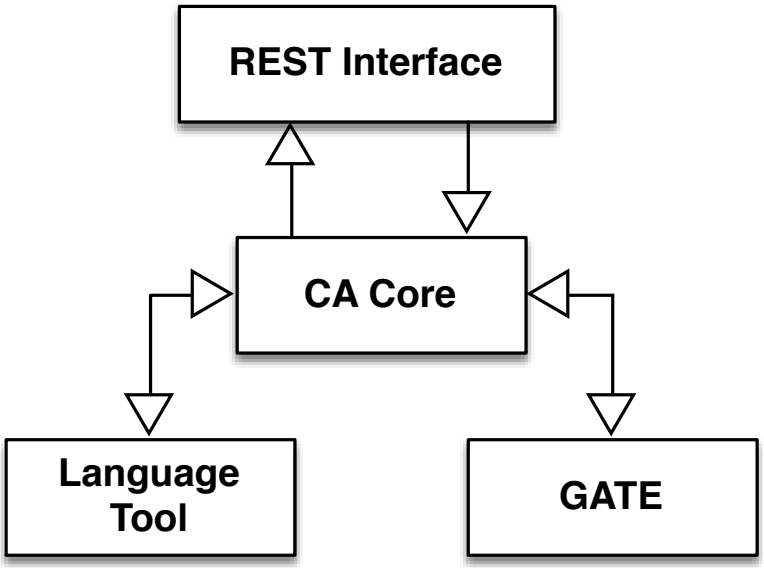


Figure 3.2: CA component architecture

```
<CollaborativeContentAnalysis language="english">
        <CollaborativeContent id="1213">
                <Title>Title of Documents</Title>
                <ContentPlain>
 The document shall be sent to the proper
   authorities as soon as possible
</ContentPlain>
                <ContentHTML>
                <![CDATA[ The document shall be <p>sent</p> to the proper
   authorities as soon as possible ]]>
                </ContentHTML>
        </CollaborativeContent>
        <QualityCriteria simplicity="true" non_ambiguity="true"
                content_clarity="true" presentation_clarity="false"
                completeness="false" correctness="true" />
</CollaborativeContentAnalysis>
```

Figure 3.3: Collaborative Content received by the CA component

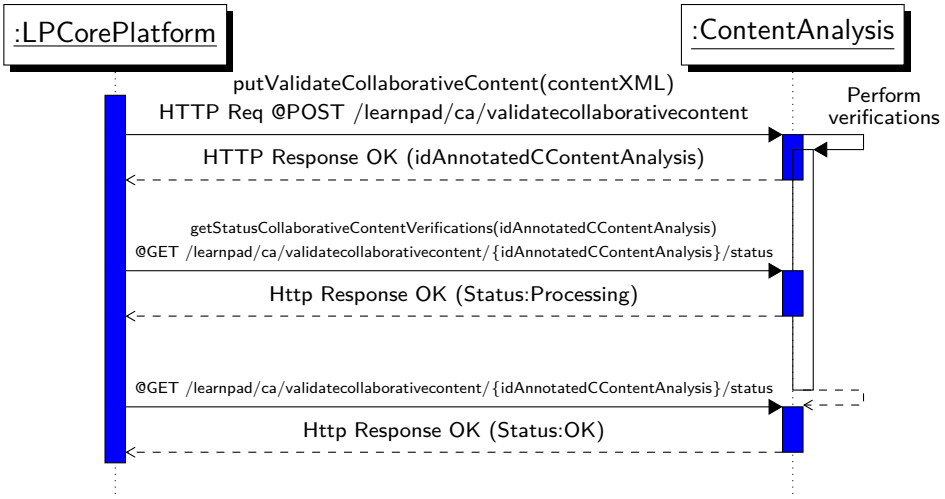


Figure 3.4: The content is sent to the CA component

3.3. Behavior

In this section we briefly describe the behaviour of the CA component. The component receives an XML file that has the form described in Fig.3.3. The XML file includes the NL content to be analysed and the checks that have to be performed. The behaviour at the architectural level is depicted in the sequence diagram in Fig. 3.4.

After the content is analysed, the CA component produces an XML file that has the form depicted in Fig. 3.6. The interaction with the Learn PAd core platform to get the analysis included in the XML file is depicted in Fig. 3.5.

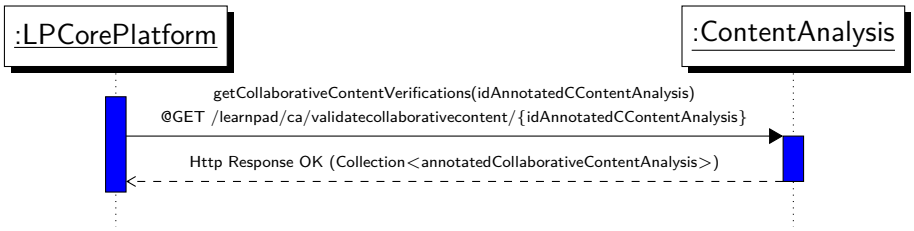


Figure 3.5: The CA component has analysed the content, and produces the results

```
<annotatedCollaborativeContentAnalysis id="1" type="Non Ambiguity">
    <CollaborativeContent id="1213">
        <Title>Title of Documents</Title>
        <Content>
 The document shall be sent to the <Node id="16"/>proper<Node id="17"/>
   authorities as soon <Node id="27"/>as <Node id="29"/>possible<Node id="30"/>
               </Content>
    </CollaborativeContent>
    <Annotations>
        <Annotation id="43" type="Lexical Ambiguity Vagueness" StartNode="16" EndNode="17"
        recommendation="The term proper is vague. Remove proper or substitute it with a
        more unequivocal term." StartNode_Offset="36" EndNode_Offset="42"/>
        <Annotation id="45" type="Lexical Ambiguity Vagueness" StartNode="29" EndNode="30"
        recommendation="The term possible is vague. Remove possible or substitute it with a
         more unequivocal term." StartNode_Offset="69" EndNode_Offset="77"/>
        <Annotation id="44" type="Lexical Ambiguity Subjectivity" StartNode="27" EndNode="30"
        recommendation="The term as possible is subjective. Remove as possible or substitute
         it with a more unequivocal term." StartNode_Offset="66" EndNode_Offset="77"/>
    </Annotations>
    <OverallQuality>BAD</OverallQuality>
    <OverallQualityMeasure>50%</OverallQualityMeasure>
    <OverallRecommendations>Quality is poor, correct the errors</OverallRecommendations>
</annotatedCollaborativeContentAnalysis>
```

Figure 3.6: XML Result analysis of the CA component

3.4. Build and Run

A guide on how to compile and run the application.

To **download** the component:

- Primary download repository: `http://www.learnpad.eu/docs/learnpad-master-M27.zip`
- The component can also be downloaded by the open source community from: `git clone https://github.com/LearnPAd/learnpad.git`

To **build and run** the component:

- `cd lp-content-analysis`
- `./build` to build app
- `cd out & ./start` to start app

To perform **cURL test**:

→ `curl -X POST -H "Content-Type: application/XML" --data @CollaborativeContentXML.xml` `http://localhost:8082/lp-content-analysis/learnpad/ca/validatecollaborativecontent/`

← return String: {idAnnotatedCContentAnalysis}

→ `curl -X GET` `http://localhost:8082/lp-content-analysis /learnpad/ca/validatecollaborativecontent/{id}/status`

← return String: {OK‖Processing‖Error}

→ `curl -X GET` `http://localhost:8082/lp-content-analysis /learnpad/ca/validatecollaborativecontent/{id}`

← return XML

To download the Wadl (i.e., XML description of the REST interface):

- `http://localhost:8082/lp-content-analysis/application.wadl`

A working **demo** of the application – not fully tested, only for demonstrative purposes – can be accessed at:

- `http://contentanalysis.isti.cnr.it:8080/uicontentanalysiscomponent/contentform.jsf`

To **use the demo** (for each step, screen-shots are reported below):

1) Add Title, Identifier, Text to be analysed, Language and select the Quality Criteria in the checkboxes (Fig. 3.7)
2) Press Get Analysis (Fig. 3.8)
3) Select one of the quality criteria to show potential errors (Fig. 3.9)
4) Check the errors, hover the mouse over the errors to see recommendations (Fig. 3.10)

A **video** of the demo – no sound – is available at: `https://youtu.be/dZ5s9X1K__c`

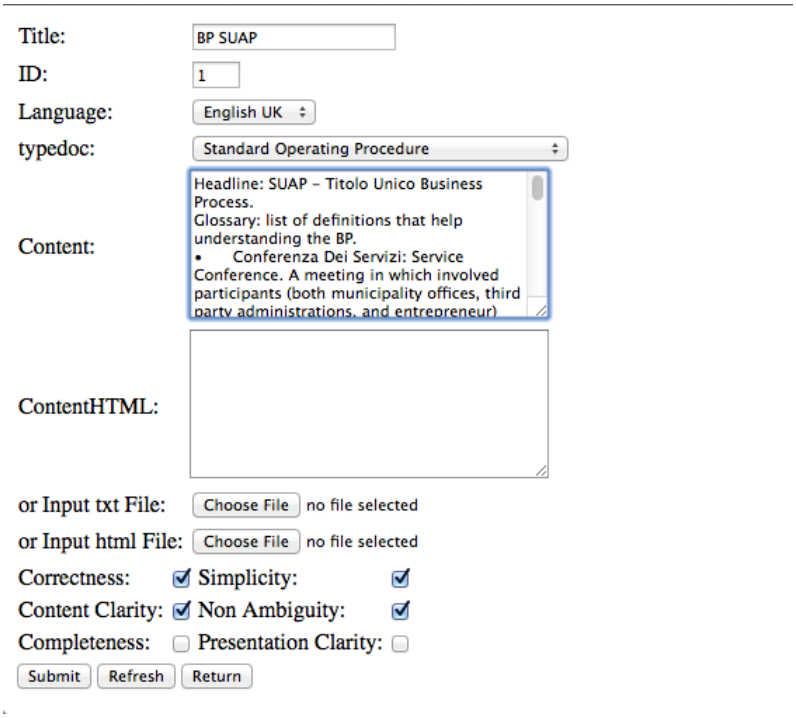


Figure 3.7: Step 1

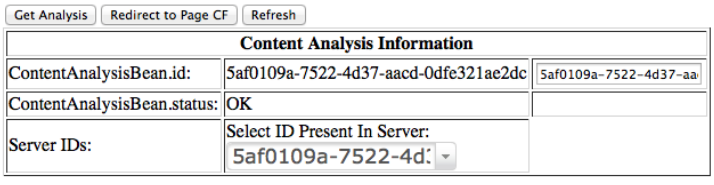


Figure 3.8: Step 2

Content Analysis Information of Correctness Title: BP SUAP	
Overall Quality	VERY GOOD
Overall Quality Measure	88.33%
Overall Recommendations	Well done, still few errors remaining
Correctness	

Content Analysis Information of Simplicity Title: BP SUAP	
Overall Quality	NOT SO BAD
Overall Quality Measure	65.67%
Overall Recommendations	Quality is not so bad, but there are several errors.
Simplicity	

Content Analysis Information of Non Ambiguity Title: BP SUAP	
Overall Quality	GOOD
Overall Quality Measure	71.64%
Overall Recommendations	Quality is acceptable, but there are still some errors
Non Ambiguity	

Content Analysis Information of Content Clarity Title: BP SUAP	
Overall Quality	BAD
Overall Quality Measure	44.78%
Overall Recommendations	Quality is poor, correct the errors
Content Clarity	

Figure 3.9: Step 3

Content Analysis Information of Non Ambiguity Title: BP SUAP	
Overall Quality	GOOD
Overall Quality Measure	71.64%
Overall Recommendations	Quality is acceptable, but there are still some errors

Non Ambiguity:

Headline: SUAP - Titolo Unico Business Process.
Glossary: list of definitions that help understanding the BP.

- Conferenza Dei Servizi: Service Conference. A meeting in which involved participants (both municipality offices, third party administrations, and entrepreneur) discuss about the case and decide if the business activity is acceptable.
- Sportello Unico Per Le Attività Produttive": is the office that permits to entrepreneurs to set up a new company or to organize a business activity.
- Titolo Unico: Standard request to start business activity. Titolo Unico is introduced by the Italian law D.P.R. 160/2010 in the article 7 in which a citizen requests to the municipality and third parties the permission to start an activity. In this case the entrepreneurs have to wait the office decision before starting the activity.

Context: In the PA context, this process defines all the tasks that are required to allow citizens to open a commercial activity.
In any BP referring to the SUAP, the entrepreneurs (possibly an intermediary acting on behalf of the entrepreneur) is the trigger of the process. At the same time municipality or aggregation are the contact point. At this level different offices of the municipality can be involved. This depends on the internal organization of the municipality and to the type of business activity. Several third party administrations and offices participate to the process. Nevertheless, we cannot generalize such third party participants and offices because they differ depending on a several variants. This means that two applications in the same municipality can involve completely different third party administrations considering the content of the application. The SUAP office is not explicitly stated as a participant, but is just a part of an organization. It can be the municipality or a part of the organization or PA that provides the SUAP.
Motivation: SUAP refers to the activities that the Italian Public Administrations have to do in order to permit to entrepreneurs to set up a new company or to organize a business activity. Reducing the administrative burden the entrepreneurs refers to a single office, the SUAP office. Contact between entrepreneurs and SUAP office has to be done completely online. SUAP office is a mediator among entrepreneurs and PAs. This means, that in case an entrepreneur wants to start a manufacturing activity, he/she needs to contact just the SUAP office that is in charge to forward the request to all the other administrations.
Intended readership: Any civil servant who has to perform the SUAP process. Could be a front-end employee, or manager.
Required tools: the following are the tools required to perform the process.
FedCohesion is the regional authentication framework supporting Federated Digital Identity management. It is used as regional standards for authentication to front and back office procedures. For what concern authentication FedCohesion supports national service card, username and password or username password and pin as different levels of security mechanisms to access the services.
Carta Raffello is free delivery to citizens, business, public offices and organizations in the region of national identity card including digital signature and personal authentication credentials.

Figure 3.10: Step 4

4 Conclusion

This deliverable describes the architecture and behaviour of the components developed within WP4 of Learn PAd namely the Model Verification component, and the Content Analysis component. Further development for the Content Analysis component will be the introduction of additional checks for the linguistic quality, and the introduction of a data-base to store previously performed checks, and increase the accuracy in natural language defect prediction.