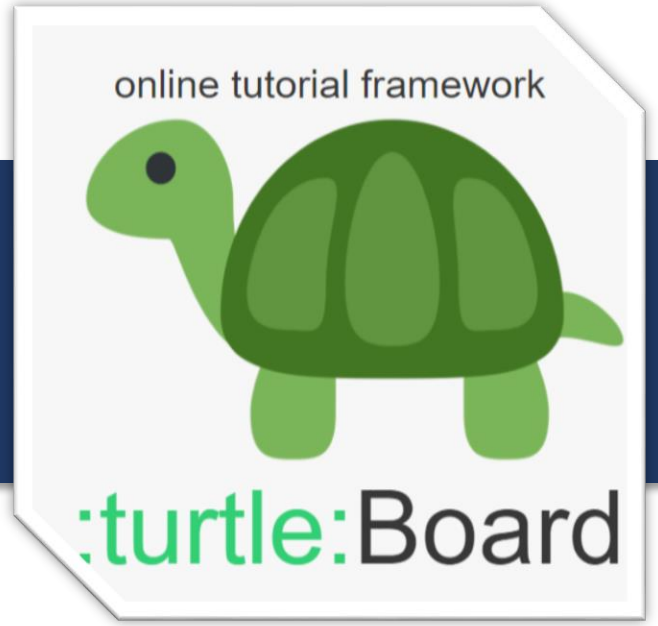
EIS
ENTERPRISE
INFORMATION SYSTEMS

UNIVERSITÄT BONN



MA-INF 3232 Lab Enterprise Information Systems

Development of an Ontology Tutorial for TurtleEditor

Gligor Tasev, Ludmila Casautan, Louai Haddad

Bonn, Germany 27/09/2017

Problem

- Many ontology tutorials exist. All of them → offline with an installed ontology editor;
- Lack of step-by-step guidance, feedback and poor didactics;
- No possibility of using hints/help;

Relevance and Importance

- Implementation of a chosen tutorial by the user into TurleEditor;
- Everything in one place;
- Online;
- Sharable;
- No installation costs;
- Easy way to learn ontology engineering;
- Getting feedback;

Challenges

- Specification of the structure of the input files (guidelines, hints, solutions);
- Parsing the files and displaying the outputs;
- Checking user's input and types of mistakes;
- Classification of user's mistakes;

Proposed Solution

review the TurtleEditor source code

+

review existing JavaScript turtorial frameworks

+

review of ontology tutorials

=

Online Tutorial Framework

step-by-step

guidelines

examples

tasks

hints

mistakes

scores

RDW/OWL

Javascript/PHP

HTML/CSS

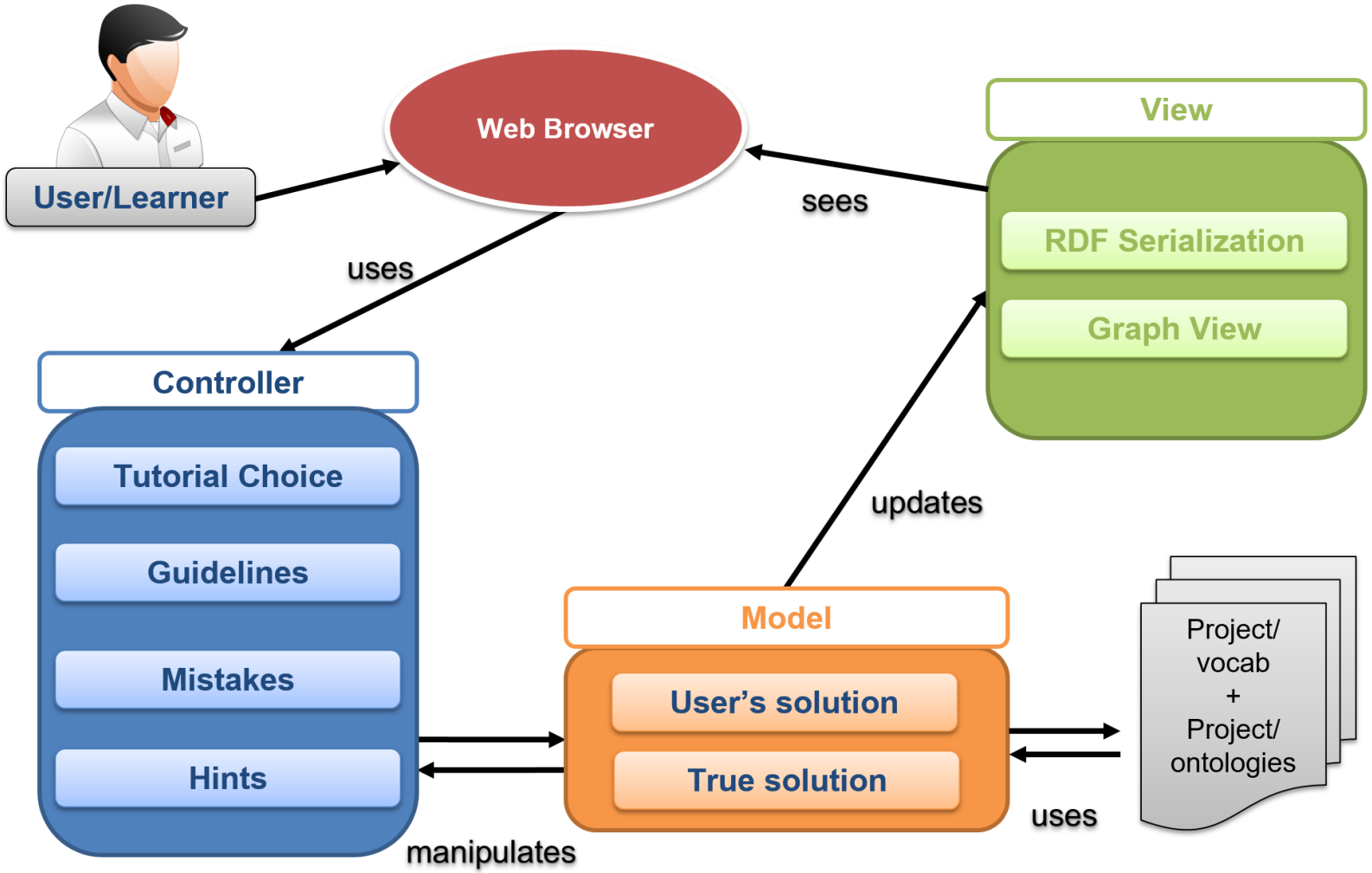
Implementation Details

- input files
  (**guidelines, hints → .ttl file**), (solutions → .ttl file);

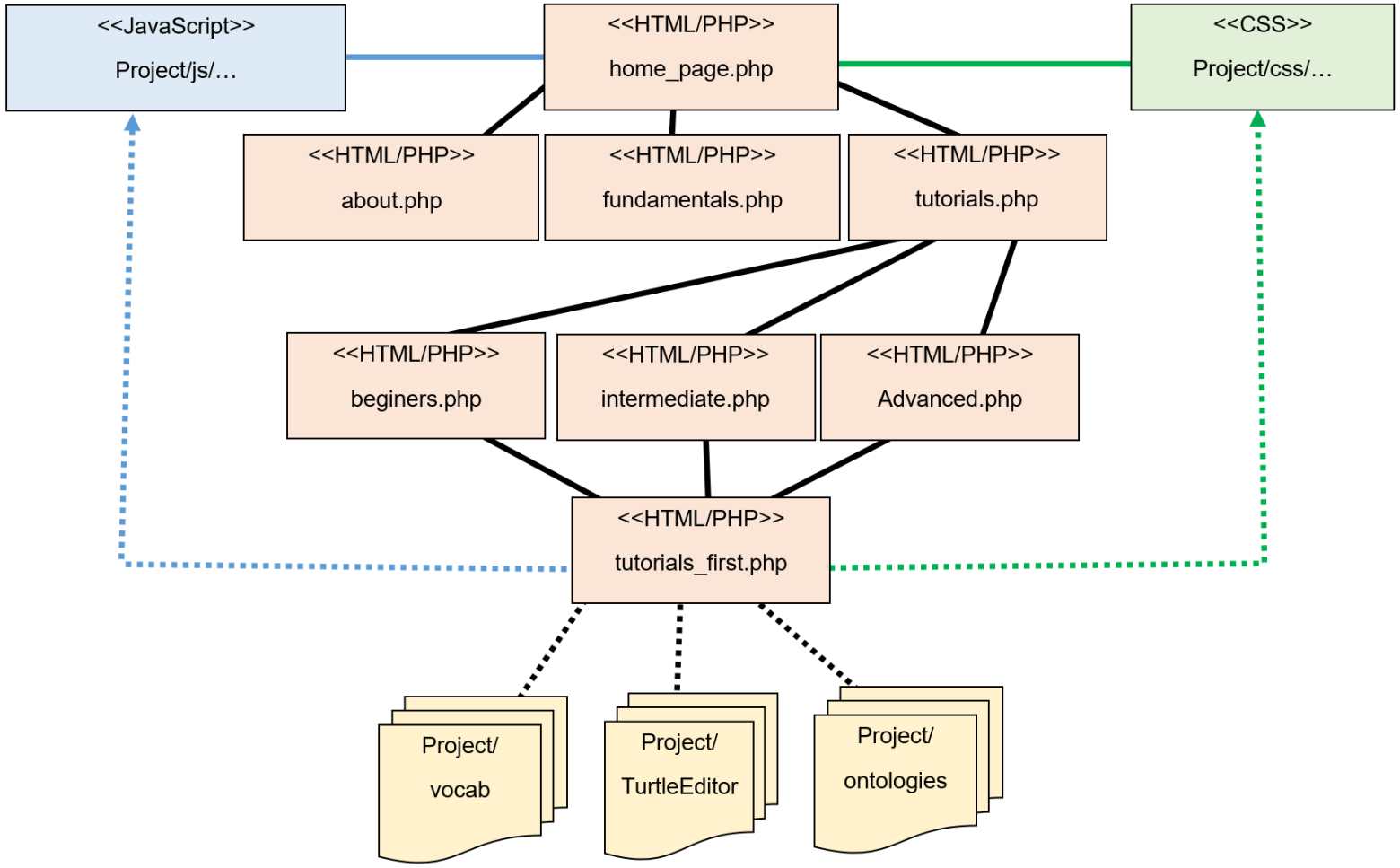
```
ex:Step1
    ex:id "1";
    ex:exampleText "To say that the dog is called Oddie and Odie is an English name you have to write a triple in the following way:";
    ex:exampleAnswer "ex:dog ex:isCalled 'Odie'@en.";
    ex:question "Please write a triple where the owner has a name Jon and this name is an English name.";
    ex:sparqlAskValidation "ex:owner ex:hasName 'Jon'@en.";
    ex:hint "When the predicate consists of two words, the we use the lowerCamelCaseNotation.".
```

- Parser for .ttl files and organization of different triples depending what do they represent (rdfstore);
- Checking user's input
  (ASK queries + comparison function: S,P,O);
- Printing mistakes and errors (syntax & semantics);
- Ontology tests;

Model-View-Controller Diagram



JavaScript Data Diagram



Use Cases Details

- Actors:

User/Learner

Developer

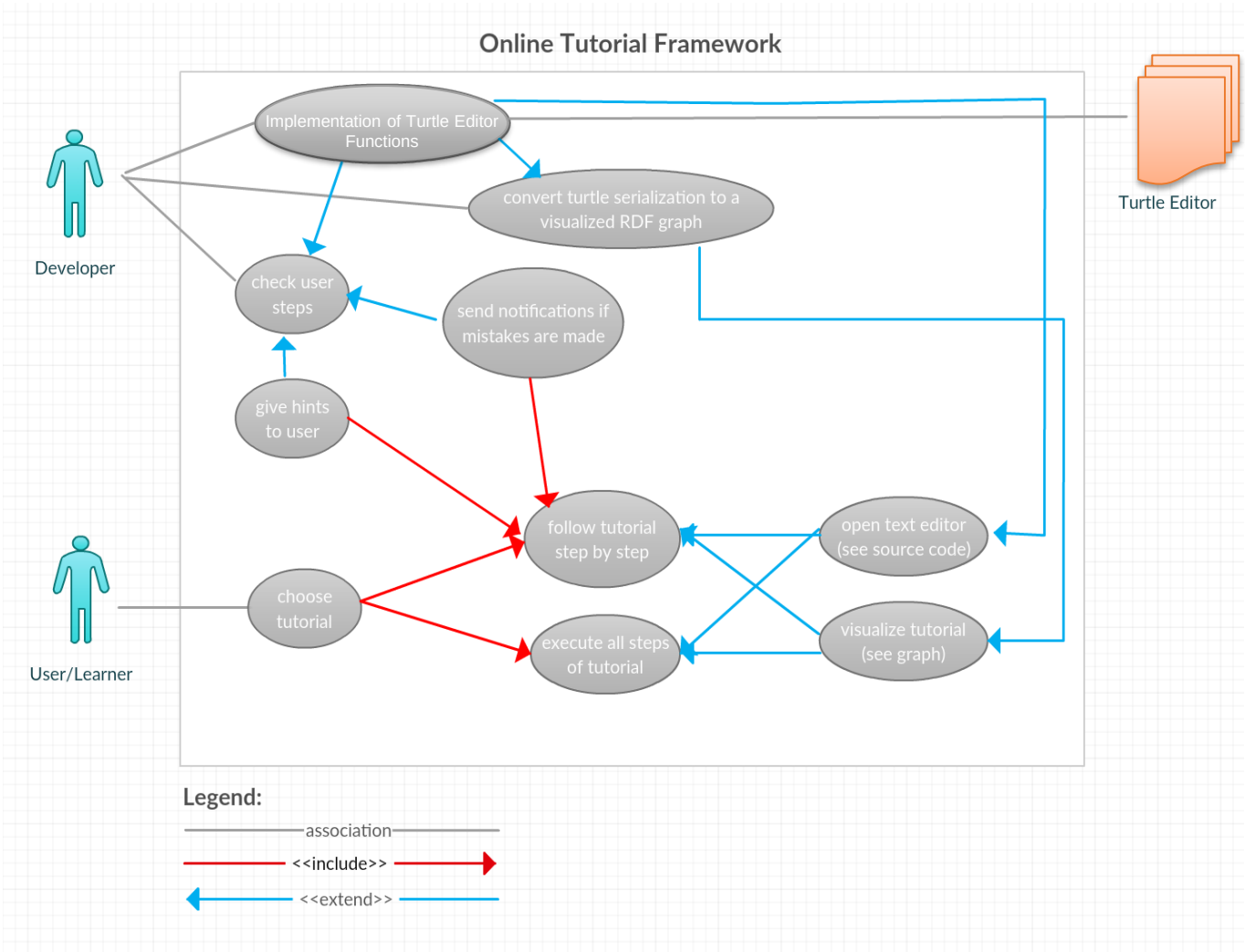
- Profile of the targeted user:

someone interested into learning Turtle serialization using online tutoring framework;

- Preconditions:

Choosing a tutorial from the framework depending on the level of previous knowledge;

Use Cases 1/2


Online Tutorial Framework
Implementation of Turtle Editor Functions
convert turtle serialization to a visualized RDF graph
Turtle Editor
Developer
check user steps
send notifications if mistakes are made
give hints to user
follow tutorial step by step
open text editor (see source code)
choose tutorial
execute all steps of tutorial
visualize tutorial (see graph)
User/Learner
Legend:
association
<<include>>
<<extend>>

Use Cases 2/2

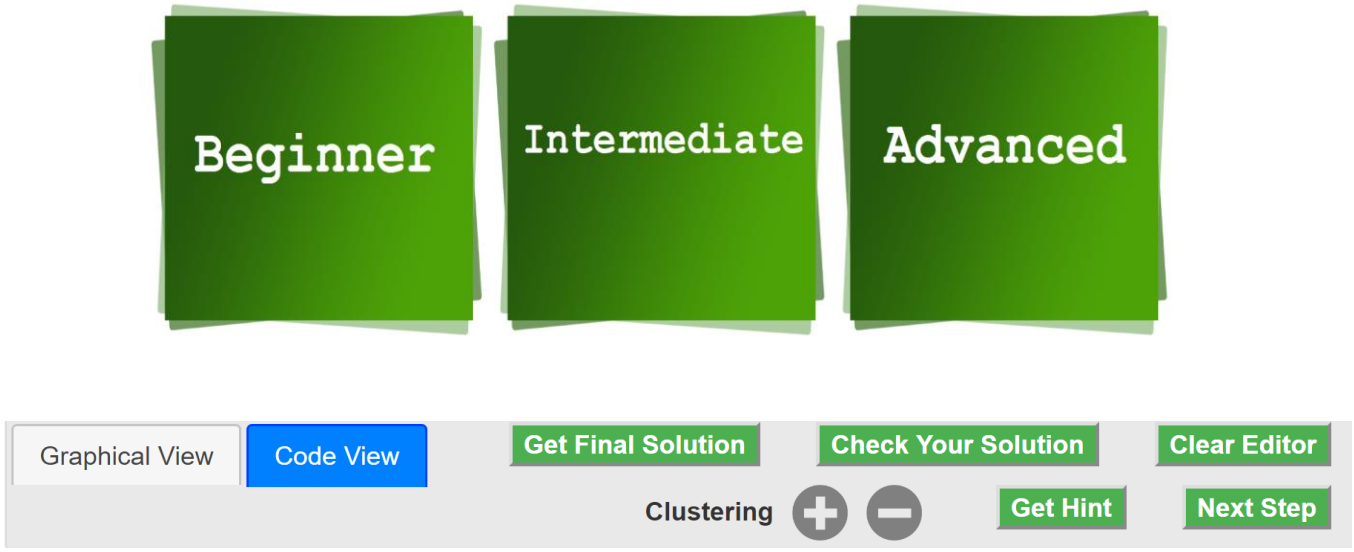
- **User/Learner**

Read Turtle Fundamentals

Choose a tutorial

- Follow the tutorial step-by-step
- Get hints
- Get feedback
- Get the input checked

Choose a tutorial




DEMO

Lessons Learned

- Anything can be represented in terms of RDF graph and after that used for a specific purpose;

- Turtle serialization → human readable
  (many things can be done in terms of structure representation);

- The amount of details in tutorial depends on the specified vocabularies.

- Better step guidelines and hints, detailed score feedback
  → less mistakes, better results;

Conclusions & Future work

- **User Profile**
  (registration, score analysis, user's learning timeline, logins, graphics, statistics);

- Variety of **vocabularies**
  (more choices, better user satisfaction, shuffling of tutorials);

- **Lessons** before each tutorial
  (preparation for the chosen tutorial, better user performance);

- Checking onload() - automatically;

- Using the **same framework structure** for implementation of a new tutorial frameworks for **other RDF serializations**;


THANK YOU