
# Deliverable 3.2

## Specification of the Learning Progress Model

| DISSEMINATION LEVEL | | |
|---|---|---|
| **PU** | Public | **X** |
| **PP** | Restricted to other programme participants (including the Commission Services) | |
| **RE** | Restricted to a group specified by the consortium (including the Commission Services) | |
| **CO** | Confidential, only for members of the consortium (including the Commission Services) | |

| COVER AND CONTROL PAGE OF DOCUMENT | |
|---|---|
| Project Acronym: | **INTUITEL** |
| Project Full Name: | Intelligent Tutoring Interface for Technology Enhanced Learning |
| Grant Agreement No.: | 318496 |
| Programme | FP7-ICT-2011.8, Challenge 8.1 |
| Instrument: | STREP |
| Start date of project: | 2012-10-01 |
| Duration: | 33 months |
| Deliverable No.: | D3.2 |
| Document name: | Specification of the Learning Progress Model |
| Work Package | 3 |
| Associated Task | 3.2 |
| Nature [1] | R |
| Dissemination Level [2] | PU |
| Version: | 1.0 |
| Actual Submission Date: | 2013-06-30 |
| Contractual Submission Date | 2013-06-30 |
| Editor:<br>Institution:<br>E-mail: | Alexander Streicher, Florian Heberle, Bela Bargel<br>Fraunhofer IOSB, Hochschule Karlsruhe (HSKA)<br>intuitel@iosb.fraunhofer.de, florian.heberle@intuitel.eu |

The research leading to these results has received funding from the European Community's Seventh Framework Programme (FP7-ICT-2011.8, Challenge 8.1) under grant agreement no 318496.

The author is solely responsible for its content, it does not represent the opinion of the European Community and the Community is not responsible for any use that might be made of data appearing therein.

[1] **R**=Report, **P**=Prototype, **D**=Demonstrator, **O**=Other

[2] **PU**=Public, **PP**=Restricted to other programme participants (including the Commission Services), **RE**=Restricted to a group specified by the consortium (including the Commission Services), **CO**=Confidential, only for members of the consortium (including the Commission Services)

# Change Control

### Document History

| Version | Date | Change History | Author(s) | Organization |
|---|---|---|---|---|
| 0.01 | 2013-03-25 | Document drafted | Bela Bargel | Fraunhofer IOSB |
| 0.02 | 2013-04-23 | Document updated | Florian Heberle | HSKA |
| 0.03 | 2013-04-26 | Document updated | Bela Bargel, Alexander Streicher, Daniel Szentes, Florian Heberle | Fraunhofer IOSB |
| 0.04 | 2013-04-30 | Document updated | Bela Bargel, Florian Heberle, Alexander Streicher | Fraunhofer IOSB, HSKA |
| 0.05 | 2013-05-10 | Document updated | Florian Heberle | HSKA |
| 0.06 | 2013-05-14 | Document updated | Alexander Streicher | Fraunhofer IOSB |
| 0.07 | 2013-05-22 | Document updated | Florian Heberle | HSKA |
| 0.08 | 2013-05-24 | Document updated | Alexander Streicher | Fraunhofer IOSB |
| 0.09 | 2013-05-27 | Document updated | Alexander Streicher, Florian Heberle | Fraunhofer IOSB, HSKA |
| 0.10 | 2013-05-31 | Document updated | Florian Heberle | HSKA |
| 0.11 | 2013-06-06 | Document updated | Florian Heberle | HSKA |
| 0.12 | 2013-06-10 | Document updated | Florian Heberle | HSKA |
| 0.13 | 2013-06-17 | Document updated | Florian Heberle | HSKA |
| 0.14 | 2013-06-21 | Document updated | Florian Heberle, Alexander Streicher, Peter Henning | HSKA, Fraunhofer IOSB |
| 0.15 | 2013-06-26 | Review | Alexander Streicher, Florian Heberle, Peter Henning | Fraunhofer IOSB, HSKA |
| 0.16 | 2013-06-27 | Consolidation | Alexander Streicher, Florian Heberle | Fraunhofer IOSB, HSKA |
| 1.0 | 2013-06-30 | Finalization | Peter Henning | HSKA |

**Distribution List**

| Date | Issue | Group |
|---|---|---|
| 2013-03-27 | Request for comments | WP01, WP02, WP03 |
| 2013-04-26 | Request for comments | WP01, WP02, WP03 |
| 2013-04-30 | Request for comments | WP01, WP02, WP03 |
| 2013-05-28 | Request for contributions | HIT, UVIE |
| 2013-05-31 | Request for comments | WP01, WP02, WP03, WP04, WP05 |
| 2013-06-06 | Request for comments | WP01, WP02, WP03, WP04, WP05 |
| 2013-06-10 | Request for comments | WP01, WP02, WP03, WP04, WP05 |
| 2013-06-21 | Review | WP01, WP02, WP03, WP04, WP05 |

**List of Contributions**

| Date | Organization(s) | Person(s) | Contribution |
|---|---|---|---|
| 2013-03-28ff | HIT, UVIE, UVA, UNIR, HSKA | Eran Gal, Christian Swertz, María Jesús Verdú, Luis de la Fuente, Florian Heberle, Dan Kohen | Comments reg. LPM input and output, also naming issues |
| 2013-05-06ff | FZI, HIT, HSKA, UNIR, UVA, UVIE | Jürgen Bock, Eran Gal, Florian Heberle, Peter Henning, Luis de la Fuente, María Jesús Verdú, Christian Swertz, Dan Kohen | General discussion about Learning Pathways |
| 2013-05-14 | UVIE | Christian Swertz, Alexandra Forstner, Alexander Schmölz | Input to user data, CM-related recommendations, Learning Pathways |
| 2013-05-15 | HIT | Eran Gal, Dan Kohen, Miki Ronnen | Remarks on learning habits, comments on didactic factors |
| 2013-05-17 | FZI, HSKA, IOSB | Jürgen Bock, Stefan Zander, Florian Heberle, Alexander Streicher | Technical discussion and development of a technical concept for the Back End |
| 2013-06-03 | HIT, UVIE | Eran Gal, Dan Kohen, Miki Ronnen, Christian Swertz, Alexandra Forstner, Alexander Schmölz | Didactic Factors |
| 2013-06-10 | UNIR | Luis de la Fuente Valentín | Overall comments |
| 2013-06-10 | HSKA | Peter Henning | Overall comments |
| 2013-06-14 | UVA | María Jesús Verdú | Overall comments |
| 2013-06-14 | UVIE | Christian Swertz | Overall comments |

| | | | |
|---|---|---|---|
| 2013-06-19 | HSKA | Peter Henning | MCS and Hypercube |
| 2013-06-23 | HIT | Eran Gal, Dan Kohen | Review, Comments on Didactic Factors and transformation rules |
| 2013-06-25 | UVIE | Alexandra Forstner, Alexander Schmölz | Review, Update of pedagogical input, comments on Didactic Factors and transformation rules |
| 2013-06-25 | UVA | María Jesús Verdú | Review, Additions to glossary, comments on pedagogical input, Didactic Factors and transformation rules |
| 2013-6-30 | HSKA | Peter Henning | Finalization |

# Table of Contents

# List of Figures

# List of Tables

# 1 Glossary

| Term | Explanation |
|---|---|
| Concept Container (CC) | Type of Learning Object on the lesson level. A CC is part of a Knowledge Domain (KD) and contains multiple Knowledge Objects (KOs) to form a lesson. |
| Content creator | Person, usually some kind of lecturer, or a domain or ontology specialist, who fulfils the task to create learning material, which can, when completing the whole INTUITEL specific workflow, be used in the recommendation creation process. |
| Cognitive Model (CM) | Description of a domain of knowledge using the terms and relations as specified in the Pedagogical Ontology. |
| Didactic Factor (DF) | Nominal (i.e. non-numeric) value that is composed of a number of data items that are relevant for INTUITEL in order to create an assertion about the learning situation, habits or preferences of a particular learner and, therefore, to recommend suitable KOs and to generate learner feedback. |
| Description of Work (DoW) | Description of Work of the INTUITEL project. |
| Knowledge Domain (KD) | Topmost cognitive container of an INTUITEL enhanced course. The KD element encapsulates a set of Concept Containers to create an outline for a domain of knowledge. Such CCs are e.g. thematically related collections of pages (learning modules), tests, surveys, files, media objects. |
| Knowledge Object (KO) | In INTUITEL a Knowledge Object (KO) is an item of knowledge, which typically corresponds to one screen page of content and to an estimated learning time of 3-10 minutes for the average learner. |
| Knowledge Type (KT) | Categorization of Knowledge Objects according to the type of knowledge that is contained, i.e. their function within the learning process. Different Knowledge Types are defined for different learning approaches: multi-stage-good-practice, multi-stage-simulation, open-inquiry-based and structured-inquiry-based. |
| Learner | A learning person using the INTUITEL enhanced LMS. |
| Learning Management System (LMS) | Software application with which content creators, lectures or learners interact. It provides mechanisms to administrate, document, track, report and deliver the educational content. |
| Learner Progress Model (LPM) | Central mediating component of the INTUITEL system which acts as a data transformation and distribution entity. |
| Learning Object (LO) | Learning Object (LO) is in INTUITEL the umbrella term for the various types of learning containers, such as Knowledge Domain (KD), Concept Container (CC) and Knowledge Object (KO). |
| Learning Object Recommender (LORE) | Interface provided by the LMS to show the learner situational relevant recommendations computed by INTUITEL. |
| Learning Pathway (LP) | “Map” describing how to find a suitable route to get the objectives of a course. |
| Media Type (MT) | Categorization of Knowledge Objects according to the type of contained media. |
| Multidimensional Cognitive | MCS is an abstract space in the form of a hypercube. Each coordinate axis is |

| | |
|---|---|
| Space (MCS) | associated with a KO and the position of the learner along this axis indicates how much of this KO the learner has learned (0-100%). |
| Web Ontology Language (OWL) | Language for describing and sharing ontologies on the World Wide Web. |
| Pedagogical Ontology (PO) | OWL-ontology, which provides terms and relations to model courses and learning pathways in a semantically rich way, in order to enable the INTUITEL Back End to automatically create learning recommendations. |
| Rating Factors (RF) | Indications of KO suitability for the learning process when certain situational dependent conditions (i.e. Didactic Factors) are present. |
| Semantic Learning Object Model (SLOM) | Format combining metadata and learning content for the INTUITEL system. |
| Tutorial Guidance (TUG) | Tutorial Guidance is an interface provided by the LMS. |
| User Score Extraction (USE) | User Score Extraction is an interface provided by the LMS. |

# 2 Introductory Remarks

The Learning Progress Model, or short LPM, is the central component of the INTUITEL system, connecting all other components. It is the first of the two big modules in the INTUITEL Back End and acts as a preliminary stage for the second one, the INTUITEL Engine (WP05). By providing functions to perform transformations of learner scores, history and pedagogical as well as domain knowledge into a position within the Multidimensional Cognitive Space,, the LPM prepares the data to be useable in context of the reasoning process.

In order to achieve this, it relies on multiple data sources about the users. First, there is the data about the learner as represented in the LMS, queryable via the USE interface (WP01, D1.1). Secondly, the pedagogical and domain knowledge as specified in WP02, in form of the Pedagogical Ontology (D2.2) and the respective Cognitive Models, provide information about the actual learning content and how to guide a learner through the learning material. And in the third place, a user model of internal INTUITEL relevant user data like session data or the navigation history.

The raw input from the data sources is grouped, i.e. classified, reducing the dimensionality of the input space. Mathematically this amounts to a projection on the cognitive space (which may have quite a few dimensions) from the even higher dimensional space of observable input.

## 2.1 Objectives

The introduction and specification of the LPM is part of this document (D3.2), its objectives of are:

- Formulation of a Learning Progress Model, e.g. as a set of rules describing the transformation of learner scores into a position within the Cognitive Model, drawing conclusions on necessary feedback and recommendation of further learning material.
- This set of rules will be derived from the Pedagogical Ontology (WP02) and the Cognitive Models for knowledge domain A – D.
- Also in this Task, the knowledge domain experts, the LMS experts and the ontology experts will work very closely together, resulting in a description of which results in the USE reading will be transformed to what dialog components and recommendations to the TUG/LORE interface.
- Specification of a set of rules for pedagogical reasoning in textual as well as processable format.
- Description of how a given learner situation has to be translated into a Learning Object recommendation and tutorial guiding dialog components.
- Description of a proper ontology for the LPM.

To summarize, the task of the LPM is to collect various kinds of information from different sources and to transform them into a format that allows the INTUITEL Engine to draw conclusions on that data.

## 2.2 Further Aspects

The results of the project INTUITEL so far show that the LPM needs to cover further aspects to achieve these goals. This mainly includes the implementation of the following aspects:

1) Communication: Development of an interface that allows the information exchange between the LPM and the SLOM metadata repository, the LMS and of course the INTUITEL Engine. This is already implicitly included in the task description, because the LPM uses learner scores as input, it is nevertheless noted here explicitly.

2) Session management: Monitoring who is currently working on which course(s) and when they open which LO. Since INTUITEL is a completely reactive system (cf. section 2.3 in D3.1), this is the essential trigger which starts the recommendation creation process.

3) User database: INTUITEL needs to include the learner history (e.g. transitions between LOs), the deductions it has made in previous situations and the respective reactions of the learners. This is necessary because some deductions rely on this knowledge (e.g. for the deduction of the next step in a Learning Pathway). Storing these pieces of information in a consistent and suitable way is another important, non-trivial aspect of the LPM.

4) Management of Learning Pathways: INTUITEL has, via the specification of the Pedagogical Ontology (D2.1), the capabilities of applying different Learning Pathways for individual learners and courses. However, one goal of INTUITEL is to go one step further and to deduce which LP is best for the learner. To do that, the LPM needs to implement a method to calculate when and how a learner-specific Learning Pathway should be recommended. This enables the LPM to analyse and react on unexpected learner behaviour. Since the learner is free to choose any Learning Object, INTUITEL learns from this and includes the learner's choices in upcoming recommendations.

The LPM itself is not only a model to describe the learning progress, as stated by its name, but also a set of different software modules that interact to provide the necessary tools and information for said model. The specification of the LPM in context of this document does thus not only cover a conceptual specification on how to prepare data for the reasoning process, but also how the LPM maintains a consistent availability of the relevant information, i.e. how it gathers them and how the different modules interact to achieve this. Additionally some remarks on its implementation as a Java program will be given.

## 2.3 Dependencies

The LPM is one of the main research aspects of the INTUITEL project. Because of the interdisciplinary character (i.e. pedagogical and technical aspects) various inputs from multiple partners must be taken into account.

Being based on the results and on-going work of D1.1 (Data Model), D2.1 (Pedagogical Ontology), D3.1 (Overall System Design) and also the SLOM specification D4.1, the LPM development efforts must be well attuned. This is necessary because in order to develop a system capable of transforming

various kinds of input to applicable statements about learning, a sophisticated theoretical foundation and a representative set of didactic factors as input variables is mandatory. Otherwise, the deliberations and development would not lead to results that are compliant to the previous results and the interplay of the multiple components in INTUITEL.

## 2.4 Structure of this Document

After the preceding introductory remarks, **chapter** 3 gives a general overview on the LPM as a module in INTUITEL and how it is structured.

**Chapter 4** defines and explains the different input types for the LPM, including the data from the LMS TUG interface, the Learning Pathways, the data from the SLOM repository as well as the preprocessed input like the user database and session management.

**Chapter 5** explains the concept of the learner position within the Cognitive Content Space (CCS) and the Multidimensional Cognitive Space (MCS). Some examples for a learner position in the MCS are given.

**Chapter 6** provides an introduction into how the Back End works as a whole. With this background knowledge, it is possible to grasp how the INTUITEL recommendation system will work and how the LPM is linked to this from a structural, conceptual and technical perspective.

**Chapter7** explains the role of the LPM in the INTUITEL Back End in more detail. Its structural concept and the relevant aspects are explained and an illustrating example is given.

Afterwards, **chapter 8** gives an overview of the Didactic Factors and an impression of what the LPM will take into account in the creation of LO suggestions. Included in this section are the transformations rules which specify how to transform input data to the Didactic Factors.

**Chapter 9** describes the envisaged realization of the LPM as a piece of software and how it may be implemented.

Concluding, **Chapter 10** summarizes the LPM.

# 3 Overview

This chapter gives a general overview on the LPM as a module in INTUITEL and how it is structured.

## 3.1 Abstracted Data Flow from LPM Perspective

This section describes how INTUITEL processes data from the perspective of the LPM to create an added value for the learning process. Basically there are four relevant elements, which are introduced here briefly. A more detailed explanation is given in the rest of this document.

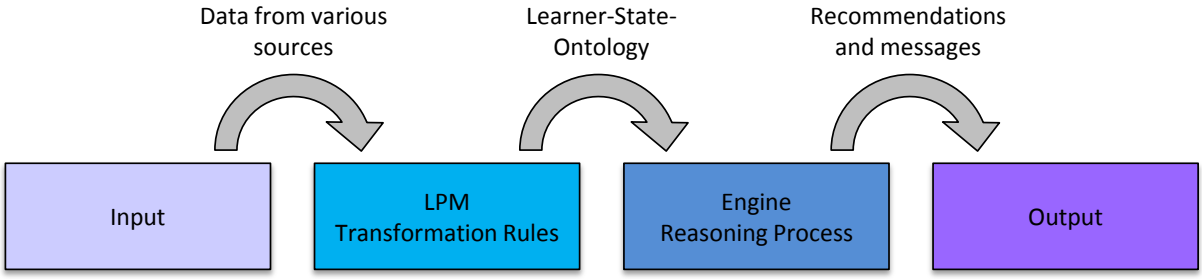


Figure 1: The basic elements in the INTUITEL data flow as seen from the LPM

Firstly, there is the **input**. This is the sum of all collected data provided for the INTUITEL Back End, including the personal and situational information about each individual learner as well as the basic didactic and domain specific data. This establishes the foundation for the deductive process that is carried out in the Back End. The more and the more precise data is available at that point, the more aspects can be included in the process and consequently lead to a better, more versatile and more individual recommendation. However, it is expected that the dimensionality of this raw input space varies dramatically among different Learning Management Systems (LMS), but may also change with time when new developments are integrated. As an example, consider the measurement of stress factors, eye movement or emotional factors during learning.

Secondly, the dimensional reduction work of the **LPM** takes place, reducing the possibly huge number of raw data items to some pedagogically useful classes. This mathematical projection however is far from trivial, because a single measurement (e.g. of learning speed) must be considered in view of other factors (e.g. learning situation). In mathematical terms, this would amount to a non-orthogonal projection. While it is not the goal of INTUITEL to perform an exhaustive study of the possibilities offered here, it is nevertheless intended to provide working examples for such dimensional reduction transformations. The goal of this step is therefore to create a custom ontology, the Learner-State-Ontology (cf. section 7.3 for more details), containing all relevant information for the INTUITEL Engine. The LPM will therefore rely on the Pedagogical Ontology and the LM Ontology[3] which contain the specification of the rules, allowing the Back End to rate the suitability of certain LOs for the learner's current situation. Especially relevant for this step is the

[3] See Task 3.4 of INTUITEL (Learning Model Ontology) which is concerned with lifting the LPM on the semantic level.

OWL-based description of the Didactic Factors (cf. section 6.4), which are the building blocks of the more complex Rating Factors (cf. section 6.5). A reference to a Java[4] class in each Didactic Factor will allow the LPM to create trivial to complex statements about learning as an input for the INTUITEL Engine.

Thirdly, the **INTUITEL Engine** uses the output of the LPM, i.e. the Learner-State-Ontology, to determine which KOs are suitable for the current situation of the learner. This basically is a reduction of the set of available LOs in dependence of the learner's Learning Pathway and the different rating aspects. The INTUITEL Engine will therefore run queries on this custom ontology to identify these KOs. The thereby created results are then restructured to contain the relevant KO data. An extra module, the Natural Language Unit, will afterwards use the results of the whole process to create message(s) for the learner if necessary.

This ends up with the fourth element, the **output** of the Back End. After restructuring the previous results in accordance to the INTUITEL Data Model, the learner specific recommendations and natural language messages are sent to the LMS and finally delivered to the learner.

## 3.2 Architectural Overview and Data Flow

As seen from an overall architectural perspective, the LPM is part of the INTUITEL Back End, which also contains the INTUITEL Engine and the Back End communication interface. Figure 2 graphically shows how the LPM is positioned in the INTUITEL overall system design. Please note that this diagram is simplified to allow an easier perception of the interplay of the individual components. For a more detailed overview, please consult deliverable 3.1.

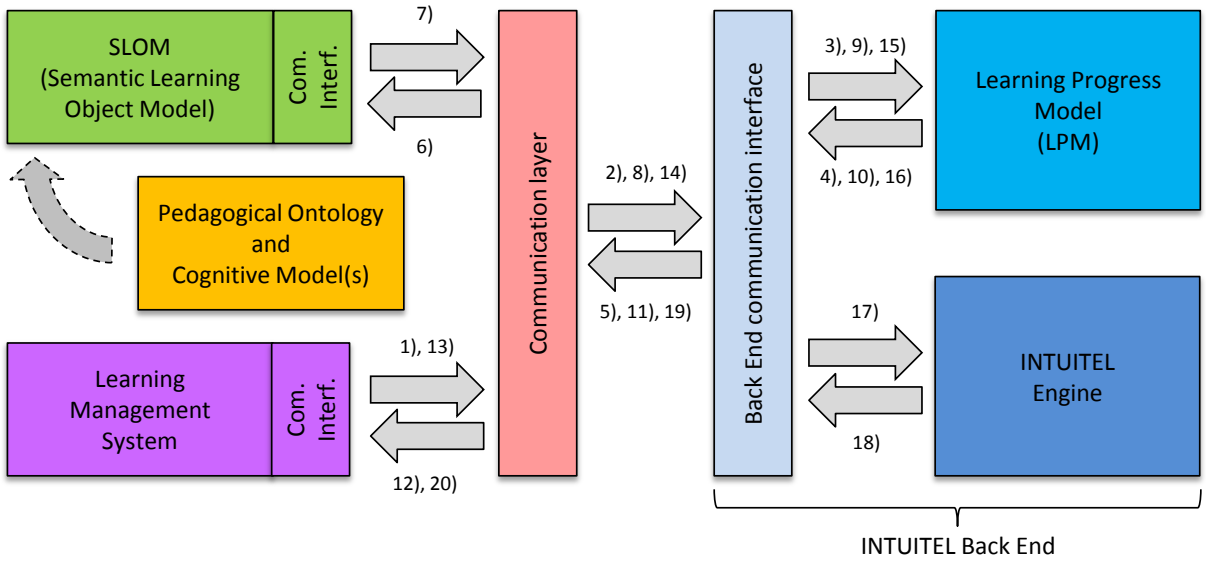


Figure 2: Simplified system architecture and data flow for the recommendation creation

[4] The LPM software components will be implemented as Java software.

The numbered arrows in Figure 2 describe the INTUITEL information flow beginning at the moment when a new Learning Object recommendation creation is triggered until the delivery to the learner. The individual steps are described in the following:

| Step(s) | Description |
|---|---|
| 1), 2) | The LMS sends a message to INTUITEL specifying that a certain learner has opened a new LO[5]. Instead of sending it directly to the LPM, the message is send to the communication layer, which thereafter forwards the message in step 2. This is necessary because the INTUITEL system could, but does not necessarily have to run on only one machine. The communication layer hides this aspect from the LMS in order to allow an easier and more lightweight implementation. |
| 3) | The Back End communication interface forwards the message to the LPM, which itself analyses the message. Depending on the current situation and available data, the LPM starts requesting different types of information from the other components. |
| 4) - 9) | Given that the respective course metadata (CCs and KOs as SLOM metadata) is not already present in the LPM, it requests it from the SLOM metadata repository. |
| 10) – 14) | In order to get the latest situational/environmental data and scores of the last KO of the respective learner, the LPM sends a USE request to the LMS. As before in the SLOM metadata request, the different communication interfaces and the communication layer forward the messages to their destinations. |
| 15) | After receiving all relevant data, the LPM applies its transformation rules and creates the input data for the reasoning process of the INTUITEL Engine. |
| 16), 17) | The reasoning data is sent to the INTUITEL Engine, which receives it in step 17. Depending on the individual INTUITEL realization, it might be possible that the LPM and the Engine run on different machines. In that case, the Back End communication interface uses the communication layer as an intermediate step. |
| 18) – 20) | When the reasoning process is complete and the recommendations and/or the natural language messages have been created for the learner, the Engine sends a LORE/TUG message in steps 18 to 20. The LMS uses this data to present the information to the learner in an LMS-specific and INTUITEL compliant way.[6] |

Table 1: Steps of the overall data flow

As may be seen, the LMS is the starting point of this interaction. The reason for this is that INTUITEL is designed to be a reactive system. It only performs actions if a user action happed earlier. That also implies that INTUITEL only creates recommendations for learners if the Back End is notified that a learner actually needs a new one, i.e. the LPM receives a message specifying that a certain learner opened a particular LO in the LMS.

## 3.3 LPM as a Mediator

Although being a “passive” component in this process, from the data flow perspective the LPM actually is the central element in INTUITEL, as illustrated in Figure 3.

[5] Actually, there are three different ways how this notification can happen. Which one is used depends on the LMS and its configuration. For simplicity, this example assumes that the LMS actively notifies INTUITEL.

[6] Please note that, due to simplicity, it has not been depicted that this information is also forwarded to the LPM to also incorporate previous recommendations in later reasoning processes.

This underlines the importance of the LPM. Since it initiates most of the data exchange between the major components in INTUITEL, the LPM, as a software module, can be seen as an information broker that triggers the data exchange in the overall system. It is thus more than a pure model of the learner – it is actually some kind of mediator. It brings the various pieces of data and information together and transforms them into a usable format containing more information than the original material itself.

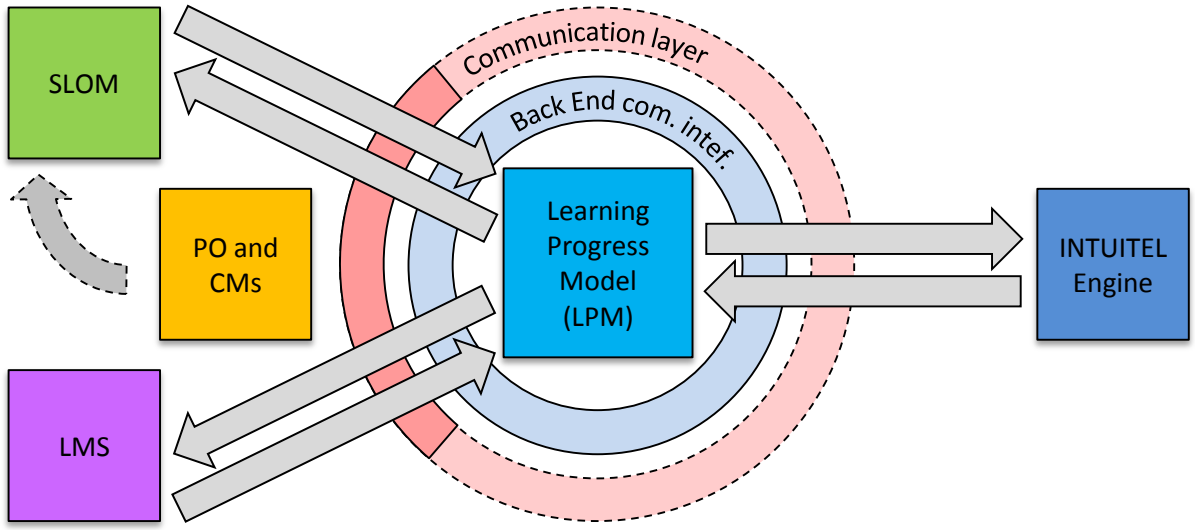


Figure 3: Simplified data flow as seen from the LPM – the LPM acts as an information mediator

## 3.4 Comparison to Real-Life Tutoring

Speaking in terms of a real-life learning scenario, INTUITEL is a workflow engine to process well established learning methods, in particular by deducing relevant and individual recommendations for learners. Without loss of generality, the following outlines the analogy between the decision process of a (simplified) real teacher and INTUITEL.

| Pedagogical Aspect | INTUITEL Analogy |
|---|---|
| A human teacher senses when a learner needs assistance. The sensing process may rely on subconscious experience based processes as well as involve cognitive reasoning | Via the LMS-interfaces, INTUITEL registers and protocols the current and precious states of the learner. The LPM serves as the *sensing* instance, either by simple rules based inference or by more complicated ontology based reasoning. |
| In looking at test results and carrying out a spoken dialog, a teacher tries to assess the learner's current needs. | Usage of personalized information INTUITEL has collected for each learner and request of up-to-date information via USE or TUG dialog. |
| By applying personal didactic/domain knowledge or by intuitive reaction, the teacher forms an opinion about the learner's situations. | Application of transformation rules to create the different Didactic Factors (cf. section 6.4) and their integration in the learner-specific Learner-State-Ontology. |

| | |
|---|---|
| The teacher evaluates the situation on a topical basis and aligns it to his or her personal knowledge about the topic. | With the Cognitive Model stored in the SLOM file, INTUITEL draws conclusions about necessary alignment of the learning process. |
| The teacher ponders which advice fits best. | Reasoning process in the INTUITEL Engine to select suitable KOs and/or the decision which natural language feedback should be given. |
| Teacher gives the learner an advice which fits his/her current situation best. | Recommendation of specific content via the LORE interface and/or natural language messages via the TUG interface. |

Table 2: Comparison between learner and INTUITEL behaviour in learner counselling

As it can be seen in this example, the workflow in INTUITEL is very much aligned to its real life model. With the LPM and the Engine as the modules that try to somehow re-enact or mimic the actions that a teacher's brain carries out in this process, the importance and complexity of the Back End is reinforced and re-emphasized.

## 3.5 Reflex Reactions

Like a human teacher the INTUITEL system should always be able to react to the learner. At best, the recommendation process is always based on sound considerations of all input information. But sometimes even a real teacher is not able to find the best recommendation to help the learner.

Possible reasons for that could be the lack of some specific knowledge or the deduction problem is too complex to be solved in an appropriate time. In such cases teachers must rely on their intuition. Their "reflexes" kick in and the teachers give nearly-as-good answers as recommendations. Ordinarily, the more experience a teacher has, the better the recommendations fit to the learner's situation and needs.

INTUITEL aims do the same in order to provide for the best possible guidance. It thus needs a mechanism which decides when to bypass the reasoning process and when to trigger a direct reaction.

Therefore, the LPM contains a specific module, the reflex module. Like in the real-life scenario, the reflex module acts when it is foreseeable that a response is not going to be created in time (e.g. when the CPU is at its full working capacity), or when it is obvious that a certain action has to take place (e.g. the connection to the SLOM repository with mandatory information is broken). Because a recommendation cannot necessarily be created in all cases, the reflex module triggers a message that either informs the learner that his recommendation is pending (e.g. "I am still thinking about it, just one moment please.") or enforces the creation of a specific question that the learner should answer (e.g. "Which Learning Pathway do you want to choose? Select one of the following …").

Another example would be to create a welcome message, if the learner just started a new session. For such a message, no computationally expensive reasoning has to take place. The LPM does thus not have to trigger a heavyweight reasoning process. This does not mean that the reasoning process will never the less take place afterwards. It is just a method to take some burden from the INTUITEL Engine in some predefined cases.

This is not only a reasonable reaction from an educational perspective, but also from a technical one. Due to the complexity of a full INTUITEL system in a distributed hosted scenario, avoiding unnecessary load is a good strategy to optimize the efficiency and performance of INTUITEL.

## 3.6 LPM Component Description

The LPM itself consists of a set of modules that manage the different tasks (see Figure 4 for a graphical overview).

**Session Manager:** This is the controlling element in the LPM. As a reactive system, INTUITEL awaits notifications on what the learners do in the LMS. The Session Manager encapsulates this functionality and tracks the LO transitions of the individual learners. If a transition between two INTUITEL-enabled LOs is noticed (i.e. the respective SLOM metadata for that course is available), the Session Manager starts the procedure which leads to a new recommendation.

**User database (interface)**: The user database is responsible for the persistent storage of all user-specific information that INTUITEL needs and collects through its runtime. Its interface provides an easy-to-use access to this information so that the other components, namely the Session Manager and the LP Selector, don't need to care for database related issues.

**Learning Pathway Selector**: Initially, INTUITEL assigns every learner a macro and a micro LP. It therefore selects them, depending on certain criteria (cf. section9.2), from the available set of LPs which come with SLOM metadata. However, they are not equally optimal for each learner. To help them to find their optimal learning procedure, this module evaluates the learner's progress. If the LP Selector notices that a learner deviates from the originally selected LP to a certain degree, it either assigns another standard LP or creates an own learner-specific one.[7]

**Reflex Module**: The Reflex Module provides methods to create reactions much earlier than using the Reasoning Engine. Like a person that reacts intuitively when some kind of stimulus occurs (e.g. the blink reflex that closes your eyes when something is approaching an eye), the reflex module acts when, for instance, a mandatory data input is missing. Cf. section 3.5 for more details.

**Rule Applicator**: This module applies the transformation rules to provide complex knowledge for the following steps on basis of the given input in a compatible format. In contrast to the INTUITEL Engine and although the rules are specified in OWL, the Rule Applicator does not use OWL-reasoning. It is implemented in Java and uses an OWL-framework to create statements about learning, i.e. the Didactic Factors (cf. section 6.4). The advantage of this approach is that the Java-based realization allows carrying out complex transformations, because all benefits of a high level programming language can be used and the whole input data is (directly) available.

**Learner-State-Ontology Creator:** The INTUITEL Engine needs its input in a suitable format in order to carry out the OWL-reasoning. Since one of the tasks of the LPM is to be a preliminary stage for the Engine that prepares the data respectively, the Learner-State-Ontology Creator takes the results of

[7] Please note that INTUITEL does not (necessarily) just assign these LPs without notifying or asking the learner. Via the TUG interface, INTUITEL can request specific feedback from the learner to make a suitable selection possible.

the Rule Applicator and all other relevant information to create a suitable output. It will furthermore have a feature to change and add KO-specific information on the basis of the available data to make a more accurate reasoning possible.

**Interface to INTUITEL Engine**: This module takes the data for the Engine and transmits them via the Back End communication interface. It will thereby be conformant to the guidelines that describe how data should be exchanged between those two components. By attaching a Dummy-Engine to this interface, the LPM will be testable on a functional basis.

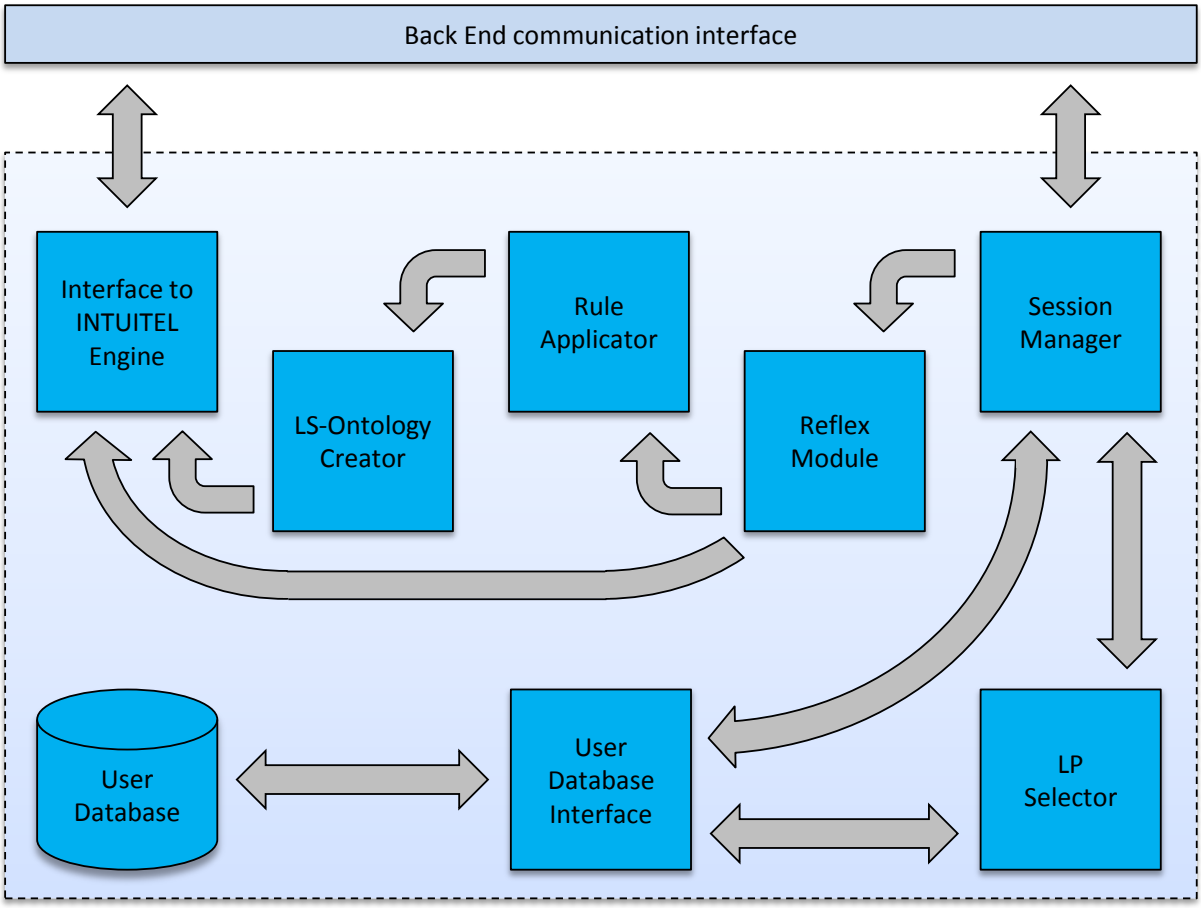


Figure 4: Simplified internal architecture and information flow in the LPM

For a more detailed description of the different modules of LPM, please see chapter 9, which contains fine-grained information on the individual components and their interplay.

# 4 LPM Input

This chapter defines and explains the different input types for the LPM. It firstly describes the learner dependent set of data that is aggregated via the USE interface and how TUG can help to collect the missing information. Further, the concept of Learning Pathways and their realization in the Pedagogical Ontology is outlined. Afterwards, the concept of the Cognitive Models is described and why a corresponding SLOM file is needed for the LPM. Finally, the input which is directly available via the internal modules is described, meaning the user database and the session management.

## 4.1 Learner Input

Learner input basically describes all personalized learner information INTUITEL collects from and via the LMS and which can be used as a source for the recommendation creation process.

This not only includes learner scores in terms of grades or results of tests, but goes much further. There is a multitude of different types or kinds of learner scores that provide diverse information about the learner from a learning habit, a learning progress and a situational perspective. They are mainly collected via the USE interface, but also the general services interface (i.e. via the Learner Update mechanism)[8]. Depending on the possibilities of the LMS and the available information, some of it might also be collected by directly asking the learner via the TUG interface.

Examples of the available learner data are, apart from the rather obvious question how good the learner is in terms of grades, amongst others:

- What and when did the learner access content?
- In which order did the learner access learning content?
- How long was the learner working on certain a Learning Object?
- Which kind of device is the learner currently using?
- How good is the learners Internet connection?
- Is the learner currently in a noisy environment?
- Other possible raw data items have been mentioned already, like emotional or stress measurements involving bodily interfaces to the learner.

In short, learner input describes all information on all aspects regarding how and what a learner currently and also previously has learned in the eLearning system across all courses.

For a detailed list and description which data can be accesses via the USE and TUG LMS interfaces, see the INTUITEL Data Model (deliverable 1.1). Examples are age, name, gender etc.

[8] For a more detailed description of the services that exchange data between the INTUITEL Back End and the LMSs, please see deliverable 1.1, the INTUITEL Data Model.

## 4.2 Pedagogical Input

The pedagogical input for the LPM consists of two parts. On the one hand, there is the terminology which has been specified in context of WP 02 – the Pedagogical Ontology. It provides a comprehensive set of entities for the modelling of learning material in INTUITEL. The structuring of learning material, independent of its actual LMS realization, allows a semantically rich description of eLearning courses. The central elements, namely Concept Containers (CCs) for abstract topic-based structuring of courses and Knowledge Objects (KOs) for the description of the actual content, allow the LPM to understand the meaning of the Learning Objects in the course.

However, what is more important for the pedagogical input is the second aspect, the availability of Learning Pathways. In order to guide learners through an eLearning course, INTUITEL needs a "map" describing how to find a suitable route. This basically is what Learning Pathways (LP) provide for the Back End. There is a basic set of INTUITEL-agreed on LPs available. They consist of four Macro Learning Pathways and six Micro Learning Pathways. The macro-level pathways are chronologically forward from old to new, chronologically backward from new to old, hierarchically bottom-up and hierarchically top-down. The six micro-level pathways can be categorized into four Knowledge Type Pathways and two Media Type Pathways. The Knowledge Type micro-level pathways are "Good Practice" multi-stage, simulated multi-stage, open-inquiry-based and structured-inquiry-based. The two Media Type Pathways are abstracting and concretise.

These modelling capabilities allow teachers to enhance their learning material with meaningful, multidimensional and pedagogically sophisticated structure that is not directly part of their course material. With this, the LPM and the INTUITEL Engine are able to deduce a didactically reasonable route through the learning material.

## 4.3 Domain Input

The previously described pedagogical input alone is as itself insufficient for the Back End, since it only provides the technical and didactical foundations. The description of the actual learning content is missing so far. In order to include this, the LPM needs the description of the knowledge domain from the lecturer. This is provided via the Cognitive Model (CM) and the SLOM metadata. Both are created by a domain specialist, but not necessarily the same person. The CM outlines the basic topics or so to say the curriculum of a certain domain of knowledge. This OWL-based description specifies which CCs are available in a specific Knowledge Domain (KD). The SLOM metadata connects this with a course in a LMS by completing the course specification with the semantically rich description of the KOs. The thereby defined metadata contains detailed information about the learning material itself (e.g. the contained media or what type of knowledge they represent).

To summarize, the domain input subsumes all information which the LPM and INTUITEL Engine need to understand the internal coherences in an eLearning course. This does not mean that INTUITEL understands the content itself, i.e. what the learner is trying to learn, but is able to react on the semantic data as mentioned before.

Technically, this is realized as a two tier solution. The information on the individual topics in the course is contained in the Cognitive Models. Such a model uses the terminology and relations from the Pedagogical Ontology and defines the internal relations (i.e. the macro Learning Pathways) of a Knowledge Domain (KD). This is the abstract top-level entity that subsumes all elements that describe a certain eLearning course in INTUITEL. The structuring entities in such a KD are the Concept Containers (CCs), which subdivide the domain in graspable, manageable and thematically related content clusters. The second tier is the SLOM metadata, which contains the description of the actual learning material, the Knowledge Objects (KOs)[9].

Take, for example, the simplified eLearning course "Math for primary-school pupils" (see Figure 5). Consisting of a number of pages introducing children to the basics of what numbers are and the basic arithmetic operations addition and subtraction, it depicts a very brief example of domain input in INTUITEL. On the first level, there is the start point of the course itself, the KD. Branching from there, the individual CCs are connected via macro LPs and allow navigating between them in a didactically reasonable way. The KOs are attached to the CCs and the micro LPs specify how a learner should best range between them on basis of the KO metadata. With this information, the Back End can deduce an optimal route for each individual learner, based on how many different LPs are available for that specific course.

Although this is not categorized as domain input, it should be noted that the individual Learning Pathways between the elements are firstly possible because the domain input provides a description of the content between which a route is possible. INTUITEL can thereby rely on the different macro LPs that the tutor has added into the Cognitive Model and also the micro LPs that are available due to their definition in the Pedagogical Ontology and the various SLOM metadata. Furthermore, habit-based custom LPs for learners are also a possibility, due to the LP-Selector module, which implements a method to measure when and how this is reasonable.

[9] See deliverable 4.1 for a detailed introduction into SLOM, which is being specified in parallel to this deliverable.

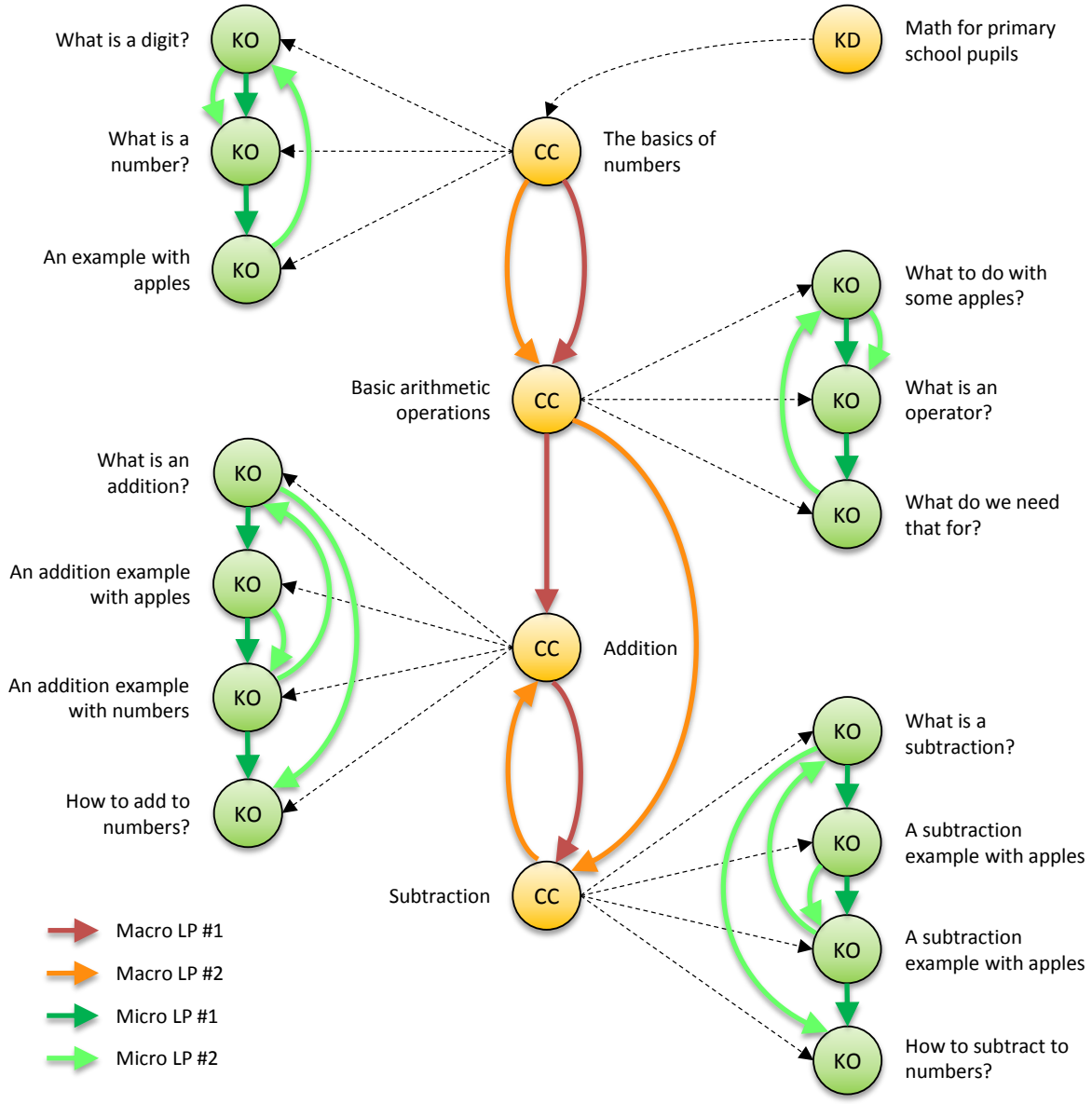


Figure 5: Exemplary diagram showing the coherences in the domain input, including different LPs

## 4.4 Preprocessed Input

The term preprocessed input subsumes all the knowledge that INTUITEL collects over time regarding each particular learner. This mainly is the general metadata that is used to create the recommendations and messages that the learners receive.

One of the fundamental information that is included here is each learner's session history, which, amongst other, includes:

- Session start time
- Session end time

- LOs visited
- Duration spent on a particular LO
- Order in which the LOs have been visited
- Selected micro/macro LP of that session
- KO and time when a LP has been switched

These are all relevant data items that INTUITEL needs to come to conclusions regarding the learning habits of the individual learners. This can then be used to create personalized statements about learning, which are the basis of the personalization of LO recommendations and messages. If, for instance, the session duration exceeds a certain amount of time and the learner needs considerably more time for a LO than it is estimated, a TUG-message could be created that recommends to make a short break.

The preprocessed input can thus be interpreted as the user model INTUITEL creates to describe the learner. It contains all the relevant metadata about the learners that is needed to better understand how they actually learn. This is also not necessarily limited to the session information, but also includes general information about the learner (e.g. gender, age, name, salutation, etc.), requested or deduced information (e.g. the LP the learner has selected or which INTUITEL concluded would be optimal for the learner), the individual learning history (i.e. which LOs has been accessed and their completion status) and might furthermore be extended to also contain previous deductions (e.g. characterizations of the learner as chaotic or methodological) that are relatively stable to decrease the computational load.

What data will exactly be part of this category is not yet fully specified. This is not only a question of what is available, but what is relevant for the recommendation creation process. The data items that are part of this input class are thus not fixed, but are open for future additions. INTUITEL will firstly come to a listing of relevant points when the first draft of the factors with recommendation relevance (i.e. Didactic and Rating Factors – cf. sections 6.4 and 6.5) has been specified.

# 5 Learner Position

This chapter explains the intentions behind the Multidimensional Cognitive Space (MCS) and the associated Cognitive Content Space (CCS). Both are newly introduced concepts that the INTUITEL project uses to formalize the task of finding the position of the learner in the eLearning content. It has been designed to translate the basic educational conditions to a mathematical level, which is much better applicable on the domain of computer science. The presented geometrical representation of Learning Pathways and learning content in form of a hypercube facilitates a better understanding of the technical realization of this task.

## 5.1 Determination of the Next KO

In the following it is outlined how to determine initial recommendations for a "next" KO to be processed by the learner, which are then modified by the Didactic Factors. Note, that throughout this chapter we are not dealing with Micro Learning Pathways or LOs that are considered alternatives, but only with Macro Learning Pathways and singular LP nodes. Consequently, for the purpose of this consideration, each KO may be considered as fundamentally different from all the others because it leads to a different knowledge gain.

The basic paradigm of outcome-oriented constructivist learning is that it should lead to a certain learning goal. This goal might be quite complex, e.g. might require that

- a certain set of concepts is learned with a required precision
- a certain set of temporal, causal or logical connections between these concepts is learned
- a certain set of methods and algorithms is learned which enables the learner to process new situations

Mathematically, this complex goal (which in fact is an ontology) may be expressed as a target position in an abstract space spanned by the concepts. We term this the Multidimensional Cognitive Space MCS. Learning therefore may be considered a movement in this abstract space – and learning should bring the learner closer to the target position.

In modern human-centred teaching processes, teachers determine the *current position* of a learner by assessing the learner's current knowledge and standing in regard to the topic and on a "meta"-level (e.g. how the learner learns). This happens either subconsciously by sensing that there are problems, delays or disturbances while the lessons are taking place, by evaluating the assignments of the learners (e.g. homework) or by having a direct conversations with them (e.g. questions during the lessons). These observations and feedback enables the teacher to come to conclusions regarding the learner's state of knowledge, habits and characteristics to ultimately support the learner in the learning process.

INTUITEL will provide means to carry out this tutorial guidance process also in technology enhanced learning, where no human teacher in traditional form is present and the learning so far may be seen

as a self-directed process. A learning object recommendation (LORE), as will ultimately be achieved by the INTUITEL system, therefore is a computer generated hint towards the self-directed learner on how the learner may come closer to the target position. Note that this recommendation is not entirely a global recommendation ("direct way to the target"), but may be locally deviant from the direct way to the target.

Consider, for example, the possibility of a LORE indicating that the learner should repeat a certain Learning Object: In this case it might correspond to a movement of the learner in the MCS which for some time takes the learner further away from the final learning target position.

The INTUITEL system therefore is handling the following tasks:

- Determination of the learner position in MCS
- Determination of a set of didactical factors influencing the learning process
- Application of the personal Learning Pathway on the course material
- Taking into account these Didactical Factors and the Learning Pathways, determination of the next step the learner has to take in order to ultimately come closer to the target position

The principles of the MCS will be reconsidered below, for now let us consider the Didactical Factors. Some of them are:

- Cognitive speed = learning speed in the context of environment, personal attitude and current learner position – is this a fast, a medium or a slow learner?
- Learning success – is this a learner with superior, good, average or inferior results?
- Learning discipline – does this learner follow suggestions easily, does the learner adhere to ascribed learning pathways?
- ... more to come

For the moment, the INTUITEL consortium relies on a rather simple "scale" for each of these factors, e.g. "learning speed" will not be measured in objective terms but in simple concepts of fast – medium – slow. This means, that a crucial module of the LPM will be responsible to translate raw input data (from the LPM) into these Didactical Factors.

Consider, for example the Didactical Factor of "learning speed". For each learner, the LMS will deliver the actual learning time for each KO to the LPM. Furthermore, the LPM will read the SLOM metadata accompanying the learning content – and will know the target learning time entered into the SLOM by the cognitive engineer (henceforth called "estimated time"). The corresponding transformation rule (or dimensional reduction rule) for the LPM then reads, for example:

> "If the learner needs more time than the estimated time to process a KO in at least 70% of all KOs, the learner will be assigned the value "slow" on the dimension of learning speed. If the learner needs less time than the estimated time to process a KO in at least 70% of the KOs, the learner will be assigned the value "fast" on the dimension of

learning speed. In any other case, the learner will be assigned the learning speed value "medium" (70-70-rule)."

However, suppose that the learner has already achieved a high level of knowledge, easy questions should then be answered faster. Therefore, one could think to shift from a 70-70-rule for learner speed determination to a 50-85-rule in this case. Or it may turn out that the learner is working in a high noise environment providing lots of distraction – in this case one could revert to an 85-50-rule.

## 5.2 Cognitive Content Space

The Cognitive Content Space (CCS) is spanned by the Cognitive Model and the Semantic Learning Object Model (SLOM). The Cognitive Model contains the pedagogical structure of a course; it is a concretization of the pedagogical ontology for a given domain of knowledge[10]. For details see deliverable D 3.1 (Overall System Design). The SLOM contains metadata referring to the actual content, cf. deliverable 4.1 (SLOM). Both of these might contribute to the desired learning goal.

Default learning goal will be that each of the Knowledge Objects contained in the SLOM are processed[11] (and therefore, by assumption, learned) by the learner – which would then amount to a definition of the learning goal by the SLOM metadata accompanying the learning content. This will be the initial establishment of the LPM.

For later development it is also possible, that a more complex learning goal is defined in the Cognitive Model. For example, the cognitive engineer could specify, that in his Cognitive Model the desired learning goal puts more emphasis on practical knowledge than on theoretical knowledge. In such a case, the Cognitive Model would contain a transformation rule to determine the cognitive position from the raw input data.

## 5.3 Multidimensional Cognitive Space

The cognitive position of a learner is determined by the amount the learner has *learned* from each of the KOs. Hence, the position for a course consisting of $N$ KO is determined by an $N$ dimensional vector $P = \{x_i\}, i = 1, \dots N$ with $x_i \in [0,1]$. Each value $x_i$ can be determined as follows:

- In a simple LMS one only knows that a KO has been processed. In this case we will follow the *strong optimistic learner assumption*: Processing means learning, consequently $x_i$ jumps from 0 to 1 when a KO has been accessed.
- A more advanced LMS will tell INTUITEL which part of a KO has been processed by the learner. In this case we will follow the *weak optimistic learner assumption*: A partial processing of a KO means that the learner has *learned* the same percentage of this KO.

[10] In the language of computer scientists: a concrete instantiation of the abstract Pedagogical Ontology.

[11] For the sake of simplicity excluding here and afterwards in this chapter the KOs that are real alternatives to already learned ones. E.g.: A textual transcript of a video does not have to be processed, if the respective video has already been seen.

- A very advanced LMS will tell INTUITEL the result of a measurement, like e.g. the percentage of points reached in a concluding test of this KO.

Let us now consider how a certain learning goal may be achieved as a sequence of cognitive positions.

- The default learning goal states, that each KO has to be *learned*. The learner starts at position $P_s$ = (0,0,....0), his target position is $P_f$ = (1,1,...1), *target position* in the MCS therefore is a value of 100% = 1.0 for each component of this vector.
- Should there be a more complex learning goal (say, a target learning profile), the Cognitive Model defines a transformation, which for simplicity may be seen as a linear transformation (a matrix) reducing the dimensionality of the MCS from N to M<N. Therefore, without loss of generality, even in this case the same INTUITEL concepts and algorithms may be used as for the default learning goal.

If we assume a learner who completes each KO before moving on, the "movement" of this learner in MCS would always be along the edges of the N-dimensional knowledge hypercube: The learner has completed KOs 1,2,3,4,...k-1, is currently working himself through the $k^{th}$ KO and still has to consider himself with KOs k+1,,k+2,....N. Hence, his cognitive position would be

$$L_{kx} = (1.0,1.0,1.0 \ldots., x, 0,0, \ldots. 0) \text{ with } x \in [0,1]$$

Here, without loss of generality, we have ordered the KOs in exactly the sequence as it is processed by the Learner (henceforth called User Learning Path (ULP)).

However, we might also assume a learner who is much less disciplined, and therefore does not complete any KO before moving on. Obviously, the sequence of cognitive positions in the MCS would be a curve inside (and not along the edges, but possibly across bounding surfaces) of the hypercube. In the extreme example, this learner would switch back and forth between the KOs in the model and finish each KO to the same degree. While this learner might nevertheless reach the final learning goal, his actual learning curve is a line close to the hyper-diagonal of the MCS. In order to achieve his final goal however, this learner would have to perform a very large number of switches between KOs, in order not to emphasize a particular KO.

For now let us assume that we have a disciplined learner who always moves along the edges of the knowledge hypercube. This amounts to a self-chosen sequence of KOs, each of them is processed (by assumption, *learned*) completely before moving on. After completing the k-th KO, the cognitive position would be

$$L_{k1} = (1.0,1.0,1.0 \ldots., 1.0, 0,0, \ldots. 0)$$

Obviously, this learner then has N-k possible choices for his next KO[12], and the *direct* cognitive distance to the target position is $\sqrt{N-k}$. Each of those possible steps would reduce the direct

12 Only if repetitions are excluded.

cognitive distance by the same amount, e.g. $\sqrt{N-k}-\sqrt{N-k-1}$. Hence, none of the remaining KOs would be preferable from the viewpoint of reducing *direct* cognitive distance to the target state.

The Cognitive Model however should define certain Learning Pathways $LP_1$,...$LP_S$. Since we reserved the "standard" numbering for the ULP, we therefore have to consider each of these predefined Learning Pathways an ordered permutation of the numbers 1..N. Since the current cognitive position after k learning steps may be arbitrarily close to Learning Pathway l – but after a number of t steps – we may not use simple path deformation rules to compare the actual current cognitive position to one that could be reached by this Learning Pathway. Therefore the LPM will implement the following algorithm:

1. Determine the Learning Pathway $LP_l$ from the Cognitive Model which runs closest (in distance d) to the current cognitive position L , and the closest corner point V (vertex) of $P_l$
2. Check each of the open choices N-k, if it would reduce the distance to V from d to d'<d and assign to it the priority d-d'.
3. Check on the Learning Pathway $P_l$ which the next KO would be after V, we assume that this would be KO no. v. If v is among the open choices N-k, add to the priority assignment of v the value +1 and terminate the loop. If not, choose the successor of v in $P_l$ and check if it is among the open choice, add to its priority the value of ½ etc. continued if either this loop terminates or if the end of $P_l$ is reached.
4. Pass the information to the next decision stage.

L is close to $P_l$ , and the next object recommended when following $P_l$ is followed by the one of the N-k open choices which has the highest cumulated priority. If there is more than one next choice with the same priority, take a random selection among those.

This algorithm ensures that a recommendation to the learner will, if the learner follows the recommendation, propagate through the MCS roughly in parallel to a certain Learning Pathway and at the same time towards this particular Learning Pathway.

It is possible (and will be checked in the concrete implementation of the LPM) to relax this and present to the next higher decision stage a selection of alternative Learning Pathways and the corresponding suggestion for the next KO.

## 5.4 Examples for Learner Positions

### 5.4.1 Example 1: Two Knowledge Objects

We now consider a system consisting of two different Knowledge Objects A and B. For reasons of formal manipulation we bring them into an ordered sequence, abstracting them as $K_i,\ i=1,2$ such that $K_1=A,\ K_2=B$. The Multidimensional Cognitive Space then is a two-dimensional square, as depicted in Figure 6.

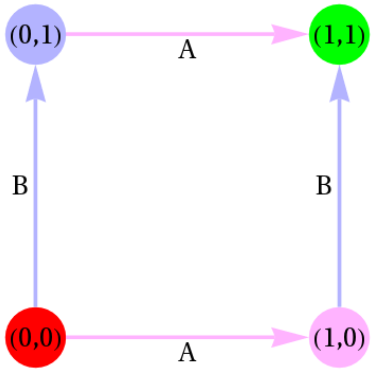


Figure 6: Two dimensional cognitive space (2D plane)

The cognitive position of a learner is determined by the amount the learner has *learned* from each of these two KO. Hence, the position is determined by a two dimensional vector $P = \{x_i\},\ i = 1,2$ with $x_i \in [0,1]$. The learner starts at position $P_s$ = (0,0), his target position is $P_f$ = (1,1).

Two distinct Learning Pathways exist:

$LP_1$ = (A, B) = ($K_1$, $K_2$)

$LP_2$ = (B, A) = ($K_2$, $K_1$)

Irrespective of which of these sequences is chosen by the learner, his cognitive position after processing the first KO will always be on the graph determined by one of these LPs, consequently one of these LPs has distance 0. Therefore a single possibility remains for a LORE-message: „Please process the other KO“, or in greater detail:

- If the learner has chosen to process A first, then his cognitive position after processing A will be on $LP_1$ and the LORE will be „Please process B“.
- If the learner has chosen to process B first, then his cognitive position after processing B will be on $LP_2$ and the LORE will be „Please process A“.

Degeneracy arises, if the learner has chosen to process one of the KOs, but has achieved zero result. In this, his cognitive position still remains in the origin, and therefore the complete freedom to choose a KO. However, for practical reasons we can stick to the above algorithm.

It may be that only one of the two possible distinct LPs has been defined in the Cognitive Model. Such may be the case if B is advanced knowledge, and A is basic knowledge, therefore only $LP_1$ = (A,B) has been found pedagogically useful. Obviously, also in this case the recommendation would be correct.

After the second learning step, the cognitive position of the learner will be the target position $P_f$ = (1,1), if at least the weak learner assumption holds. If it does not hold, the cognitive position of the learner (as determined by measuring his results) may be anywhere within the MCS.

### 5.4.2 Example 2: Three Knowledge Objects

We now consider a system consisting of three different Knowledge Objects A, B and C. For reasons of formal manipulation we bring them into an ordered sequence, abstracting them as $K_i,\ i = 1..3$ such that $K_1 = A,\ K_2 = B,\ K_3 = C$. The Multidimensional Cognitive Space then is a three dimensional cube, as depicted in Figure 7.

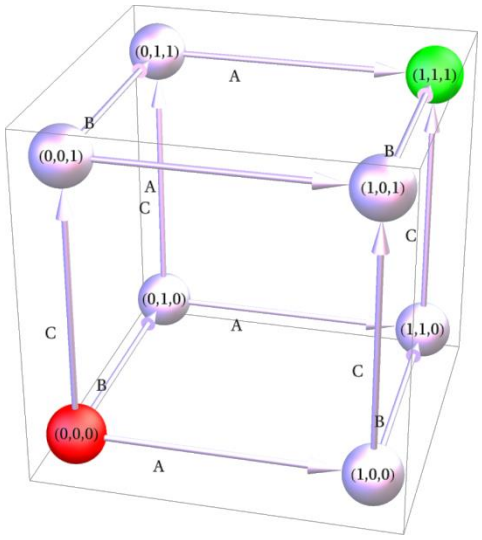


Figure 7: Three dimensional cognitive space (3D cube)

The cognitive position of a learner is determined by the amount the learner has *learned* from each of these three KOs. Hence, the position is determined by a three dimensional vector $P = \{x_i\},\ i = 1..3$ with $x_i \in [0,1]$. The learner starts at position $P_s$ = (0,0,0), his target position is $P_f$ = (1,1,1).

Six distinct Learning Pathways exist:

LP1 = (A,B,C) = ($K_1$ ,$K_2$ ,$K_3$)

LP2 = (B,A,C) = ($K_2$ ,$K_1$ ,$K_3$)

LP3 = (B,C,A) = ($K_2$,$K_3$ ,$K_1$)

LP4 = (C,B,A) = ($K_3$,$K_2$ ,$K_1$)

LP5 = (C,A,B) = ($K_3$ ,$K_1$ ,$K_2$)

LP6 = (A,C,B) = ($K_1$ ,$K_3$ ,$K_2$)

It is important to realize in this simple model that the individual learning steps (which consist, in our picture, of processing and thereby *learning* a KO) are not commutative concerning the cognitive position of the learner. For example, LP1 and LP2 differ only in the order of KO A and B – but nevertheless lead to completely opposite corners of the MCS cube. The cognitive position therefore contains some information about the order in which KO have been accessed – but not monotonously so, because all six LPs will end up in the target position $P_f$.

Both the strong and the weak learner assumption (cf. section 5.2) will lead to this final position: If a learner has accessed and completed all KOs, the learner will have learned everything that is available. Of course, the nemesis of "fact" now raises its head over the beautiful corpse of "theory": Pedagogically it is obvious that not all LPs are desirable. Obviously therefore the LPs specified in the Cognitive Model for a given domain must be a subset of the available LPs. Another conclusion drawn from this is that ultimately the strong and weak learner assumption will have to be replaced by measurements of learning results.

### 5.4.3 Example 3: Four Knowledge Objects

We now consider a system consisting of four different knowledge objects A, B, C and D. For reasons of formal manipulation we bring them into an ordered sequence, abstracting them as $K_i,\ i = 1..4$ such that $K_1 = A,\ K_2 = B,\ K_3 = C,\ K_4 = D$ . The Multidimensional Cognitive Space then is a four dimensional hypercube or tesseract. A three dimensional projection of this hypercube is depicted in Figure 8.

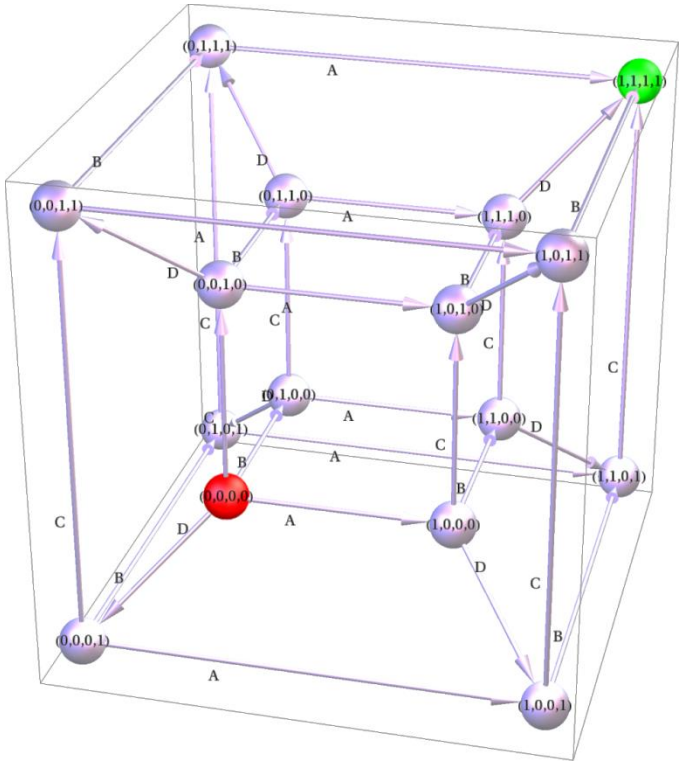


Figure 8: Four dimensional cognitive space (4D hypercube) for four knowledge objects

The cognitive position of a learner is determined by the amount the learner has *learned* from each of these four KOs. Hence, the position is determined by a four dimensional vector $P = \{x_i\},\ i = 1..4$ with $x_i \in [0,1]$. The learner starts at position $P_s$ = (0,0,0,0), lower left of the „inner" cube of the above projection, his target position is $P_f$ = (1,1,1,1), upper right of the „outer" cube of the above projection.

In principle 4! = 24 distinct Learning Pathways exist – and therefore, since the learner is free to choose, may be transgressed by the learner. Not all of these are didactically meaningful, hence let us assume that the KO are grouped:

- The two pairs A,B resp. C,D each constitute a thematic group.
- The pairs A,C resp. B,D each constitute a chronological group.

Such a grouping could arise, if

- A describes the contribution of Comenius to educational theory
- B describes the contribution of Habermas to educational theory
- C describes the contribution of Comenius to communicative didactics
- D describes the contribution of Habermas to communicative didactics

Consequently, two Learning pathways are defined in the Cognitive Model:

$LP_h = (A,B,C,D) = (K_1, K_2, K_3, K_4)$ as *hierarchical* LP

$LP_c = (A,C,B,D) = (K_1, K_3, K_2, K_4)$ as *chronological* LP

These two LP's differ only in one permutation of $K_2$ ,$K_3$ . They are depicted by the thick arrows in the projection in Figure 9.

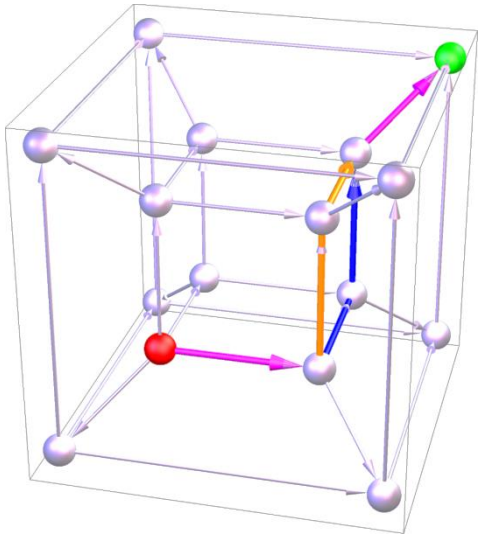

Figure 9: The two didactically meaningful learning pathways (hierarchical and chronological), shown as thick arrows

Let us now follow a learner through this system. First of all let us assume a disciplined learner: The learner follows the initial advice and processes KO A, which puts the learner into cognitive position (1,0,0,0). Obviously, the learner is still following both LPs defined in the Cognitive Model. If no other factor (like a pre-chosen LP) will lead the INTUITEL Engine to another conclusion, the learner will be offered two choices by the INTUITEL system:

> *„If you want to follow the hierarchical LP, please process B. If you want to follow the chronological LP, please process C".*

Now, conversely we assume an undisciplined learner who, instead of following the initial advice to process A, has chosen to process D instead. This would put the learner after the first learning step into cognitive position (0,0,0,1). Using the algorithm described in section 5.3, the LPM would determine that the learner is close to position (0,0,0,0) which is on both well-defined LP. The learner would then get the advice to process A as the next KO, bringing the learner (if the advice is followed) into position (1,0,0,1). There, the learner is close to position (1,0,0,0), which is on both LP. Therefore, with equal priority (unless other factors come into play) the learner will be advised to process B or C. Suppose B is followed, this would put the learner into position (1,1,0,1) – where the learner would finally get the advice to process C leading to the learning goal. The actual (e.g. „realized“) LP then is (D,A,B,C) – where at least three KO are in the sequence which has been termed adequate by the cognitive engineer.

However, our learner might be so undisciplined that instead of following the first advice the learner takes a complete different route – processing C after having done the first step of processing D. This would take the learner into position (0,0,1,1) – and the closest of the well-defined LP's is the chronological LP with position (1,0,1,0). The learner would therefore get the advice „You are close to the chronological LP, please process B“. If the learner follows this advice, the learner is led into position (0,1,1,1) – where then the learner would get the final advice to process A and therefore the overall realized LP would be (D,C,B,A). In this realization, only the two KO C,B are in the chronological sequence described in the Cognitive Model.

We might also encounter a totally undisciplined learner, who also does not follow the second advice, e.g. processing first D, then C and then A. This would again bring the learner close to the chronological LP, and the learner would receive the final advice to process B. The actual (e.g. „realized“) LP then is (D,C,A,B) – whereas in the previous scenario, only the two KO A,B are in the chronological sequence described by the Cognitive Model.

# 6 Back End Concept

In order to understand what the specific tasks of the LPM are in the recommendation creation process, it is advisable to first understand how the Back End works as a whole. Due to the usage of OWL-reasoning, the approach of how recommendations are created is different to how a common software solution would approach that task. This is accountable to the capabilities of OWL-reasoners, which have a completely different focus than more common programming frameworks like .Net and Java.

Thus, before chapter 7 explains the LPM concept, this chapter outlines the overall Back End concept. Although these issues are rather part of the INTUITEL Engine and the LM ontology, their conception and introduction here is reasonable in order to specify a coherent system. It shall therefore be noted that the details of this approach are not fully defined yet. Due to the wide-ranging linkage with the other Back End development tasks (i.e. especially the INTUITEL Engine), a lot of work will be put into the refinement of these descriptions to create a fully sound and complete system.

## 6.1 Set-Based Rating of Learning Objects

One of the main features of OWL-reasoners is the identification of elements in a set. This is a very useful functionality for INTUITEL, because creating a KO recommendation can be interpreted as the task of finding the elements of the set that contains the optimal KOs for the learner. Thus, the fundamental question of the Back End is: What are the defining criteria of this set and how to evaluate them?

Building on the previous introductions on what input is available for the LPM (cf. section 4) and on their more detailed descriptions in D1.1 and D2.1, it can be registered that there basically are three influential factors concerning the suitability of a KO:

- the learner's macro Learning Pathway
- the learner's micro Learning Pathway
- the situational dependent information of the learner

The first two points are relatively unproblematic, because this basically only requires data that is already available in the present system. This is at first the data which specifies the Learning Objects that are part of the course, i.e. the SLOM metadata (including the CM). Secondly, the history of the learner is needed to filter those LOs that are not available for the recommendation (e.g. because they are already finished). Thirdly, the learner's personal Learning Pathway is required, to find the LOs that are next in the sequence.

Given that this information is handed over to the Engine in a suitable way, the LOs that are relevant in context of the learner's LP can be identified. This is possible, because the present problem can be described on basis of sets, allowing the Engine to iteratively reduce the number of LOs. In order to understand this procedure, it is advisable to visualize this step-by-step.

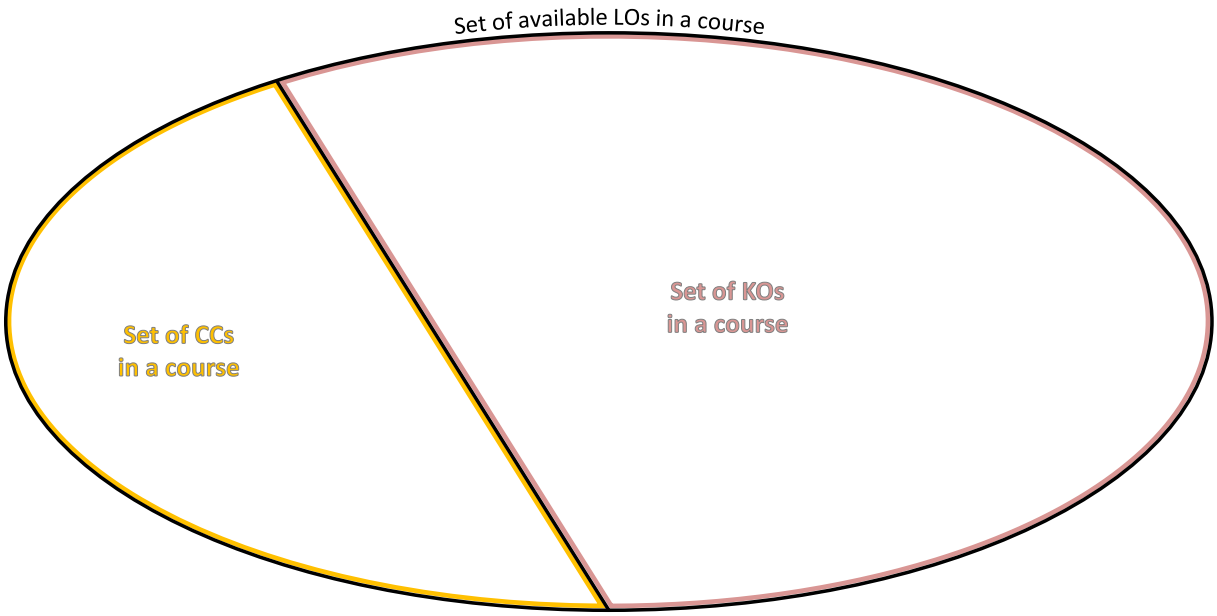


Figure 10: Segmentation of LOs in a course into the respective sets of CCs and KOs

As the first step, it should be clear that a LO is (in INTUITEL) an umbrella term that combines Concept Containers and Knowledge Objects[13]. Each LO is thus either a CC or a KO. So, when starting with a set that contains all LOs of a certain course, it is possible to segment it into the set of CCs of that course and into the set of KOs of that course (cf. Figure 10).

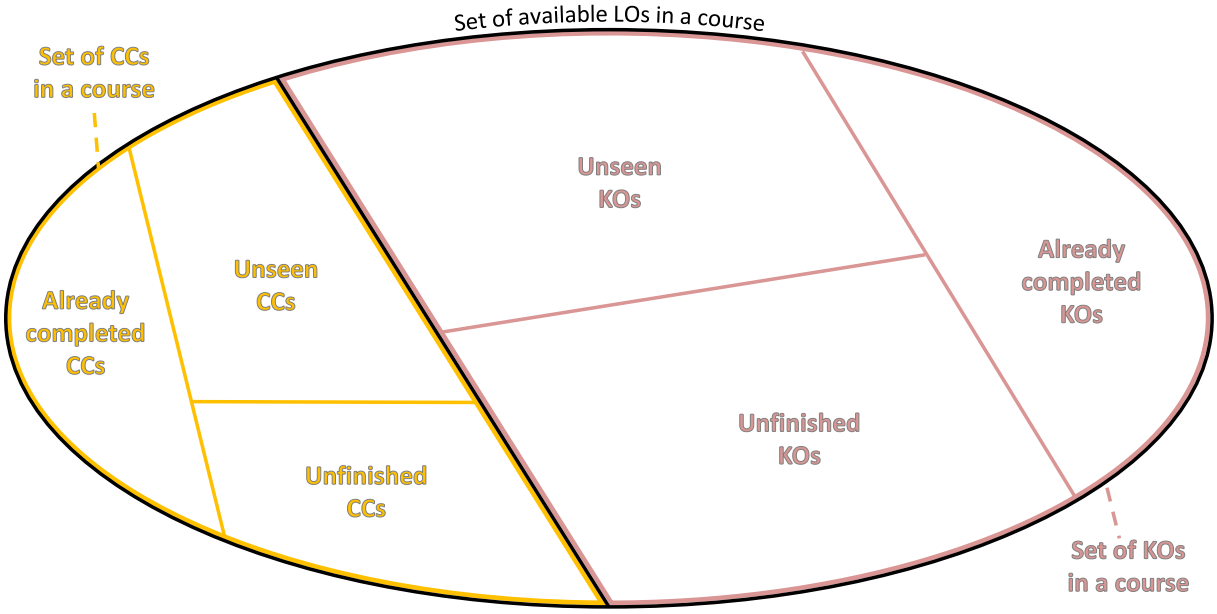


Figure 11: Segmentation of the LOs in a course regarding their completion status

These two sets can further be segmented into sets that differentiate LOs depending on whether they are rated as unseen, unfinished or already completed (cf. Figure 11). The criteria to categorize LOs

[13] For the sake of simplicity, Knowledge Domains (KDs), which are by definition also LOs, are excluded here.

are thereby only a matter of definition. Depending on which information is used and what exactly should be considered in the recommendation creation process, the definitions can be adjusted. One of the tasks of the INTUITEL Back End is to decide which of the available criteria is reasonable. There actually are multiple ways to define the individual sets:

- Criteria to rate the completion state of KOs:
  - Access status, i.e. has the learner already visited the KO or not?
  - Duration spend on the KO, i.e. the ratio between actual and estimated learning time.
- Criteria to rate the completion state of CCs:
  - Completion state of the contained micro LP.
  - Percentage of completed KOs that are attached to that CC.
- Criteria to rate a KO as unseen:
  - Access status, i.e. rate a KO as unseen if it was not yet accessed.
  - Duration, i.e. rate a KO as unseen if the learner has not spent more than a certain amount of time on that KO.
  - Seen percentage, i.e. rate a KO as unseen if less than a certain percentage of is content has been seen.
- Criteria to rate a CC as unseen:
  - Percentage of unseen KOs attached to the CC.
  - Percentage of unseen KOs attached to the CC that are part of the current micro LP.
- Criteria to rate a KO as unfinished:
  - Duration spend on the KO.
  - Seen percentage of the KO.
  - Grade achieved in the KO.
- Criteria to rate a CC as unfinished:
  - Completion states of all attached KOs.
  - Completion states of all attached KOs that are part of the active micro LP.

Please note that this list is not necessarily complete and it might also be useful to apply a combination of these factors. It might even be reasonable or advantageous to omit a definition and use the remaining set as the missing one (i.e. define what is complete and what is unfinished and use the remaining set as the unseen). How exactly this will be carried out in a later productive system is irrelevant for the basic functionality. As long as the definition is sound, it only remains a question of what is the most reasonable one to find the most fitting set.

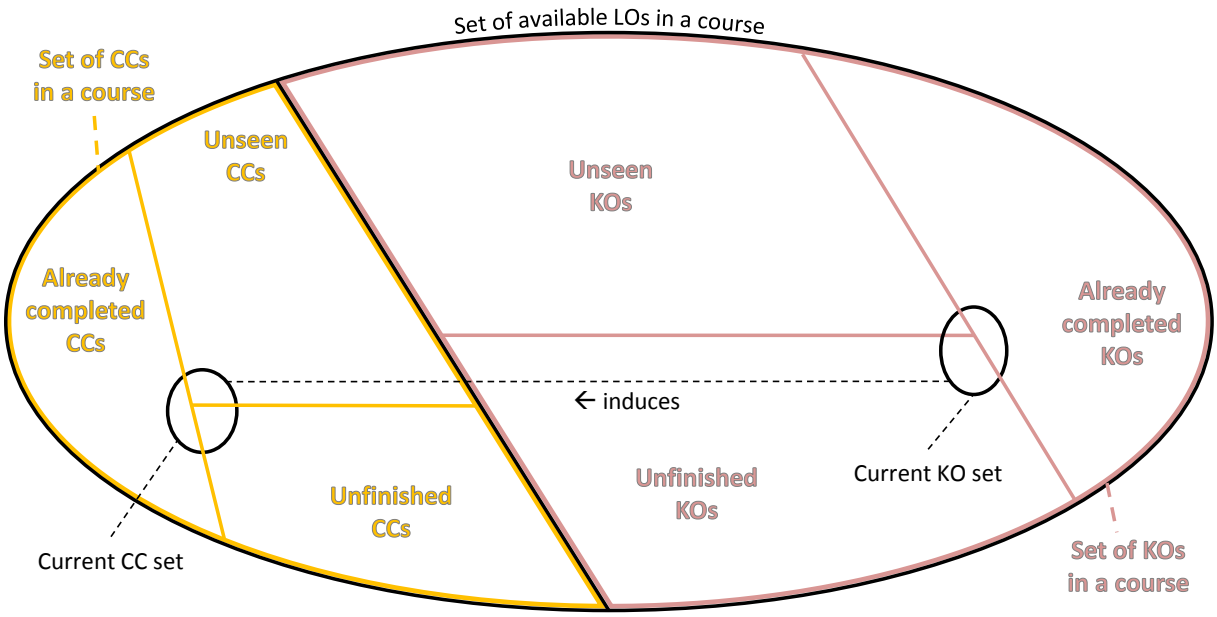


Figure 12: Sets of current CCs and KOs in the set of available LOs

Independent of these sets is it possible to subdivide the set of available LOs of a course regarding their distance to the current Learning Pathway positions. For this, firstly the set of currently active KOs is identified (cf. Figure 12), which is either empty or contains exactly one item. It is empty if the learner has never worked on the course (i.e. course learning history is empty) and has not yet accessed a KO in the current session. If the learner has already accessed a KO in the LMS, the set contains exactly this KO (i.e. the last one), which is rated as either unfinished or already complete.

In order to continue with the process, there must be a current KO. The Back End does thus have to decide which KO should be selected if the set is empty. If the macro and micro LP is already available (e.g. via a default value), the KO is selected that is first in the sequence. If they are not yet specified, the Back End can either decide itself (e.g. by choosing the most popular LPs or the LPs that are most fitting to the learner's habits) or ask the learner via TUG.

Based on that, it is possible to identify the set of current CCs, which contains all CCs that reference on the currently active KO. The set thus contains either one CC or multiple ones, because a KO can be part of more than one CC.

The set of CCs that are next when following the macro LP is a logical consequence of the previous set. When it is known which CC(s) is/are currently active, the next CCs are those that are subsequent as described in the respective LP definition. What should be considered as "next" is thereby a question of what is reasonable. It could, for instance, just include those CCs that are the direct followers of the current CC(s), or also the ones that are some steps afterwards. This does not necessarily have to be a fixed definition, since it might be reasonable to add more items, depending on the number current CCs. The more CCs are selected, the more KOs are part of the recommendation creation process.

With these sets, the Back End is able to identify the CCs that have recommendation relevance for the respective learner (cf. Figure 13). This firstly is the intersection of the set of current CCs with the

union of unfinished and unseen CCs, because the already completed CCs are irrelevant for further recommendations. If there are no currently active and unfinished CCs, the set must be extended to also include the intersection of the set of macro-LP-next CCs with the union of the unseen and unfinished CCs. This could also generally be the case to increase the recommendation basis. It would likewise be possible to investigate those two sets individually to come to a separated result in order to calculated different recommendation values.

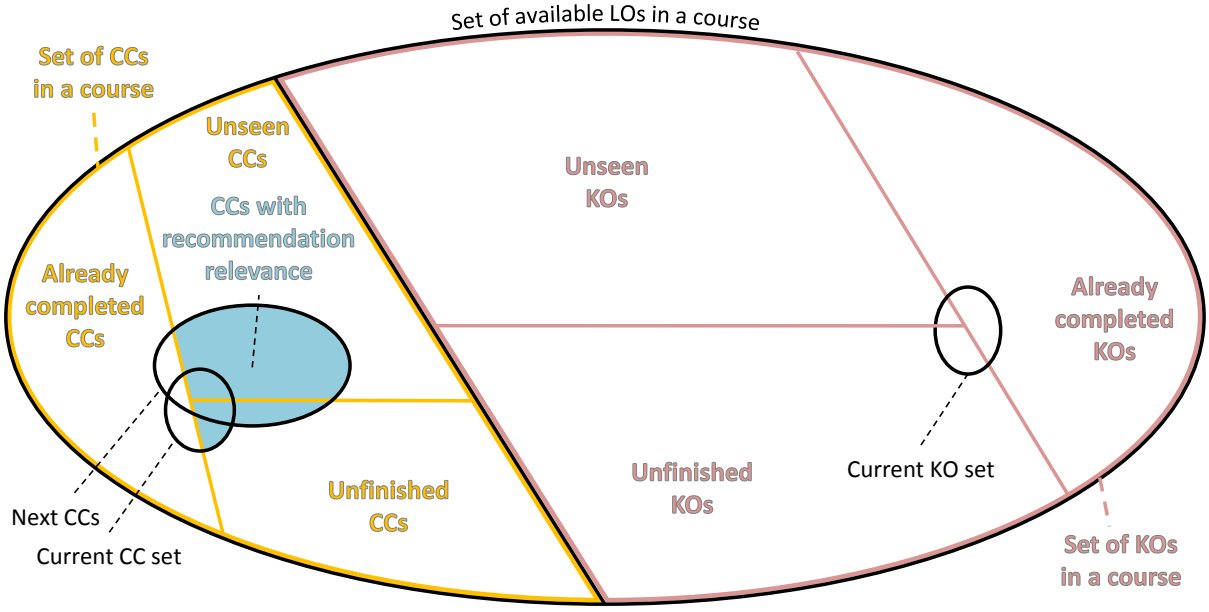


Figure 13: Set of CCs with recommendation relevance

If the set of CCs with recommendation relevance is empty, the course is either complete or there are no more CCs available in that particular macro LP. In the first case, a recommendation cannot be created. In the second case, a new macro LP needs to be selected or the process is finished too.

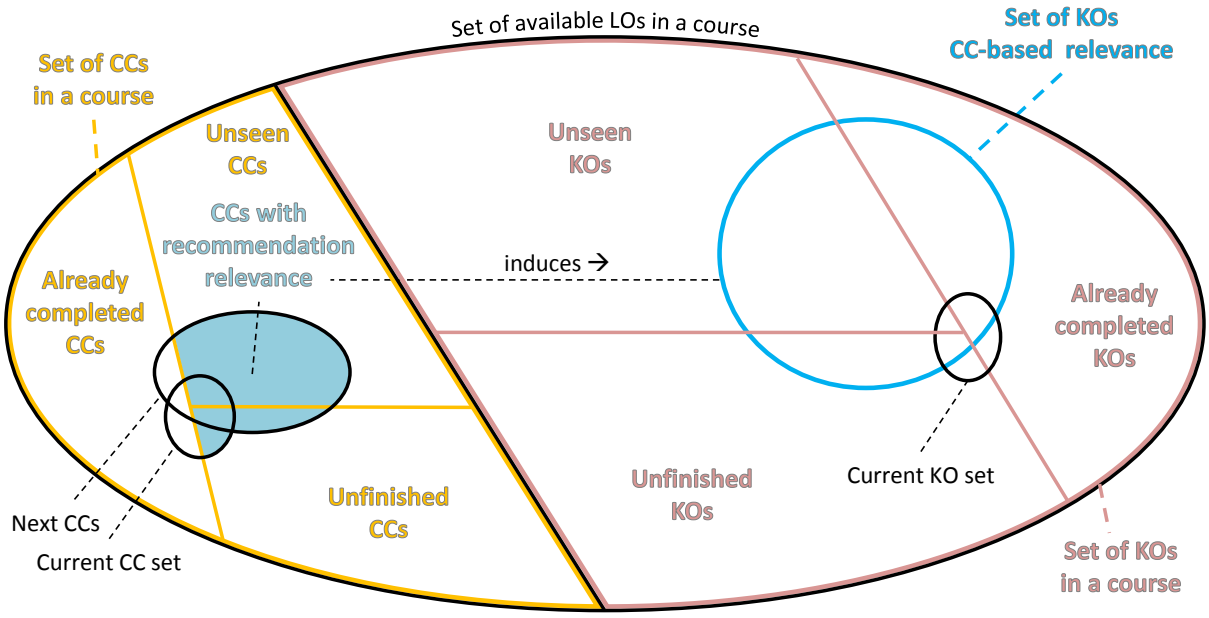


Figure 14: Set of KOs with CC-based relevance

If at least one recommendation relevant Concept Container has been identified, it is subsequently possible to create a list of KOs with that have a CC-based relevance. This is the set of KOs that are attached to this/these particular CC(s). As with its CC counterpart, this set could contain KOs that are rated as completed, unfinished or unseen. It might also include the currently active KO.

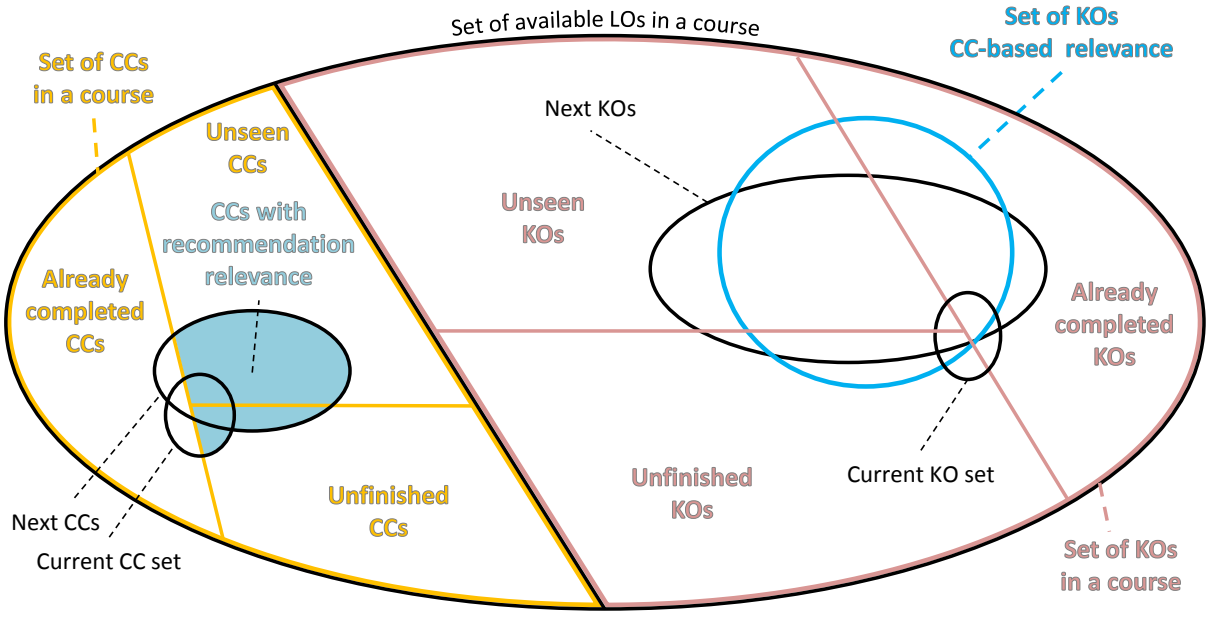


Figure 15: Set of KOs that are next regarding the micro LP

This is already a big reduction of the available LOs, but it can be reduced even further, when also including the micro LP information (cf. Figure 15). This is the set that contains all KOs that are “next” as seen from the currently active KO and those KOs that are “next” regarding the “next” CCs. As

previously, the term “next” is a matter of definition, which is based on the question what is reasonable to include here.

As the last step in the procedure, to identify those KOs that are recommendation relevant, the Back End has to select those KOs that fulfil a list of requirements:

- KO is next or currently active
- KO is attached to a CC that is next or currently active
- KO must be unfinished or unseen (i.e. not already completed)

Expressed on a set-basis, this is the intersection of the set of currently active KOs, with the set of KOs with CC-based relevance and the union of unfinished and unseen KOs. Depending on the later implementation this could, should or must be extended with the intersection of the “next” KOs with the set of KOs with CC-based relevance and the union of the unseen and unfinished KOs. For an easier understanding, the diagram in Figure 16 clarifies these coherences graphically.

If this respective set is empty, the same method as already used for an empty set of CCs with recommendation relevance is applied. Either has the micro and/or macro LP to be changed to include other LOs, or the process is finished, since there are no more KOs available to choose from.

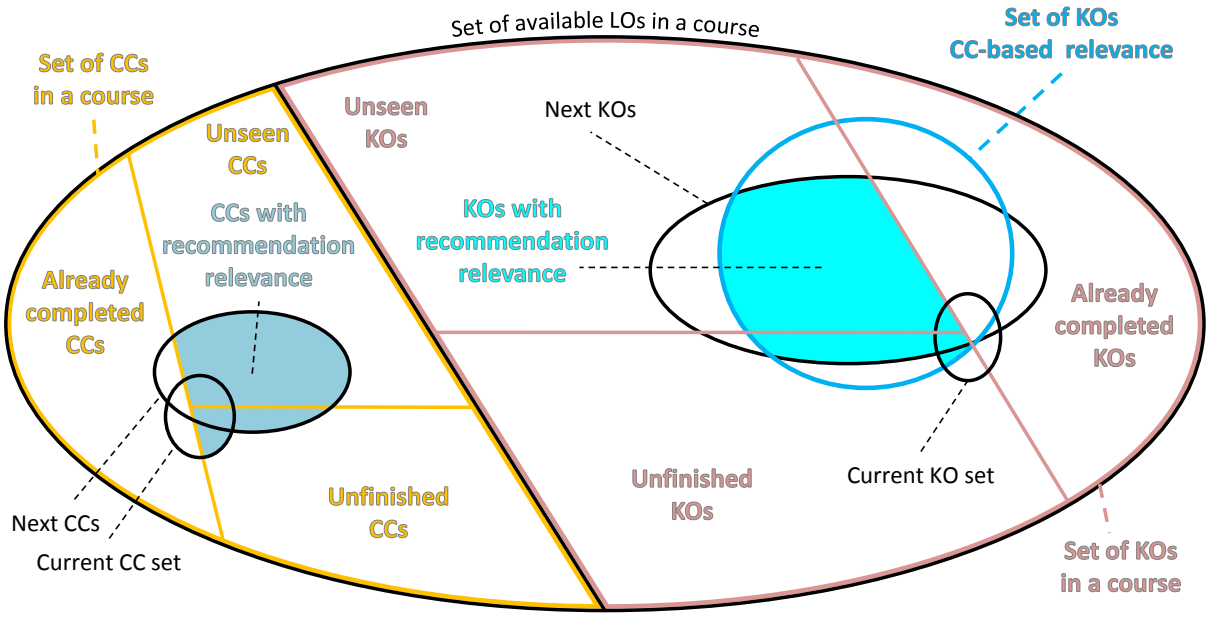


Figure 16: Set of KOs with recommendation relevance

When now coming back to the initial statement that there are tree influential factors for the recommendation (macro LP, micro LP and situational dependent information), this so far only includes the first two aspects. In order to personalize the recommendation even further[14], the information that INTUITEL collects about the learner must be included in this procedure. To do this,

[14] The LPs are selected in a per person basis. So, the recommendations are actually already personalized, but they can be adapted on a much higher level.

the Back End specifies a method to include this information in a suitable way, which is implemented via the so called Didactic Factors and Rating Factors (cf. section 6.4 and 6.5 for a detailed introduction).

A Didactic Factor is a nominal (i.e. non-numeric) value that consists of a number of data items that are available in INTUITEL. It is thereby possible to use every piece of information of the INTUITEL data basis and combine them freely to create an assertion about the learning situation, habits or preferences of a particular learner. This also includes that learners can be contrasted to their (anonymized) course mates to also include the preferences of them. The Back End is thus able to include diverse, meaningful, personalized information in the recommendation creation process.

On the next level, these Didactic Factors are combined with properties of Knowledge Objects to state their individual suitability for the learner. The Back End can thus apply rules that, for instance, state that the combination of a slow internet connection and a large video file is bad and thus lower the recommendation value of that particular KO.

The procedure of connecting this information with the learning content (i.e. the available KOs in a course) is also expressible via the set-based approach as it has been applied to find the (LP-) relevant KOs. Therefore, for each Rating Factor, a set is specified that fulfils a particular rating rule. The task of the Back End is then to find the elements that meet the respective conditions and to combine all this knowledge (cf. Figure 17). This is done by calculating the intersection of the set of KOs with a general recommendation value (i.e. those elements that have been selected previously on basis of LPs) and those sets that express the optimum regarding the different Rating Factors.

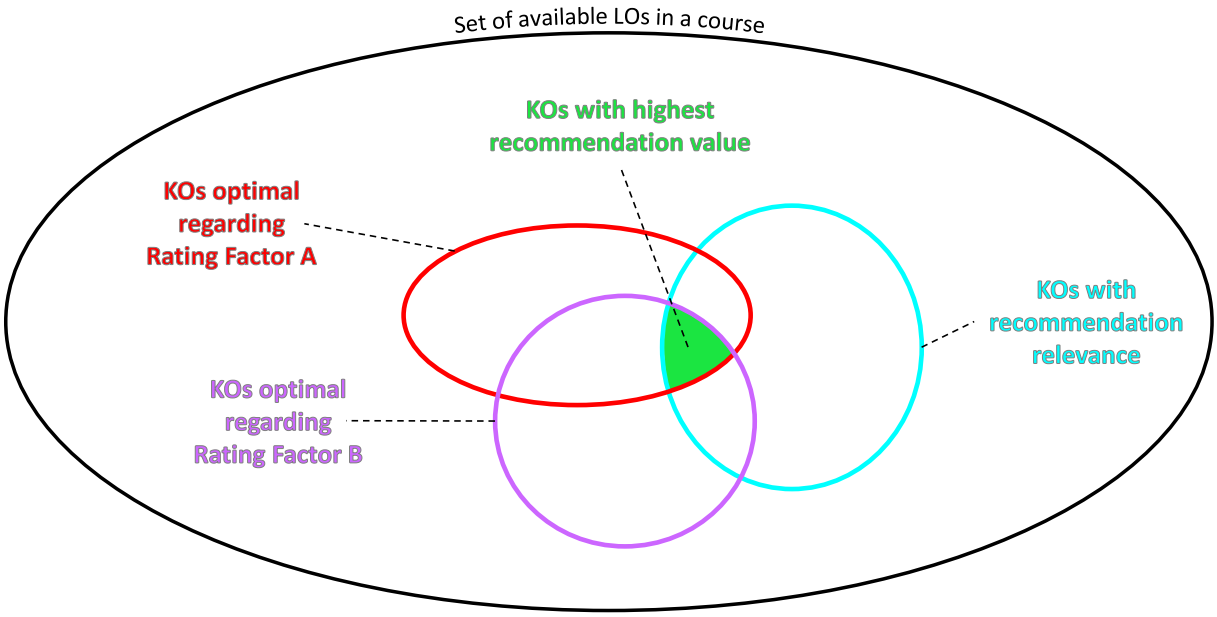


Figure 17: Set of KOs with the highest recommendation value

The complicated aspect of this approach is the question of how to express what is optimal regarding a Rating Factor and how to reasonably combine these sets. Although there might at first just be a rating stating that KOs are optimal regarding a Rating Factor, a more advanced version of INTUITEL

will certainly want to include more levels to differentiate in more detail. This means that a KO might, for example, be optimal, less optimal or bad. The Back End would thus not only have to identify the KOs of these sets but also define how to combine them in a reasonable way. Is a KO, for instance, optimal if it fulfils the most positive rating criteria or if it fulfils the least negative criteria or a combination of them? This of course increases the complexity of the process on a noticeable level. It is thus one of the main research questions of the Back End how to

1) express Rating Factors in an Ontology
2) express the rating levels (optimal, less optimal, bad – or another reasonable differentiation)
3) combine them to find the KOs with the highest suitability for the learner

These are all complex tasks and cannot be answered right away. Especially the last question is interesting because this basically is also a combinatorial problem. The current and upcoming tasks, i.e. the LM Ontology and the INTUITEL Engine, will have to look at these questions in detail to find an applicable and technically possible solution.

As these descriptions show is the complete Back End concept already specified on conceptual level. This allows the INTUITEL consortium to divide the individual aspects to the tasks and work packages in the most fitting way. It also shows that the suggested approach offers a high level of freedom to find the most suitable KOs for creating a recommendation. For INTUITEL, this is advantageous and valuable because it also allows a high level of customization of the process. Unfortunately, this also increases the complexity because the sets need to be specified correctly in order to not accidently omit relevant elements.

However, after solving the open questions and after technically implementing them, INTUITEL will be able to create recommendations with a high quality. Based on the approach of LPs and Didactic and Rating Factors, the criteria that INTUITEL uses to come to a conclusion are also highly customizable and extendable. This allows the developers to improve the process by adding new rating aspects in future versions and allows including new measures (e.g. eye-tracking) that will be available in specific or future LMS.

## 6.2 Functional Principle of the INTUITEL Back End

This section contains a description of how the Back End implements the previously introduced concept, to enable a better comprehension of the technical realization of the outlined functionality. There are multiple components that are part of the recommendation creation process and the individual tasks need to be distributed between them. These descriptions also show which tasks will be conducted by the LPM and which tasks the other modules will take over.

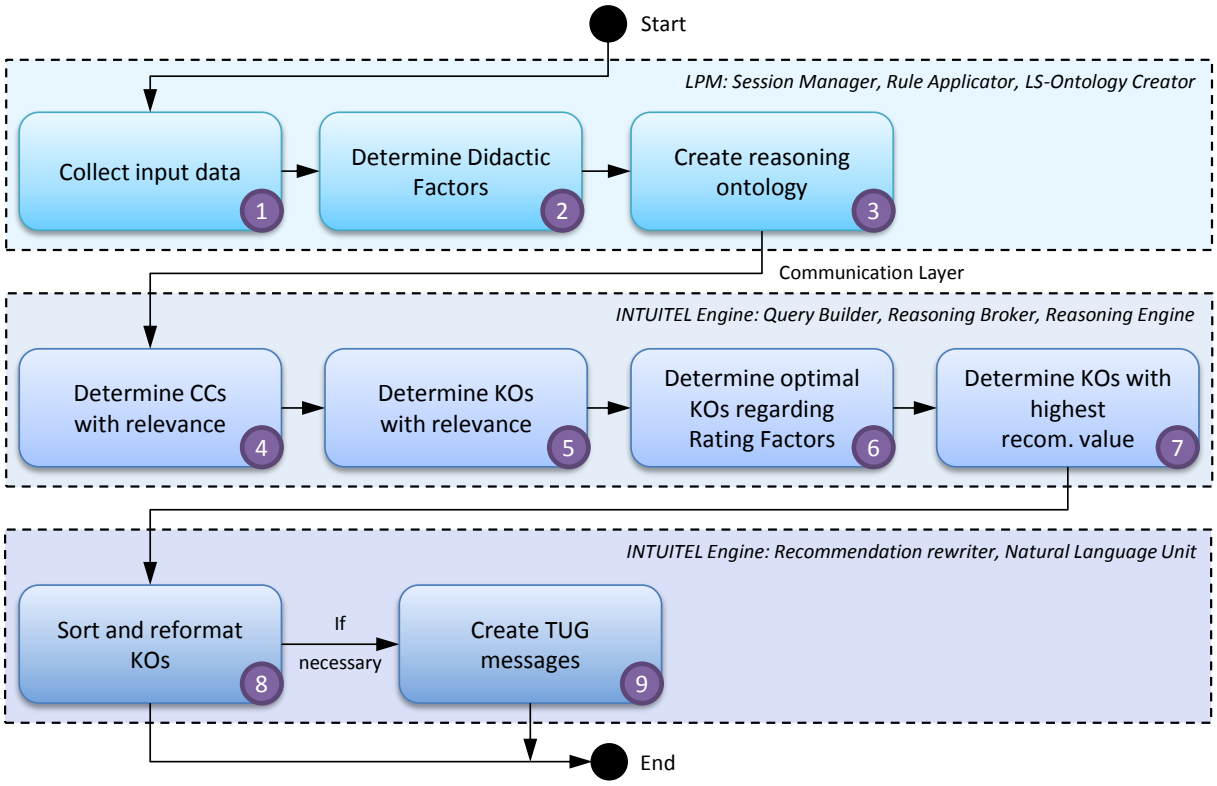


Figure 18: Simplified functional procedure for the recommendation creation in the Back End

The functional process has three stages[15], as depicted in Figure 18. There firstly is the stage of data preparation where the LPM collects and reformats the data for the INTUITEL Engine. Secondly, the reasoning process is conducted to determine the elements in the different sets as described in section 6.1. Thirdly, the results are sorted and reformatted to be in a format that is compatible with the rest of the INTUITEL system. Also, the TUG (natural language) messages are created.

In more detail, the process is going to be carried out as described in Table 3.

| Step | Description |
|---|---|
| Start | The Session Manager identifies a valid LO transition (meaning a certain learner opened a new LO in an INTUITEL-enabled course) and starts the recommendation creation process. |
| 1. | Collection of the data of the different input types as outlined in section 5.1. This step will also clarify which data can be included in the following process, since not all and the same data will be available for each learner and LMS. As much is needed and available via SLOM (inclusive the CM), the LMS and the LPM database will thereby be combined. |

[15] The reflex module has been omitted in this description for the sake of simplicity. In a nutshell, this just adds a shortcut after the data collection to the TUG-message creation.

| | |
|---|---|
| 2. | Conversion of the previously collected input data items into nominal (i.e. non-numeric) values, i.e. the Didactic Factors. This is not only a discretization (or quantisation), but a data-reduction as well. When also taking into account that the data can be combined in any arbitrary way, this is also a data-transformation.<br>The goal of this step is to apply as much rules as possible. If not all data items are available for a particular rule, the Didactic Factor will not be added to the LPM output.<br>This step will rely on an OWL-based description of Didactic Factors but will be carried out as part of the Java implementation of the LPM. |
| 3. | The INTUITEL Engine needs the output of the LPM in a compatible way, i.e. as ontology. Since the data is not necessarily available in a fitting format, the LPM converts the data appropriately. This also includes the addition of certain data items that, for instance, describe the completion state (unseen, unfinished, complete) of the contained LOs.<br>The thereby created Learner-State-Ontology will be transmitted to the INTUITEL Engine via the communication layer. |
| 4. & 5. | The set of available LOs is reduced in an iterative way to step-by-step identify the LOs that are relevant from a macro and micro LP perspective. The remaining set will then only contain the elements that are basically suitable for a recommendation for a certain learner.<br>If afterwards or during this procedure the set of LOs is empty, the process is either aborted because the course is complete regarding the LPs, or other measures will be taken to determine new LPs to change the selection criteria. |
| 6. | Similarly, to the procedure in step four and five are the KOs being identified which express a certain suitability for the learning progress regarding the different Rating Factors.<br>The Rating Factors are specified in OWL as part of the LM Ontology and affect the suitability of a KO in respect to their properties and the learner-dependent Didactic Factors. |
| 7. | As the last task of the Reasoning Engine, the set of KOs with the highest recommendation value will be calculated. The challenges regarding the specification and differentiation of the values of this aspect have been outlined in section 6.1. |
| 8. | Since the output of the reasoning process is probably not in a format that is compatible with INTUITEL (i.e. conform to the INTUITEL Data Model – D1.1) does the Recommendation Rewriter edit the results. It will therefore create a ranking and create the respective LORE message payload. |
| 9. | If at any stage of this process the creation of a TUG message is triggered, this module will create a natural language message that will be presented to the learner. |
| End | The recommendation(s) and TUG message(s) are send to the LMS. |

Table 3: Steps in the functional procedure of the Back End

## 6.3 The Tasks of the LPM

The specific tasks of the LPM regarding the recommendation creation process can be summarized with three core themes:

1) Data aggregation
2) Data storage
3) Data transformation

In order to accomplish these tasks, the LPM will rely on a multitude of different modules, which have shortly been introduced in section 3.6 and are explained in more detail in chapter 9. Building on these descriptions and the results of the previous deliverables, the data itself and how to aggregate it is already specified. It is therefore just a matter of what data is needed and available in the specific context. Thereby, especially the aspect of

- persistent data storage has to be taken into account. Depending on the data the Didactic Factors need, INTUITEL consequently needs to guarantee its presence, which means that
- the particular database tables need to include the historic learner data,
- which could also include the past conclusions/deductions of INTUITEL (e.g. changes of the LP).

The already specified Didactic Factors (cf. chapter 8) are the most important basis for identifying the therefore relevant information. However, as the recommendation creation process will be refined in the upcoming months, it is very probable that the table structures will be adapted and extended to be in line with the current status of INTUITEL. This is foreseen and desired to achieve up-to-date and sophisticated results.

When more specifically having a look at the recommendation creation process, the LPM has to be implemented in a technologically suitable way. This means that the LPM will have to combine different technologies to aggregate, transform and forward information from and to other INTUITEL components. This basically means that the LPM will have to implement

- the communication layer library to interact with other INTUITEL endpoints,
- an OWL-framework to read and write OWL files and
- a framework to exchange data with the LPM database.

In order to perform its main task in the recommendation creation process, the LPM (in combination with the LM Ontology) will need to specify and implement

- a standard method to describe Didactic Factors and
- a template (i.e. a class/interface structure) to implement transformation rules in Java.

The LPM will therefore create a Java based program that

- manages learner update messages,
- provides mechanisms to store and retrieve data comfortably and
- automatically applies the predefined transformation rules.

To complete its range of functions, the LPM will also include components to

- trigger direct user feedback before the reasoning process starts (Reflex Module),
- determine the most suitable LP (Learning Pathway Selector) and
- create the optimized data basis for the Reasoning Engine (LS-Ontology Creator).

Explicitly not task of the LPM is finding suitable LOs for a learner. This is solely task of the INTUITEL Engine. The LPM is only indirectly involved in this process since it provides and edits the relevant information.

## 6.4 Didactic Factors

A Didactic Factor is a compound of a number of data items from INTUITEL in a way so that the combination of them describes a fact that is relevant for the recommendation creation. They are the fundamental building blocks of the Rating Factors (cf. section 6.5), which are used to evaluate the suitability of KOs. For this purpose, everything that is available in the whole collection of INTUITEL data (see chapter 4), meaning the SLOM metadata, the Learning Pathways and especially the learner-specific information (e.g. the learning history as contained in the INTUITEL logs) that are stored or collected just-in-time from the LMSs, can be used.

From a technical perspective, a Didactic Factor is an OWL class which contains its own textual description. It furthermore also links to a Java class, containing its Transformation Rule. These are the instructions that specify in which combination of input data the respective Didactic Factor is valid. This combination of OWL and Java allows a very high flexibility regarding their specification, because all features of a high-level-programming language can be used. This especially also includes functionalities that would not be available in an OWL-only solution, as, for instance, mathematical methods to calculate the ratio between two values.

As seen from a reasoning point of view, the basic task of a Didactic Factor is to combine information in a way that allows its usage in context of an OWL-reasoner. These complex software modules have foundationally different intentions than programming frameworks like, for instance, Java or .Net. Instead of iteratively executing program code to produce various results, OWL-reasoners are specialized on testing the consistency of statements and the identification of relations between entities. By drawing conclusions on a dataset (i.e. an ontology) a reasoner can deduce statements that, for instance, allow to determine whether a CC is fitting for a certain learner's LP. The Didactic Factors are especially relevant in this process because natural or real numbers are problematic in that context. This entails that INTUITEL needs to reformat the input in a way that is compatible with such a system. One aspect of the Didactic Factors is consequently to transform the non-nominal values into a nominal form, e.g. by transforming the continuous value $x = 5, x \in \{n \in \mathbb{N} | 1 \leq n \leq 10\}$ into the categorized statement "medium".

There are four fundamental forms of Didactic Factors:

1) Trivial statement

   The most basic realization of a Didactic Factor is the n:1 relaying of input data. This means that certain data items are combined and translated into a format that is compatible with the Engine.

   Example: Gender (m/w) from USE → Gender (m/w) as Didactic Factor

2) Trivial input combination with grading

Different nominal data items can be connected to create a combined statement that entails some kind of grading.

Example: Connection type (Edge, UMTS, etc.) → Connection type as slow, medium, fast

3) Complex statement

A more complex use case for a Didactic Factor is the discretization of numerical values into a nominal one.

Example: Noise level in DB → Noise level as quiet, tolerable, loud

4) Complex input combination with grading

The combination of different (kinds of) input values, via e.g. mathematical functions, can also result in graded Didactic Factors.

Example: KO history and estimated learning times → Learning velocity slow, normal, fast

How exactly Didactic Factors are instantiated and how their validity is tested in certain learning situations is described in more detail in chapter 7. Which Didactic Factors are already foreseen for the first implementation of the INTUITEL Back End is described in chapter 8. Their OWL-based realization is part of the LM ontology.

## 6.5 Rating Factors

A Rating Factor is a logical conjunction of Didactic Factors and KO properties. This allows to state that KOs with a defined set of attributes are less or more suitable for the learning process when certain situational dependent conditions (i.e. Didactic Factors) are present. Rating Factors are thus, apart from the Learning Pathways, the fundamental influence factors for suggesting KOs to learners. They are not directly related to the tasks of the LPM, but are crucial for the concept of the Back End. Explaining their purpose and defining their basic nature is thus also relevant for the LPM.

As the connecting link between the input data and the recommendation process in the INTUITEL Engine, Rating Factors are purely described in OWL. Each of them names a set of Didactic Factors and KO properties and connects them logically. Take, for instance, the Rating Factor “Suitability of connection” which describes a condition under which certain KOs are less suitable:

| Didactic Factors | KO properties | Rating |
|---|---|---|
| (connection type medium AND connection stability bad)<br>OR<br>(connection type slow AND connection stability medium)<br>OR<br>(connection type slow AND connection stability bad) | size large<br>OR<br>Media Type video | negative |

Table 4: Description of Rating Factor “Suitability of connection”

As can be seen in Table 4, this Rating Factor has only one manifestation as a negative rating. By specifying that the combination of a bad connection (i.e. combination of connection type and

stability) is linked to the size of a KO or its Media Type, it is possible to state a reduced suitability for KOs that require a good connection to the Internet.

Given that it should only be expressed that a certain condition leads to an increased or reduced suitability, defining only one set of conditions for that Rating Factor is sufficient. Defining more combinations is only necessary when the suitability should be segmented further. How this rating and the combination of the different elements will be described in OWL is yet up for discussion. Likewise, it remains to be seen how many rating statements are reasonable. It would either be reasonable to punish certain conditions, to reward them or to do both in contrast to a neutral state[16]. This could of course also be extended to even more rating-levels (e.g. very negative, negative, neutral, positive and very positive). The more distinction possibilities are available, the more detailed the ratings, but likewise the more time-consuming and complex the reasoning process. A decision on this aspect and especially a specification of how this can be expressed in a reasoner-compatible way in OWL is one of the major research questions of the LM ontology and the INTUITEL Engine.

When mapping this approach on real learning situations, a Rating Factor describes something a tutor knows and that is of relevance for optimally guiding a learner through the learning process. It is thus some kind of didactic knowledge tutors have acquired in their own education that is based on the experience they have gathered throughout the years or is something they intuitively know. By applying this knowledge on the current situation, i.e. the learning material and the learner they are currently instructing, tutors can help their learners to learn better. Rating Factors try to mimic this behaviour with the help of the Didactic Factors and KO metadata to come to similar conclusions.

[16] A neutral state would not necessarily have to be modelled, because the absence of a rating already states the neutrality of the remaining KOs. However, to make sense, there generally must be a set of KOs that fall into this category, because otherwise the modelling of a second rating-set would unnecessarily complicate the Rating Factor.

# 7 LPM Concept

After the general concept of the Back End has been described, the actually interesting aspect for this document is how the LPM exactly fits into this and how it performs its tasks. To provide more detailed insights into the functionality of the LPM, this chapter contains a description of the structural concept of the LPM, the input and output values as well as their associated transformation rules. Furthermore, an illustrative example shows how the part of LPM in the recommendation creation process works.

## 7.1 Structural Concept

The LPM basically acts as a processor of the different input values. It takes some data, processes it and yields some output. Figure 19 illustrates this concept regarding the LPM on a very basic level.

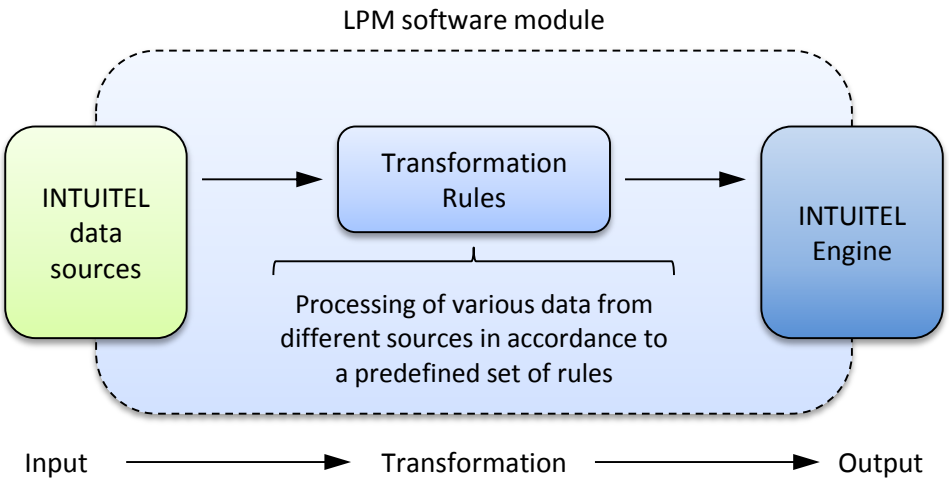


Figure 19: LPM processing concept

The LPM internally acts as a transformation entity. Information is fed into it, which is then processed in a module that generates the output in respect to a set of predefined rules. The basic questions for the LPM are the following four:

1) What is this input data and where does it come from?
2) What output data is valuable for the INTUITEL Engine?
3) Why are these transformations necessary?
4) How to describe these rules?

Regarding the first question, the answer has implicitly already been given in this document. As a mediating entity in information flow in INTUITEL, the LPM combines the learner specific input from the LMS, the pedagogical and domain input from the Pedagogical Ontology, the respective Cognitive

Models and SLOM metadata, as well as the preprocessed input already stored in the user database. Thus, the LPM can make use of all data available in INTUITEL. (cf. chapter 4 for more details).

The answer on the second question is a consequence of the overall concept of the Back End. In general, all information that is relevant for the reasoning process is valuable output data. These are particularly for the LPM the so called Didactic Factors (cf. section 6.4), the building blocks of the Rating Factors (cf. section 6.5). However, the availability of the Didactic Factors alone is insufficient, since the Engine needs an ontology to reason about. Consequently, the output of the LPM is a custom tailored ontology which contains all currently relevant information to rate the suitability of KOs (cf. section 7.3 for more details.)

This transformation process is necessary, because of three reasons. Firstly, it is complicated to use OWL-reasoners with continuous data. A mathematical statement like, for instance, “1 < 2” is problematic in that context. The same issue appears when the Engine should conclude that a certain KO with a video is less suitable if the user has less than 1 Mbit/s available. By transforming the problematic parts into discrete values (e.g. ‘very slow internet connection’), this is much easier. Secondly, the transformations reduce the amount of data for the Engine on a noticeable level. The more data is presented to the reasoner, the longer it takes to come to a result. It is thus reasonable to not hand over all available data, but only those items that are relevant. Thirdly, this approach also allows creating complex Didactic Factors. With the usage of a high-level programming language like Java, there are many more possibilities to combine and to check them against each other. This consequently also allows the creation of high level Rating Factors.

The last question is how to describe these rules. This basically is a two-part approach. Firstly, there is the LM Ontology which defines each specific Didactic Factor. To link them with their respective transformation rule, the package and class containing the Java code that determines if it applies is specified. Secondly, there are said Java classes that are present in an LPM compatible way (meaning they are implemented as the LPM architecture foresees it). Therefore, the LPM defines a set of interfaces that describe how a rule has to be specified (cf. section 7.4 for more details.)

If the basic processing concept of the LPM is combined with these answers, a more detailed overview of the LPM concept results (see Figure 20).

This also shows one important aspect why this approach is so advantegous. Although the overall concepts of the Back End and the LPM respectively are quite complex, the individual parts can be reduced to relatively simple abstraction levels (as illustrated in the beginning of this section). This modularization and simplification makes each aspect of this approach easier managable and solvable.

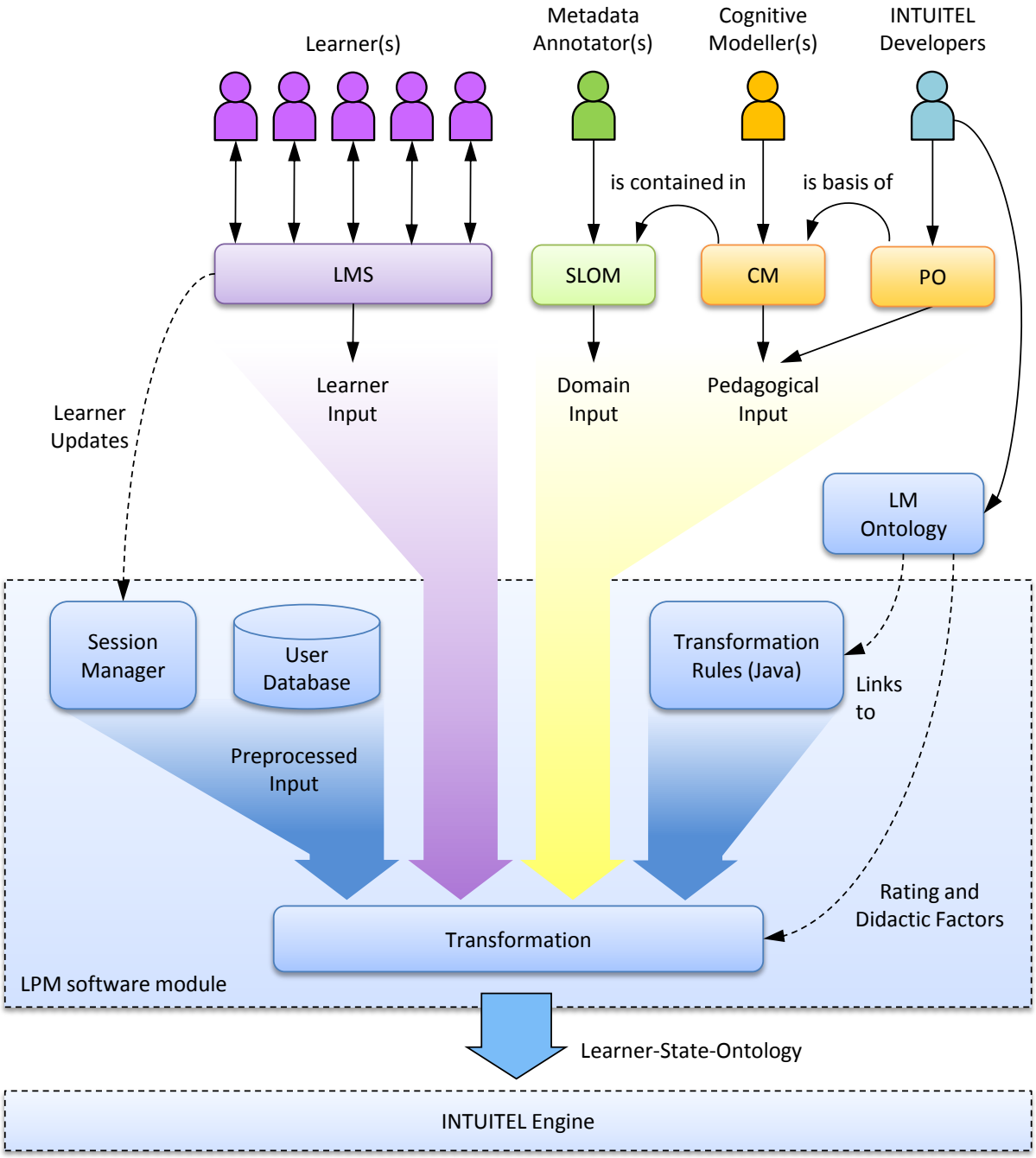


Figure 20: Detailed structural concept of the LPM

Using the possibilities of Java and OWL allows to benefit from the advantages of both technologies and results in a relatively lightweight crossover. In Java, developers can easily model complex evaluation methods, whereas OWL allows to describe that semantically and in a reasoner-compatible way.

Benefits:

- All data that is available can be used and combined in various ways.

- It seperates the programmatical issues from the pedagogical ones.
- Unittests allow to test the LPM and the recommendation creation process.
- It is possible to use math to e.g. apply statistical models or to calculate average values.
- Inclusion of (anonymized) knowledge about other students is possible.
- Very flexible and powerful ways to reduce data for the Engine.
- The building blocks (i.e. the Didactic Factors) need to be calculated only once, but can be resused in multiple Rating Factors.
- All (personal and sensitive) data about learners remains in one central location.
- Description and processing of transformations is much more powerful in Java than in OWL.
- Possibility to transform data in multiple threads (i.e. process is highly parallelizable).
- Definitions of Didactic Factors are readable without accessing the LPM source code.
- Which Didactic Factors are used is easily configurable.
- Definitions of Didactic Factors can easily extended.
- Definitions of Didactic Factors can easily be changed.

Especially the last two points about the extension and changing of Didactic Factors bear powerful customization options for the application of INTUITEL. Even though INTUITEL is composed of an interdisciplinary team of educational and computer scientists, it is not possible to cover all factors that influence learning. Later users will certainly come up with new factors and want to integrate them. Just by adding a new entry in the ontology and by adding a new class, new building blocks are directly available. Likewise, changing the definition of how a Didactic Factor is generated, LMS vendors could customize INTUITEL to be optimally usable in context of their specific system. Every organization or institution might have special demands that should or have to be considered in the recommendation creation process. Providing a functionality that is not fixed, but allows customization, is the best way to reach a bigger target audience and increase the potential of INTUITEL.

## 7.2 Illustrating Example

To illustrate the mechanism of the LPM with its transformation rules, the following shows the example of a learner velocity, i.e. a statement about how much time a learner needed in contrast to the estimated learning time.

The procedures are applicable on every other Didactic Factor. Only the setting and the rules vary in accordance to the respective learner situation and the INTUITEL configuration.

### 7.2.1 Setting

The setting of the learning situation in this example is the following:

- The learner has an already running session.
- The learner just opened KO P4.
- The learner has previously visited KOs P1, P2 and P3.

Furthermore, the following information is present in the LPM:

- The estimated learning times of all KOs is specified in the SLOM metadata.
- The start and end times of the learning times of P1, P2 and P3 are stored in the LPM.
- The start time of P4 is also present in the LPM (no end time, because it has not yet been closed).

Now, the following happens in the LPM:

1) The Session Manager notices that a new recommendation is needed.
2) To prepare the following process, the LPM updates its learner data (i.e. a USE request is send).[17]
3) The recommendation creation process is started.
4) The reflex mechanism tests if a reflex applies. (This is not the case for this example.)
5) The Rule Applicator applies as much rules as possible, including the learner velocity rule.
6) The custom ontology is created on basis of the previous step.
7) The custom ontology is handed over to the INTUITEL Engine.

The relevant part for this example is the fifth point, which is described in more detail in the following section.

### 7.2.2 Transformation Rule Example

In the transformation process, the LPM accesses the OWL definitions of each Didactic Factor and calls the respective Java classes which are specified there. Theoretically, there are multiple ways to define which conditions lead to a certain statement about learner velocity. What actually is carried out depends on the concrete instance of INTUITEL and what the respective INTUITEL operators perceive to be reasonable. It is also possible to define multiple rules for one Didactic Factor, with the used input as distinguishing feature. For the sake of this example, let's assume that two different rules are specifies:

- Learning velocity is calculated by analysing the last visited KO (lower priority).
- Learning velocity is calculated by analysing the last five visited KOs (higher priority).

In each case, the approach is the same. The OWL-definition of the Didactic Factor contains a natural language description of the Didactic Factor:

[17] Collecting SLOM data is not necessary, because the session is already running and the data is thus already available.

"Statement about how fast a learner completes Learning Objects in contrast to the estimated learning time."

It furthermore also contains the link to the class specifying the respective rule:

"eu.intuitel.lpm.rules.LearningVelocity"

By executing the respective methods in this class, the LPM firstly tries to execute the ones with the higher priority. This is not possible in this example, since there are only three KOs in the learner's history, whereas five KOs are needed. The LPM then applies the second rule, which is possible because the last visited KO's learning time is available. What happens now basically is the application of a function that states that if a certain threshold is (not) exceeded, a particular Didactic Factor results.

Let's assume that the estimated learning time is 3 minutes and the actual learning time was 2 minutes 30 seconds. Let's further assume that the transformation rule differentiates five cases:

1) Estimated time – actual time > 2 min => No rating
2) Estimated time – actual time < 2 min AND > 1 min => fast learner
3) Estimated time – actual time < 1 min AND > -1 min => normal learner
4) Estimated time – actual time < -1 min AND > -2 min => slow learner
5) Estimated time – actual time < -2 min => No rating

In the present example, this would result in the statement that the learner is a normal learner.

Please note that this is only an example. There can be arbitrarily many combinations of input values, but only a subset of them is pedagogically meaningful and exact enough. If, for example, the estimated learning time is quite high (e.g. 1 hour – which is non-compliant to the INTUITEL guidelines), completing the KO more than one minute earlier or later is certainly common. Thus, specifying well-engineered rules is an important factor regarding the accuracy of INTUITEL.

### 7.2.3 Template for Didactic Factors

To standardize the specification of transformation rules a template can be of assistance. In general, there are three relevant parts that need to be clarified. Firstly, it must be clear what input is relevant. Secondly, the output must be specified, i.e. the different Didactic Factors. Thirdly, the rule itself must be described on a textual basis. This approach is reasonable, since the programmatic realization is not necessarily carried out by the same person and other persons should be able to understand how the transformation rule is applied (also in retrospective).

When bringing these three aspects together, a simple table can be created. With the data from the present example, a concrete Didactic Factor template could look like the following:

| Input | Transformation Rule | Output |
|---|---|---|
| KO history<br>Start-times of KOs<br>End-times of KOs<br>Estimated learning times | Compare the actual learning time of the recent KO(s) with their estimated learning time and rate the temporal difference. | LearningVelocityFast<br>LearningVelocitySlow<br>LearningVelocityNormal |

Table 5: Didactic Factor description template with example content learning velocity

This shows that it is, in principle, fairly easy to describe a Didactic Factor. The actually problematic aspect is to be clear regarding the actual implementation of the transformation rule in order to exactly cover the relevant cases. The false combination of input data has an impact on KO selection (e.g. results in false-positives) and must thus be managed with great care. Furthermore, finding actually relevant and meaningful Didactic Factors is another aspect that requires sophisticated technological (i.e. one must know what is technologically possible) and especially pedagogical knowledge (i.e. finding and defining what actually has a relevance for the rating of KOs).

## 7.3 Learner-State-Ontology

The output of the LPM consists of two parts. Firstly, there is the output of the actual task that the LPM carries out, the set of applicable Didactic Factors. These are defined in context of the LM Ontology and are the building blocks of the Rating Factors. The LPM prepares the input and particularly prepares the Didactic Factors that are passed to the Engine. It, for instance, checks the connectivity of the learner and ranks it, to state that the learner has a good or bad connection. This represents the learner-specific knowledge of the reasoner and allows the Engine to include this information in the deductive process.

However, the set of actually relevant Didactic Factors alone is not sufficient for the INTUITEL Engine. An OWL reasoner needs an ontology to perform its work and therefore, the LPM must create one that contains all relevant data, i.e. the second part of the LPM output. This basically is a merge of the SLOM metadata of the respective course (including its Cognitive Model), the micro Learning Pathway information from the Pedagogical Ontology and the LM ontology, containing the machine-readable descriptions of Didactic and Rating Factors. To combine this information, the LPM creates the so called Learner-State-Ontology. This only temporarily valid collection depicts the current status of the learner in that course in respect to his or her learning history and current environment. How this ontology will actually be created depends on how the LM ontology will be structured, how much information should be handed over to the Engine and how the information should be edited to be most suitable for the reasoner. Consequently, the LPM will probably add statements that are relevant for the reasoning process (e.g. marking a KO as the currently viewed one) and probably also remove unnecessary parts (e.g. factors that are not valid).

To which degree this will be relevant is not yet foreseeable until the Back End specification progresses. Depending on the amount of additions that need to be made to the content, it might be advantageous or even necessary to add a functionality to define this in a more standardized way.

This could easily be adapted from the Didactic Factor approach and thus result in relatively a small workload.

A concluding statement about the actual shape of the Learner-State-Ontology cannot be made at this point. So, although it is not exactly clear how said ontology will look like from a technical perspective, its actual information content is already specified. This is an important aspect for the following tasks in INTUITEL. Changing the output format is only a minor change and easily implementable. The LPM architecture therefore foresees a module that is solely charged with the creation of the Learner-State-Ontology, the LS-Ontology Creator (cf. section 9.6). This allows the LPM to restructure the output without having to change the modules that are concerned with data management and the application of the transformation rules.

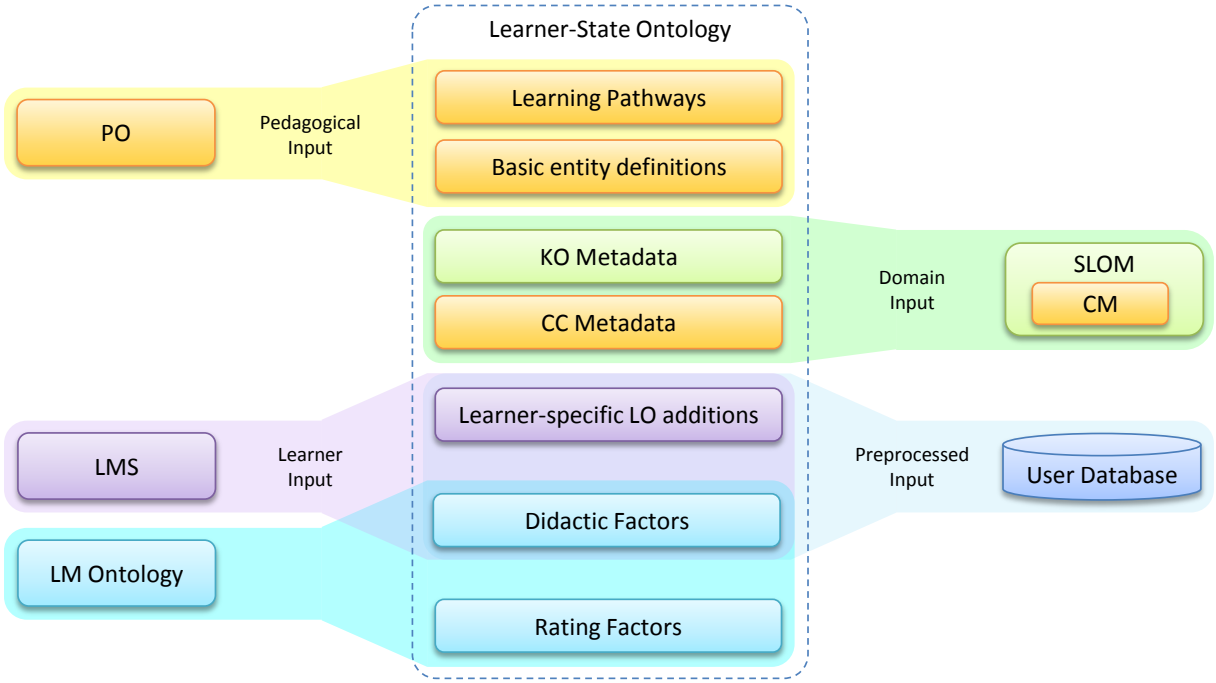


Figure 21: Composition of the Learner-State Ontology

## 7.4 Transformation Rules

A transformation rule of a Didactic Factor generally is a set of methods in a Java class. It specifies if the rules to form a certain Didactic Factor can be satisfied.

Depending on how the LM Ontology is going to implement the different levels of a Didactic Factor (e.g. 'good connectivity' vs. 'bad connectivity', hierarchically connected with a basic 'connectivity' Didactic Factor or without it), the actual implementation might also have to test which characteristic of it is valid.

Basically, transformation rules reduce/convert various kinds of input values in various formats (numeric values, Boolean values or Enumerations) to well-defined nominal output values. The output values are then used in the reasoning process in the INTUITEL Engine.

For its implementation, the OWL-description of the related Didactic Factor textually describes what the particular factor expresses, i.e. its semantic meaning in the learning or rating process. This also includes directions or hints how the respective input variables should be combined to achieve the desired results. The programmers who implement the transformation rules can then build on these descriptions and implement them accordingly.

This is also one of the major advantages of the Didactic Factor approach, because educational experts don't necessarily require sophisticated programming knowledge or need to know how the software works internally. Furthermore, the OWL description of a Didactic Factor should always be specifiable in the same way. If the general principle is clear, it should be possible to just fill out a template in order to specify a new Didactic Factor.

This reduces the initial burden of applying or implementing new rules because the technical and educational aspects are separable. In an interdisciplinary fashion, it allows for a collaboration of technical and educational experts to work out and improve the measures in INTUITEL - which in turn increases the durability of the project results.

A transformation rule thus is a programmatic realization that combines input in a way to actually state the educationally relevant factor that is of interest for the reasoning process.

In the simplest scenario, this is just an if-then-else clause which maps value ranges. For instance, it could be a statement that says if the learner's connection stability is in the interval 80 to 100, then the connection stability is rated as very good, and if not, the remaining interval could be evaluated in the same manner.

Another example is the calculation of the learner's learning velocity v=d/t. The velocity v is the distance d divided by time t. The distance would be the number of Learning Objects that have been completed in the specified timespan t. To be compatible with an OWL reasoner the numerical output v is interpreted and mapped to a nominal class value, like for example 'fast learning velocity'.

# 8 Specification of Didactic Factors Transformation Rules

This chapter provides an overview of the Didactic Factors that have been developed in context of the LPM so far.

This list does not claim to be complete or that the respective items are final. Changes might occur in the following development process or due to the results of connected tasks. This list will nevertheless give a detailed introduction into which aspects the INTUITEL consortium considers to be relevant for the selection of suitable Learning Objects and which are also applicable with the previously described Back End concept.

The ordering of the elements does not connote any kind of prioritisation. Though the Didactic Factors have been reviewed by the INTUITEL partners, an exhaustive evaluation regarding their value for the recommendation process is not part of this development step.

## 8.1 Didactic Factor Descriptions

Please note that the following statements generally refer on the course-specific data, if it is not explicitly stated otherwise.

| # | Name | Description |
|---|---|---|
| 01 | Knowledge actuality | Ranking of time between now and the last learning session. |
| 02 | Course-focused KO learning speed | Ranking of learning time the learner on average differs from the estimated learning time in contrast to the same measure of the other course participants. |
| 03 | Learner-focused KO learning speed | Ranking of learning time the learner on average differs from the estimated learning time of completed KOs of this session in contrast to same measure over all KOs over all sessions. |
| 04 | Course-focused filtered KO learning speed | Ranking of learning time the learner on average differs from the estimated learning time in contrast to the same measure of the other course participants when only having a look at KOs that have the same KT and MT. |
| 05 | Learner-focused filtered KO learning speed | Ranking of learning time the learner on average differs from the estimated learning time of completed KOs of this session in contrast to same measure over all KOs over all sessions when only having a look at KOs that have the same KT and MT. |
| 06 | Course-focused session length | Statement about the average session length as compared to the average session length of other course participants. |

| | | |
|---|---|---|
| 07 | Learner-focused session length | Statement about the current session length as compared to the average session length of the learner. |
| 08 | Time exposure | Comparison between the amount of time the learner and the other course participants spent on the course. |
| 09 | Learning Pathway permanence | Ranking of the amount of KOs the learner completed on the current LP combination in contrast to the same measure for the other course participants. |
| 10 | Recent learning pace | Comparison of the actual learning time the learner needed for the last 10 KOs in contrast to the estimated learning time. |
| 11 | Session learning pace | Comparison of the actual learning time the learner needed for the KOs in this session in contrast to their estimated learning time. |
| 12 | Course-focused LP usage type | Statement about the LP usage as measured on the learner's pathway switches and the switches of the other course participants. |
| 13 | Learner-focused LP usage type | Statement about the LP usage as measured on the learner's pathway switches. |
| 14 | Course-focused learning success | Success of the learner regarding scores in contrast to the other course participants. |
| 15 | Learner-focused learning success | Success of the learner regarding scores in the learner's current session in contrast of the own score history. |
| 16 | Course-focused KO repetition quantity | Comparison of the number of repeated KOs with the number of repetitions of the other course participants. |
| 17 | Learner-focused KO repetition quantity | Comparison of the number of repeated KOs in the recent KO history and the average of repeated KOs of one learner. |
| 18 | Course-focused CC repetition quantity | Comparison of the number of repeated CCs with the number of repetitions of the other course participants. |
| 19 | Learner-focused CC repetition quantity | Comparison of the number of repeated CCs in the recent KO history and the average of repeated CCs of one learner. |
| 20 | Course KO completion | Statement about the completion of the course regarding the completion states of KOs. |
| 21 | Course CC completion | Statement about the completion of the course regarding the completion states of CCs. |

| | | |
|---|---|---|
| 22 | CC KO completion | Statement about the completion of the current CC regarding the completion states of the connected KOs. |
| 23 | Course-focused KO completion tendency | Comparison of the learner's and the other course participants' ratio of completed KOs in contrast to the uncompleted ones of the session. |
| 24 | Learner-focused KO completion tendency | Comparison of the learner's session-related and overall ratio of completed to uncompleted KOs. |
| 25 | Course-focused MT preference | Statement about the MT preference as measured on all course participant selections. |
| 26 | Learner-focused MT preference | Statement about the MT preference as measured by the learner's learning history. |
| 27 | Course-focused MT dislike | Statement about the MT dislike as measured on all course participant selections. |
| 28 | Learner-focused MT dislike | Statement about the MT dislike as measured by the learner's learning history. |
| 29 | Course-focused KT preference | Statement about the KT preference as measured on all course participant selections. |
| 30 | Learner-focused KT preference | Statement about the KT preference as measured by the learner's learning history. |
| 31 | Course-focused KT dislike | Statement about the KT dislike as measured on all course participant selections. |
| 32 | Learner-focused KT dislike | Statement about the KT dislike as measured by the learner's learning history. |
| 33 | LP leaving position | Statement at which point (in the sense of completed LOs) the learner leaves a LP. |
| 34 | Course-focused learning efficiency | Ranking of how much time the learner needs to complete a KO in contrast to the time the other course participants needed for it. |
| 35 | Learning attention | Statement about how much attention the learner pays to the content as measured by an eye-tracking device connected to the LMS. |
| 36 | Blindness | Statement if the learner is blind. |

| | | |
|---|---|---|
| 37 | Deafness | Statement if the learner is deaf. |
| 38 | Gender | Statement about the learner's gender. |
| 39 | Age | Statement about the learner's age. |
| 40 | EQF Level | Statement about the learner's European Qualification Framework (EQF) level. |
| 41 | Learner level | Statement about the level of knowledge the learner possesses. |
| 42 | Device resolution | Ranking of the resolution of the device the learner uses to access the LMS. |
| 43 | Device pixel density | Ranking of the pixel density of the device the learner uses to access the LMS |
| 44 | Connectivity level | Ranking of the connectivity between the learner's access device and the LMS. |
| 45 | Noise level | Ranking of the environmental noise level of the learner. |
| 46 | Learning Environment | Statement about the type of environment the learner is currently located at. |
| 47 | Battery status | Statement about the current battery status of the device the learner is using to access the LMS. |

Table 6: Description of Didactic Factors

## 8.2 Didactic Factor Transformation Rules

Please note, that due to a better readability, standard deviation is generally abbreviated by the symbol σ in the transformation rules.

| # | Input | Transformation Rule | Output |
|---|---|---|---|
| 01 | ld = Last login date<br>cd = Current date | if ld - cd > 14 days<br>→ LastLoginLongAgo<br>else<br>→ LastLoginRecently | LastLoginLongAgo,<br>LastLoginRecently |
| 02 | lAvg = Learner's average difference between actual and | if lAvg > oAvg + σ<br>→ KoSpeedSlow | KoSpeedFast<br>KoSpeedSlow |

| | | | |
|---|---|---|---|
| | estimated learning time<br>oAvg = Others' average difference between actual and estimated learning time | else if lAvg < oAvg - σ<br>→ KoSpeedFast<br>else<br>→ KoSpeedNormal | KoSpeedNormal |
| 03 | lAvgRec = Learner'σ average difference between actual and estimated learning time of the KOs of this session<br>lAvgGen = Learner's average difference between actual and estimated learning time of the KOs of all sessions | if lAvgRec > lAvgGen + σ<br>→ KoSpeedSlower<br>else if lAvgRec < lAvgGen - σ<br>→ KoSpeedFaster<br>else<br>→ KoSpeedNormal | KoSpeedFaster<br>KoSpeedSlower<br>KoSpeedNormal |
| 04 | lCouples[,] = Learner's average difference between actual and estimated learning time of KOs, which is differentiated into KT and MT couples (only the three topmost types each)<br>oCouples[] = Others' average difference between actual and estimated learning time of KOs, which is differentiated into KT and MT couples (only the three topmost types each) | For each couple {<br>if couple not null for learner {<br>lAvg += learner's value for couple<br>oAvg += others' value for couple<br>} }<br>lAvg /= count of not null couples<br>oAvg /= count of not null couples<br>if lAvg > 110% of oAvg<br>→ FilteredKoSpeedSlow<br>else if lAvg < 90 % of oAvg<br>→ FilteredKoSpeedFast<br>else<br>→ FilteredKoSpeedNormal | FilteredKoSpeedFast<br>FilteredKoSpeedSlow<br>FilteredKoSpeedNormal |
| 05 | lSessionCouples [,] = Learner's average difference between actual and estimated learning time of KOs, which is differentiated into KT and MT couples (only the three topmost types each) of the current session<br>lGeneralCouples [,] = Learner's average difference between actual and estimated learning time of KOs, which is differentiated into KT and MT couples (only the three topmost types each) of all sessions | For each couple {<br>if couple not null for learner {<br>lAvgS += learner's value for couple in current session<br>lAvgG += learner's value for couple in all sessions<br>} }<br>lAvgS /= count of not null couples<br>lAvgG /= count of not null couples<br>if lAvgS > 110% of lAvgG<br>→ FilteredKoSpeedSlower<br>else if lAvgS < 90 % of lAvgG<br>→ FilteredKoSpeedFaster<br>else<br>→ FilteredKoSpeedStable | FilteredKoSpeedFaster<br>FilteredKoSpeedSlower<br>FilteredKoSpeedStable |

| | | | |
|---|---|---|---|
| 06 | lSession = Length of current sessions of the learner<br>oAvgSession = Average length of all sessions of course participants | if lSession > oAvgSession + σ<br>→ SessionLenghthLong<br>else if lSession < oAvgSession - σ<br>→ SessionLenghthShort<br>else<br>→ SessionLenghthAvg | SessionLenghthLong<br>SessionLenghthShort<br>SessionLenghthAvg |
| 07 | lSession = Length of current sessions of the learner<br>lAvgSession = Average length of all sessions of the learner | if lSession > lAvgSession + σ<br>→ SessionLenghthLong<br>else if lSession < lAvgSession - σ<br>→ SessionLenghthShort<br>else<br>→ SessionLenghthAvg | SessionLenghthLonger<br>SessionLenghthShorter<br>SessionLenghthStable |
| 08 | lAvgTime = Average time the learner spent on course<br>oAvgTime = Average time others spent on course | if lAvgTime > oAvgTime + σ<br>→ TimeExposureLong<br>else if lAvgTime < oAvgTime - σ<br>→ TimeExposureShort<br>else<br>→ TimeExposureAvg | TimeExposureLong<br>TimeExposureShort<br>TimeExposureAvg |
| 09 | lAmount = Number of KOs the learner completed on current LP combination<br>oAmount = Average number of KOs others have completed on the same LP combination | if lAmount > oAmount + σ<br>→ LpPermanenceHigh<br>else if lAmount < oAmount - σ<br>→ LpPermanenceLow<br>else<br>→ LpPermanenceNormal | LpPermanenceHigh<br>LpPermanenceLow<br>LpPermanenceNormal |
| 10 | lRecLt = Learner's learning time of the last 10 KOs of this session (ignoring sneak peeks and less than 10 if not enough available)<br>eRecLt = Estimated learning time the same KOs | if lRectLt > 160% of eRecLt<br>→ RecentPaceVerySlow<br>else if lRecLt > 110% of eRecLt<br>→ RecentPaceSlow<br>else if lRecLt < 90 % of eRecLt<br>→ RecentPaceFast<br>else if lRecLt < 40 % eRecLt<br>→ RecentPaceVeryFast<br>else<br>→ RecentPaceNormal | RecentPaceVerySlow<br>RecentPaceSlow<br>RecentPaceFast<br>RecentPaceVeryFast<br>RecentPaceNormal |
| 11 | lSesLt = Learner's learning time of the KOs of this session (ignoring | if lSesLt > 160% of eSesLt<br>→ SessionPaceVerySlow | SessionPaceVerySlow<br>SessionPaceSlow |

| | | | |
|---|---|---|---|
| | sneak peeks)<br>eSesLt = Estimated learning time the same KOs | else if lSesLt > 110% of eSesLt<br>→ CurrentPaceSlow<br>else if lSesLt < 90 % of eSesLt<br>→ CurrentPaceFast<br>else if lSesLt < 40 % eSesLt<br>→ SessionPaceVeryFast<br>else<br>→ CurrentPaceNormal | SessionPaceFast<br>SessionPaceVeryFast<br>SessionPaceNormal |
| 12 | lSwitches = Number of pathway switches of the learner divided by learner's session count<br>oSwitches = Number of pathway switches of the others divided by their total session count | if lSwitches > oSwitches + σ<br>→ LpUsageChaotic<br>else if lSwitches < oSwitches - σ<br>→ LpUsageRigid<br>else<br>→ LpUsageNormal | LpUsageChaotic<br>LpUsageRigid<br>LpUsageNormal |
| 13 | lSwiSession = Number of pathway switches of the learner's current session<br>lSwiTotal = Learner's average number of pathway switches per session | if lSwiSession > lSwiTotal + σ<br>→ LpUsageMoreChaotic<br>else if lSwiSession < lSwiTotal - σ<br>→ LpUsageMoreRigid<br>else<br>→ LpUsageStable | LpUsageMoreChaotic<br>LpUsageMoreRigid<br>LpUsageStable |
| 14 | lSco = Average learner score<br>oSco = Average score of course participants | if lSco > oSco + $\sigma^2$<br>→ SuccessVeryGood<br>else if lSco > oSco + σ<br>→ SuccessGood<br>else if lSco < oSco - σ<br>→ SuccessSufficient<br>else if lSco < oSco - b<br>→ SuccessInsufficient<br>else<br>→ SuccessAverage | SuccessVeryGood<br>SuccessGood<br>SuccessAverage<br>SuccessSufficient<br>SuccessInsufficient |
| 15 | scoRec = Average learner score in the learner's current session<br>scoGen = General average learner score | if scoRec > scoGen + σ<br>→ SuccessBetter<br>else if scoRec < scoGen - σ<br>→ SuccessWorse<br>else<br>→ SuccessStable | SuccessBetter<br>SuccessStable<br>SuccessWorse |

| | | | |
|---|---|---|---|
| 16 | lRep = Learner's KO repetition quantity divided by visited KO count<br>oRep = Others' KO repetition quantity divided by visited KO count | if lRep > 140% of oRep<br>→ KoRepetitionOften<br>else if lRep < 60% of oRep<br>→ KoRepetitionSeldom<br>else<br>→ KoRepetitionNormal | KoRepetitionOften<br>KoRepetitionSeldom<br>KoRepetitionNormal |
| 17 | lRepRec = learner's KO repetition ratio in the KOs of the learner's current session<br>lRepGen = learner's overall KO repetition ratio in this course | if lRepRec > 140% of lRepGen<br>→ KoRepetitionMoreOften<br>else if lRepRec < 60% of lRepGen<br>→ KoRepetitionRarer<br>else<br>→ KoRepetitionStable | KoRepetitionMoreOften<br>KoRepetitionStable<br>KoRepetitionRarer |
| 18 | lRep = learner's CC repetition quantity divided by visited CC count<br>oRep = others' CC repetition quantity divided by visited CC count | if lRep > 140% of oRep<br>→ CcRepetitionOften<br>else if lRep < 60% of oRep<br>→ CcRepetitionSeldom<br>else<br>→ CcRepetitionNormal | CcRepetitionOften<br>CcRepetitionSeldom<br>CcRepetitionNormal |
| 19 | lRepRec = learner's CC repetition ratio in the recent CCs<br>lRepGen = learner's overall CC repetition ratio in this course | if lRepRec > 140% of lRepGen<br>→ CcRepetitionMoreOften<br>else if lRepRec < 60% of lRepGen<br>→ CcRepetitionRarer<br>else<br>→ CcRepetitionStable | CcRepetitionMoreOften<br>CcRepetitionStable<br>CcRepetitionRarer |
| 20 | koCompletion = completed KOs divided by total number of KOs in current course excluding KO alternatives | if koCompletion >= 100%<br>→ KoCompletionFull<br>else if koCompletion >= 75%<br>→ KoCompletionHigh<br>else if koCompletion >= 50%<br>→ KoCompletionMed<br>else if koCompletion >= 25%<br>→ KoCompletionLow<br>else<br>→ KoCompletionVeryLow | KoCompletionFull<br>KoCompletionHigh<br>KoCompletionMed<br>KoCompletionLow<br>KoCompletionVeryLow |
| 21 | ccCompletion = completed CCs divided by total number of CCs in current course excluding CC | if ccCompletion >= 100%<br>→ CcCompletionFull | CcCompletionFull<br>CcCompletionHigh |

| | | | |
|---|---|---|---|
| | alternatives | else if ccCompletion >= 75%<br>→ CcCompletionHigh<br>else if CcCompletionMed >= 50%<br>→ CcCompletionMed<br>else if ccCompletion >= 25%<br>→ CcCompletionLow<br>else<br>→ CcCompletionVeryLow | CcCompletionMed<br>CcCompletionLow<br>CcCompletionVeryLow |
| 22 | ccKoCompletion = completed KOs divided by total number of KOs in current CC excluding KO alternatives | if ccKoCompletion >= 100%<br>→ KoCcComplFull<br>else if ccKoCompletion >= 75%<br>→ KoCcComplHigh<br>else if ccKoCompletion >= 50%<br>→ KoCcComplMedium<br>else if ccKoCompletion >= 25%<br>→ KoCcComplLow<br>else<br>→ KoCcComplVeryLow | KoCcComplFull<br>KoCcComplHigh<br>KoCcComplMedium<br>KoCcComplLow<br>KoCcComplVeryLow |
| 23 | lRatio = Learner's ratio of completed KOs to accessed but not completed KOs in the current session<br>oRatio = Average ratio of completed KOs to accessed but not completed KOs of all course participants' sessions | if lRatio > oRatio + σ<br>→ KoComplTendencyHigh<br>else if lRatio < oRatio - σ<br>→ KoComplTendencyLow<br>else<br>→ KoComplTendencyNormal | KoComplTendencyHigh<br>KoComplTendencyLow<br>KoComplTendencyNormal |
| 24 | lRatio = Learner's ratio of completed KOs to accessed but not completed KOs in the current session<br>lAvgRatio = Learner's average ratio of completed KOs to accessed but not completed KOs across all sessions | if lRatio > lAvgRatio + σ<br>→ KoComplTendencyHigher<br>else if lRatio < lAvgRatio - σ<br>→ KoComplTendencyLower<br>else<br>→ KoComplTendencyStable | KoComplTendencyHigher<br>KoComplTendencyLower<br>KoComplTendencyStable |
| 25 | source = List of KO alternatives of the course<br>comCount = Number of KOs completions from source that are of MT communication media<br>interCount = Number of KOs completions from source that are | if difference is to small<br>→ no rating<br>if comCount > interCount {<br>if comCount > presCount<br>→ CoursePrefMtCom<br>else | CoursePrefMtCom<br>CoursePrefMtInteractive<br>CoursePrefMtPres |

| | | | |
|---|---|---|---|
| | of MT interaction media<br>presCount = Number of KOs completions from source that are of MT presentation media | → CoursePrefMtPres<br>}<br>else if interCount > presCount<br>→ CoursePrefMtInteractive<br>else<br>→ CoursePrefMtPres | |
| 26 | source = List of KO alternatives of the course<br>comCount = Number of KOs the learner completed from source that are of MT communication media<br>interCount = Number of KOs the learner completed from source that are of MT interaction media<br>presCount = Number of KOs the learner completed from source that are of MT presentation media | if difference is to small<br>→ no rating<br>if comCount > interCount {<br>if comCount > presCount<br>→ LearnerPrefMtCom<br>else<br>→ LearnerPrefMtPres<br>}<br>else if interCount > presCount<br>→ LearnerPrefMtInter<br>else<br>→ LearnerPrefMtPres | LearnerPrefMtCom<br>LearnerPrefMtInter<br>LearnerPrefMtPres |
| 27 | source = List of KO alternatives of the course<br>comCount = Number of KOs completions from source that are of MT communication media<br>interCount = Number of KOs completions from source that are of MT interaction media<br>presCount = Number of KOs completions from source that are of MT presentation media | if difference is to small<br>→ no rating<br>if comCount < interCount {<br>if comCount < presCount<br>→ CourseDislikeMtCom<br>else<br>→ CourseDislikeMtPres<br>}<br>else if interCount < presCount<br>→ CourseDislikeMtInter<br>else<br>→ CourseDislikeMtPres | CourseDislikeMtCom<br>CourseDislikeMtInter<br>CourseDislikeMtPres |
| 28 | source = List of KO alternatives of the course<br>comCount = Number of KOs the learner completed from source that are of MT communication media<br>interCount = Number of KOs the learner completed from source that are of MT interaction media | if difference is to small<br>→ no rating<br>if comCount < interCount {<br>if comCount < presCount<br>→ LearnerDislikeMtCom<br>else<br>→ LearnerDislikeMtPres<br>} | LearnerDislikeMtCom<br>LearnerDislikeMtInter<br>LearnerDislikeMtPres |

| | | | |
|---|---|---|---|
| | presCount = Number of KOs the learner completed from source that are of MT presentation media | else if interCount < presCount<br>→ LearnerDislikeMtInter<br>else<br>→ LearnerDislikeMtPres | |
| 29 | source = List of KO alternatives of the course<br>CoopCount = Number of KOs completions from source that are of KT cooperative knowledge<br>InteractCount = Number of KOs completions from source that are of KT interactive knowledge<br>ReceptCount = Number of KOs completions from source that are of KT receptive knowledge | if difference is to small<br>→ no rating<br>if CoopCount > InteractCount<br>{<br>if CoopCount > ReceptCount<br>→ CoursePrefCoop<br>else<br>→ CoursePrefReceptive<br>}<br>else if InteractCount > ReceptCount<br>→ CoursePrefInteractive<br>else<br>→ CoursePrefReceptive | CoursePrefCoop<br>CoursePrefInteractive<br>CoursePrefReceptive |
| 30 | source = List of KO alternatives of the course<br>CoopCount = Number of KOs the learner completed from source that are of KT cooperative knowledge<br>InteractCount = Number of KOs the learner completed from source that are of KT interactive knowledge<br>ReceptCount = Number of KOs the learner completed from source that are of KT receptive knowledge | if difference is to small<br>→ no rating<br>if CoopCount > InteractCount<br>{<br>if CoopCount > ReceptCount<br>→ LearnerPrefCoop<br>else<br>→ LearnerPrefReceptive<br>}<br>else if InteractCount > ReceptCount<br>→ LearnerPrefInteractive<br>else<br>→ LearnerPrefReceptive | LearnerPrefCoop<br>LearnerPrefInteractive<br>LearnerPrefReceptive |
| 31 | source = List of KO alternatives of the course<br>CoopCount = Number of KOs completions from source that are of KT cooperative knowledge<br>InteractCount = Number of KOs completions from source that are of KT interactive knowledge<br>ReceptCount = Number of KOs | if difference is to small<br>→ no rating<br>if CoopCount < InteractCount<br>{<br>if CoopCount < ReceptCount<br>→ CourseDislikeCoop<br>else<br>→ CourseDislikeReceptive | CourseDislikeCoop<br>CourseDislikeInteractive<br>CourseDislikeReceptive |

| | | | |
|---|---|---|---|
| | completions from source that are of KT receptive knowledge | }<br>else if InteractCount < ReceptCount<br>→ CourseDislikeInteractive<br>else<br>→ CourseDislikeReceptive | |
| 32 | source = List of KO alternatives of the course<br>CoopCount = Number of KOs the learner completed from source that are of KT cooperative knowledge<br>InteractCount = Number of KOs the learner completed from source that are of KT interactive knowledge<br>ReceptCount = Number of KOs the learner completed from source that are of KT receptive knowledge | if difference is to small<br>→ no rating<br>if CoopCount > InteractCount<br>{<br>if CoopCount > ReceptCount<br>→ LearnerDislikeCoop<br>else<br>→ LearnerDislikeReceptive<br>}<br>else if InteractCount > ReceptCount<br>→ LearnerDislikeInteractive<br>else<br>→ LearnerDislikeReceptive | LearnerDislikeCoop<br>LearnerDislikeInteractive<br>LearnerDislikeReceptive |
| 33 | leavingPos[,] = 2d-array containing the number of KOs that the learner completed before leaving a LP and the total number of steps on that LP<br>ratio = Average ratio between leaving positions and LP lengths as described in leavingPos[,] | if ratio > 80% && ratio < 99%<br>→ LpLeavingEnd<br>else if ratio > 60%<br>→ LpLeavingLate<br>else if ratio > 40%<br>→ LpLeavingMiddle<br>else if ratio > 20%<br>→ LpLeavingEarly<br>else<br>→ LpLeavingBeginning | LpLeavingBeginning<br>LpLeavingEarly<br>LpLeavingMiddle<br>LpLeavingLate<br>LpLeavingEnd |
| 34 | lTimePerKo = Sum of time the learner spent on the course divided by the number of completed KOs<br>oTimePerKo = Sum of time the others spent on the course divided by the number of completed KOs | if lTimePerKo > oTimePerKo + σ<br>→ LearningEfficiencyHigh<br>else if lTimePerKo < oTimePerKo - σ<br>→ LearningEfficiencyLow<br>else<br>→ LearningEfficiencyNormal | LearningEfficiencyHigh<br>LearningEfficiencyLow<br>LearningEfficiencyNormal |
| 35 | eyeMovementPerKo[,] = 2d-array containing the MT of the KOs of current session and the tracked | for each entry in the array {<br>distractionLvl += Rating of distraction depending on of the MT | LearnerIsDistracted<br>LearnerIsFocused |

| | | | |
|---|---|---|---|
| | eye-movement pattern | of the current array-entry<br>}<br>distractionLvl /= array entry count<br>if distractionLvl > 50%<br>→ LearnerIsDistracted<br>else if distractionLvl < 20%<br>→ LearnerIsFocused<br>else<br>→ LearnerIsAttentive | LearnerIsAttentive |
| 36 | blind = Boolean statement from learner profile or from TUG stating that the learner is blind or not | if blind<br>→ LearnerIsBlind | LearnerIsBlind |
| 37 | deaf = Boolean statement from learner profile or from TUG stating that the learner is deaf or not | if deaf<br>→ LearnerIsDeaf | LearnerIsDeaf |
| 38 | gender = Statement of the gender of the learner from the learner profile or from TUG | if gender == male<br>→ LearnerIsMale<br>else if gender == female<br>→ LearnerIsFemale | LearnerIsMale<br>LearnerIsFemale |
| 39 | age = Number specifying the age of the learner from the learner profile or from TUG | if age > 65<br>→ LearnerIsSenior<br>else if age > 21<br>→ LearnerIsAdult<br>else if age > 14<br>→ LearnerIsAdolescent<br>else if age > 6<br>→ LearnerIsKid<br>else if age > 3<br>→ LearnerIsChild<br>else<br>→ LearnerIsBaby | LearnerIsSenior<br>LearnerIsAdult<br>LearnerIsAdolescent<br>LearnerIsKid<br>LearnerIsChild<br>LearnerIsBaby |
| 40 | eqfLevel = Number specifying the learner's EQF level from the learner profile or from TUG | if eqfLevel == 1<br>→ LearnerEqfLvl1<br>else if eqfLevel == 2<br>→ LearnerEqfLvl2<br>else if eqfLevel == 3<br>→ LearnerEqfLvl3 | LearnerEqfLvl1<br>LearnerEqfLvl2<br>LearnerEqfLvl3<br>…<br>LearnerEqfLvln |

| | | | |
|---|---|---|---|
| | | …<br>else if eqfLevel == n<br>→ LearnerEqfLvln | |
| 41 | level = Statement about the learner's level of knowledge regarding the course content as declared in the learner's profile or as stated via TUG | if lvl == Beginner<br>→ LearnerIsBeginner<br>else if lvl == Intermediate<br>→ LearnerIsIntermediate<br>else if lvl == Advanced<br>→ LearnerIsAdvanced | LearnerIsBeginner<br>LearnerIsIntermediate<br>LearnerIsAdvanced |
| 42 | width = width of the screen of the learner's device in pixel<br>height = height of the screen of the learner learner's device in pixel | if width * height > 2073600<br>→ ScreenResolutionXL<br>else if width * height > 921600<br>→ ScreenResolutionL<br>else if width * height > 589824<br>→ ScreenResolutionM<br>else if width * height > 409920<br>→ ScreenResolutionS<br>else<br>→ ScreenResolutionXS | ScreenResolutionXS<br>ScreenResolutionS<br>ScreenResolutionM<br>ScreenResolutionL<br>ScreenResolutionXL |
| 43 | width = width of the screen of the learner's device in pixel<br>height = height of the screen of the learner learner's device in pixel<br>diagInch = diagonal size of the display in inches | diagPix = Sqrt ( $w^2 + h^2$ )<br>ppi = diagPix / diagInch<br>if ppi > 480<br>→ PpiExtraExtraHigh<br>else if ppi > 320<br>→ PpiExtraHigh<br>else if ppi > 240<br>→ PpiHigh<br>else if ppi > 160<br>→ PpiMedium<br>else if ppi > 120<br>→ PpiLow | PpiExtraExtraHigh<br>PpiExtraHigh<br>PpiHigh<br>PpiMedium<br>PpiLow |
| 44 | dConType = Type of connection used to access the LMS<br>dConStab = Number specifying the stability of the internet connection | if dConType == wire \|\| ((dConType == LTE \|\| dConType == HSDPA) && dConStab > 80 )<br>→ ConnectivityGood<br>else if dConStab > 60 \|\| ((dConType == UMTS \|\| dConType == Edge) && dConStab > 40) | ConnectivityGood<br>ConnectivityOk<br>ConnectivityPoor |

| | | | |
|---|---|---|---|
| | | → ConnectivityOk<br>else<br>→ ConnectivityPoor | |
| 45 | noise = Current noise level of the learner's environment in db | if noise > 120<br>→ EnvTooLoud<br>else if noise > 70<br>→ EnvLoud<br>else noise > 50<br>→ EnvGood<br>else<br>→ EnvQuiet | EnvTooLoud<br>EnvLoud<br>EnvGood<br>EnvQuiet |
| 46 | onCampus = Boolean value if learner currently is on campus as declared via a TUG dialogue<br>isIndoors = Boolean value if learner currently is indoors as declared via a TUG dialogue | if onCampus && isIndoors<br>→ OnCampusIndoors<br>else if !onCampus && isIndoors<br>→ OutOfCampusIndoors<br>else if onCampus && !isIndoors<br>→ OnCampusOutdoors<br>else<br>→ OutOfCampusOutdoors | OnCampusIndoors<br>OutOfCampusIndoors<br>OnCampusOutdoors<br>OutOfCampusOutdoors |
| 47 | battery = Current battery status of the device the learner uses to access the LMS | if battery > 20%<br>→ BatteryOk<br>else if battery > 10%<br>→ BatteryLow<br>else<br>→ BatteryVeryLow | BatteryOk<br>BatteryLow<br>BatteryVeryLow |

Table 7: Specification of Didactic Factor transformation rules

# 9 LPM as a Software Module

This chapter describes the LPM modules (cf. section 3.6) in detail and from a technical perspective. It can be seen as a guideline for the programmers of the LPM on what is expected from the implementation of these modules. Each of the following sections contains an overview of the principal ideas and functionalities of each module. From a technical perspective, it should suffice to read only this chapter instead of the whole deliverable.

## 9.1 Session Manager

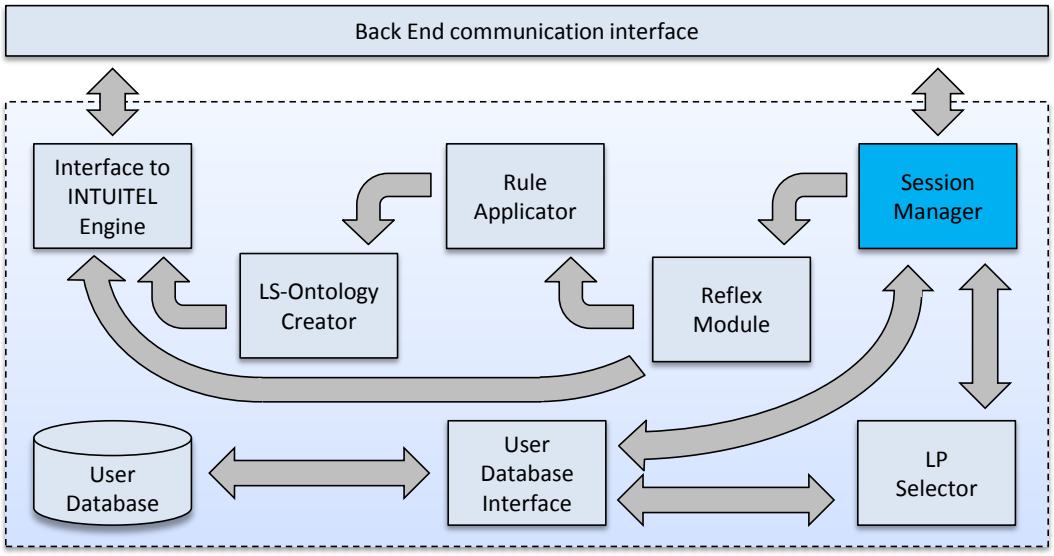


Figure 22: Session Manager in the LPM architecture

The Session Manager is the entity in the LPM that administers the information from other INTUITEL components via the Back End communication interface.

As a managing entity it is charged with the collection, processing and forwarding of all incoming information. It also requests information from the SLOM metadata repository and uses the LMS USE-service to provide a sufficient data basis for the recommendation creation process. Its main purpose is to monitor the LO transitions of learners via the Learner Update service (specified in context of the INTUITEL Data Model - D1.1).

This mechanism is available in three different versions that differentiate between the LPM and the LMS being the initiator of the update message. The Session Manager is thus not only passively receiving the information from the LMS, but could also actively request it in this context. Which procedure shall be used is specified by the LPM in the initialization service which specifies the configuration that has to be applied for the particular LMS.

Via the learner updates mechanism of the LMS USE-service the Session Manager is able to determine who opens which KO and when. This makes it possible for INTUITEL to detect when a learner needs a

new recommendation. In general, this is the case when SLOM metadata is available for a LO that a learner just accessed. If this is the case, the Session Manager triggers the LP Selector (cf. section 8.2) and starts to collect all necessary information (e.g. from the User Database – cf. section 8.3). The LP Selector provides the Session Manager with the information which LP has been chosen or, if not already set, how the system has to react in order to determine a suitable one. All this information is then handed over to the Reflex Module (cf. 8.4), which represents the first step in the recommendation creation process.

## 9.2 Learning Pathway Selector

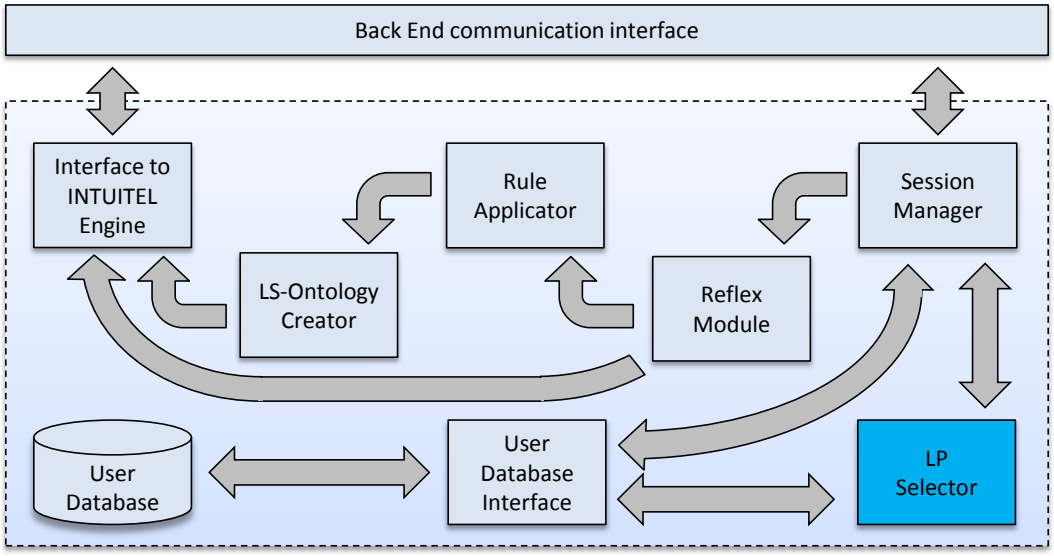


Figure 23: Learning Pathway Selector in the LPM architecture

The Learning Pathway Selector component manages the Learning Pathway (LP) related issues in the LPM. It has two main tasks: (1) the initial selection of a suitable LP, and (2) the refinement of LP selections.

(1) If a LP is yet undefined, the LP Selector can choose between different possibilities to assign a LP to the learner:

   a. Assign a LP if no options are available (only a single LP is specified). This is less desired because it does not include any measures regarding what is didactically fitting for the learner. However, there might be cases that only provide a single LP, which makes the selection of this particular one inevitable.

   b. Find a matching LP by evaluating the LO history and LPs of other participants. The best match could be simply the most popular LP. As a possible optimization technique to increase the LP-suitability, the individual learning styles of the learners could be matched to identify the learners that are most similar to the learner in scope. When then

determining the most popular LP, the result should be more fitting. This can be seen as learner profile analytics[18].

c. Ask the user via TUG:

   i. Unguided questioning. Given that the learners are ranked as didactically experienced, a question could be generated that directly asks for the method they want to use. Although this might result in the most fitting LP, this approach can be problematic because it is difficult to generally determine what kind of learner INTUITEL addresses and also if this learner actually knows enough about these LPs to make a profound decision.
   ii. Guided questioning. A more learner-friendly solution is to ask the learner a directed question. One such question could be to ask the learners in which country they have gone to school. Since the school systems across the world usually prefer certain didactic approaches, it is possible to assume that a learner will be more familiar with the LP that is preferred in this particular country. However, this is likewise (but less) problematic because this is only a probability measure that, in order to be applicable, firstly needs the respective data for all countries that come into question. Furthermore, the popularity of a LP does not necessarily mean that it is actually suitable for the particular learner. It is thus probable that a set of questions might be used to also include aspects regarding the personal learning habits.
   iii. The easiest and probably also most intuitive solution is to combine the LP selection with the actual content. This can be done by preselecting a set of LPs and to determine the first KOs on these paths. By then asking the learners with which element they want to start, INTUITEL can indirectly let them chose their LP.

All these possibilities lack certainty regarding the accuracy of their results. They are rather heuristics than actual measures and the initially chosen LPs should thus be refined in the learning process, which is the second task of the LP Selector.

(2) Refinement of the selected LPs. With the help of the learner's LO history is the LPM able to track the LP that the learner has actually chosen in the course. This allows the LP Selector to analyse the learning behaviour to ultimately optimise the learning process by determining another, more fitting LP for each individual learner. This can be carried out on different ways:

   a. Ask the learners after a certain amount of (learning) time or after a certain number of learning steps whether they are satisfied with their Learning Pathway or not. If the latter is the case, further questions can be asked to determine which LP would be more fitting or to automatically choose another LP on basis of the satisfaction level of the learner (e.g. the lower the satisfaction, the more different is the next selected LP).

[18] Educational data mining, i.e. learner profile analytics is not part of INTUITEL and beyond the scope of the project.

b. An algorithm to determine the distance to certain LP has been described in section 5.3 of this paper. While other approaches, like e.g. a generalized edit distance measure (e.g. Levensthein distance) could be used to determine the distance between Learning Pathways, they have been revoked on the basis of a careful study of different LP. The algorithm in section 5.3 will act on a purely local basis in the cognitive space to determine the locally best LP. INTUITEL will then either assign the learner this particular LP or directly ask whether a LP change is desired.

## 9.3 User Database (Interface)

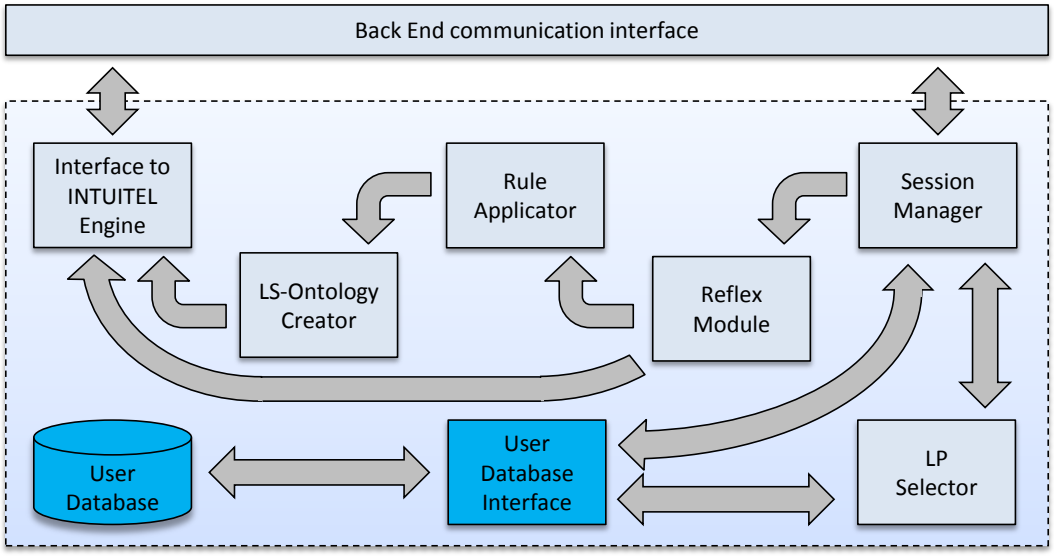


Figure 24: User Database (Interface) in the LPM architecture

The User Database is the place where the LPM stores all user relevant information for all learners persistently across different learning steps and sessions.

The User Database Interface is part of the data access layer and acts as a mediator between the user database and the Session Manager and the LP Selector. It decouples the database technology from the actual implementation of the other components. The queries for commonly and frequently requested data items will be predefined and encapsulated in ease-to-use methods to make a lightweight implementation of transformation rules possible.

One of the main input types of the LPM is the so called pre-processed input (cf. section 4.4). This subsumes all the knowledge that INTUITEL collects during its runtime regarding each particular learner and mainly includes four different aspects which are detailed in the rest of this section:

- User model
- User learning history
- User sessions
- User learning pathways

The **user model** is the representation of the static data about the learner like, for example, name, gender and date of birth. Typically, this is the data that an LMS collects regarding the user profiles. Apart from that, the user model also includes the previously deduced (quasi-) static information about the learner. It remains to be seen what data items are qualified for this. The foundational idea is that there are some statements about learner characteristics that don't need to be updated every learning step. If, for instance, a Didactic Factor evaluates the learning history every 20 learning steps to state that a learner works methodological or not, this needs to be stored across different sessions.

The **learning history** represents the actual learning path that each learner has taken per session and per course. This information is necessary to calculate values that need a number of past LOs to create certain statements. It is therefore necessary to not only store the order of LOs but also the access and leaving times as well as the scores and completion levels. Likewise relevant are the recommendations that INTUITEL made for the respective learning steps. This way, the Back End could evaluate at which points the learner deviated from the LO suggestions and might be able to include conclusions in upcoming recommendations. Although it might be possible that some of this information is also persistently available in the LMS, it must be stored in the LPM too. This is a necessity because it is not guaranteed that each LMS stores the information (even if it is capable of doing so) and to keep the computation times as low as possible. If the LO history would have to be requested each time (from the LMS), the processing duration would increase dramatically.

**User session data** is of importance to also include past learning sessions in the recommendation process. With this information, it is, for instance, possible to evaluate if there are correlations between learning success, used devices and media types for a certain learner. This can then be used to deduce better (i.e. better fitting) and more personalized recommendations. Thus the session related data must contain all information that is consistent for one session (e.g. used device) and that is relevant for Didactic Factors.

**User learning pathways**: in order to find the most fitting recommendations, INTUITEL relies on predefined Learning Pathways that are assigned to each learner individually. The Back End thus always needs the information which learner uses which macro and micro LP. Furthermore, to also include historic LP-based information, the LP-switches with their times and LO-references need to be stored persistently.

## 9.4 Reflex Module

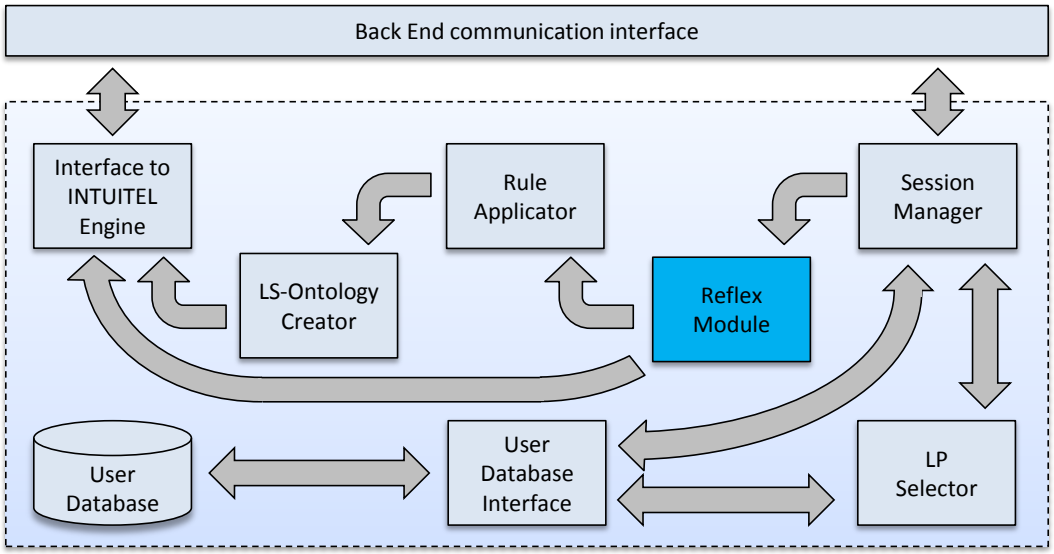


Figure 25: Reflex Module in the LPM architecture

The Reflex Module is the first component in the recommendation creation workflow after the Session Manager has triggered the process. It acts as a first rough filter to decide if the complex and computationally expensive reasoning can be avoided at an early stage (cf. section3.5).

This can be seen as a first optimization technique. The calculation of Didactic Factors, the creation of the LS-Ontology and of course the reasoning process in the INTUITEL Engine need time and performance. It is thus advisable to shorten the procedure when it is obvious that it will not come to a (qualitative) conclusion.

This is where the Reflex Module comes into play to reduce the load on the servers and to optimize reaction times. It contains a set of rules that test if the available data is sufficient to perform the recommendation creation process.

If this is not the case, the Engine is contacted directly to trigger the creation of TUG-messages that explains why the process has been aborted. Where possible, these messages will contain questions to ask the user about missing or additional information.

However, the Reflex Module does not necessarily have to abort the recommendation process. It can also trigger the creation of the TUG messages in parallel to the reasoning process. This makes it possible to detach the TUG message creation from the reasoning process, ultimately making it more lightweight.

## 9.5 Rule Applicator

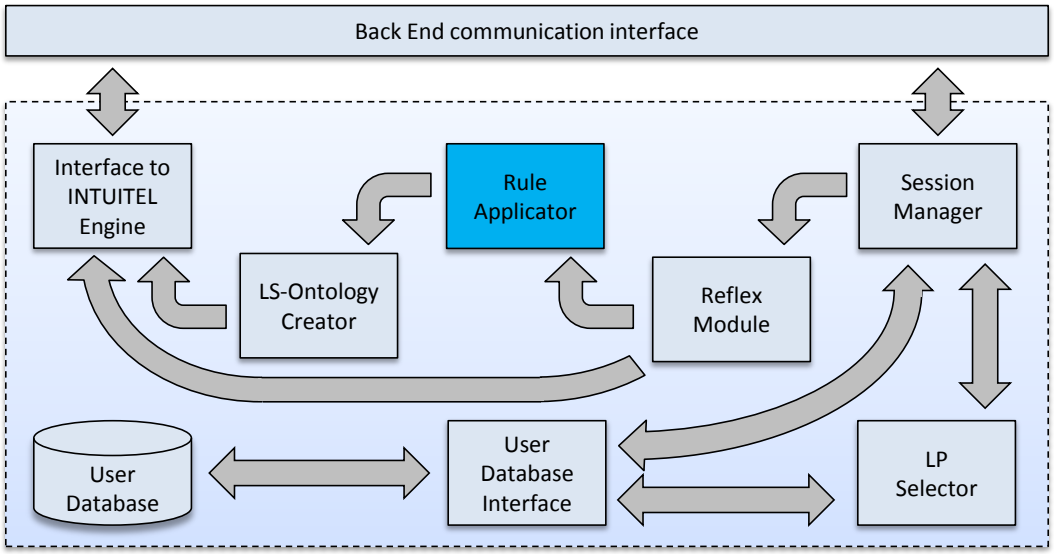


Figure 26: Rule Applicator in the LPM architecture

The task of the Rule Applicator is the application of the transformation rules of the Didactic Factors that are part of the LM Ontology. It consists of the three sub-components OWL Reader, Rules Framework and Rules Processor.

**OWL Reader.** Extracts the definitions of the Didactical Factors from the OWL LM Ontology and map them to the transformation rules in the LPM source code. Hence the LM Ontology indirectly specifies which rule should be applied. Users can easily customize their INTUITEL instance by changing the references in the DF definitions or by just removing them. This feature makes it very easy for the users and administrators to see which measures are used for the determination of the suitability of the Learning Objects.

**Rules Framework.** The Rule Applicator also needs a framework to standardly implement Didactic Factors. This is necessary to manage them in a generic way. It thereby has to be determined which methods this class/interface set needs in order to include all relevant methods to apply each Didactic Factor. In the most basic configuration, this is a constructor-method and a second one to start the evaluation. The important question regarding the implementation is how exactly the LM Ontology will specify DFs regarding the ranked (e.g. slow vs. medium vs. fast) definition of Didactic Factors on an OWL basis. This will have to be regarded in the specification of this framework.

**Rules Processor.** Lastly, a system to automatically apply these rules needs to be designed. This basically represents an internal management component that controls the extraction of the DF information from the LM ontology and the application of the transformation rules. It also collects the respective results and forwards them and the remaining information stack (e.g. the SLOM metadata) to the LS-Ontology Creator in a suitable way.

## 9.6 Learner-State-Ontology Creator

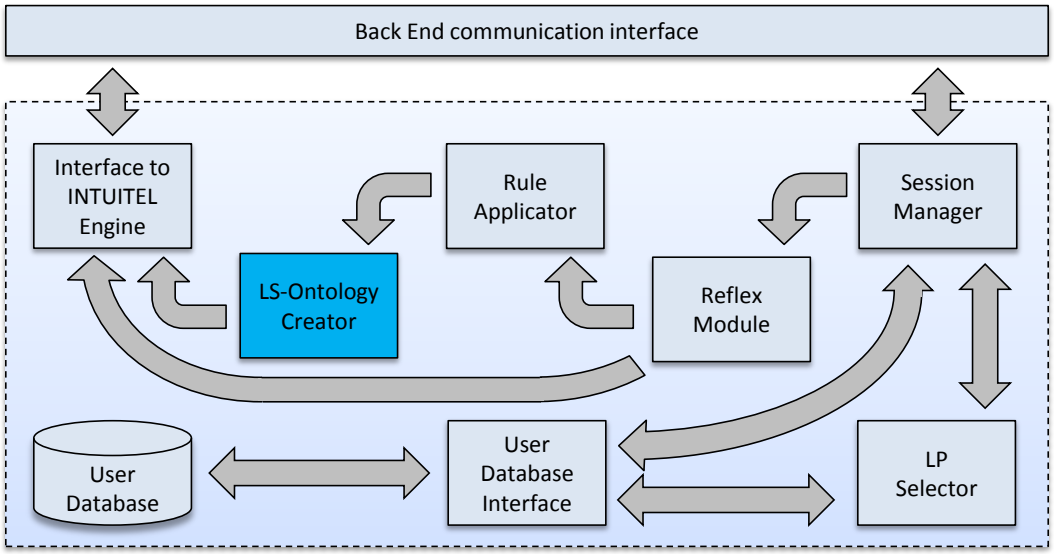


Figure 27: Learner-State-Ontology Creator in the LPM architecture

The LS-Ontology Creator is the last module in the recommendation creation workflow in the LPM that puts all information pieces together to form an ontology which is used by the INTUITEL Engine as input. For a detailed description of the LS-Ontology cf. section 7.3.

There is also another important functionality that is concerned with attaching information to LOs in the SLOM metadata. This is very similar to the approach taken with the Didactic Factors, but with the difference that DFs are generalized statements about the learning habits, styles and history of the learners. In contrast, the information that is added in context of the LS-Ontology creation is LO-specific. This is a must-have feature because the general concept of the INTUITEL Back End requires the availability of the completion states of LOs and the marking of current/active LOs in the INTUITEL Engine in order to follow the Learning Pathways of the learner.

How similar this feature will be to the Didactic Factor transformation rules will be a matter of the relevance and quantity of such LO-specific additions. INTUITEL currently only foresees the two statements mentioned above, but there might come up further ones when the LM Ontology, SLOM and the INTUITEL Engine tasks progress. It is thus possible that this feature will be extended or even integrated into the Rule Applicator, to take the importance of this functionality into account.

## 9.7 Communication Interfaces

The basic purpose of the communication interfaces is to enable a fast and easy data exchange between the internal software components of the LPM and the other INTUITEL modules.

In general, the INTUITEL system[19] can be operated on a single server instance or in a cluster and with a varying range of functions of the communication layer. To hide this level of complexity from the

[19] Here, the INTUITEL system means the INTUITEL endpoints excluding the LMS.

internal components of the LPM, the communication interfaces encapsulate the technical aspects of the communication.

Basically, there are two communication interfaces: the Back End Communication Interface to exchange data with other INTUITEL components (LMS, SLOM etc.), and the interface to the INTUITEL Engine.

### 9.7.1 Back End Communication Interface

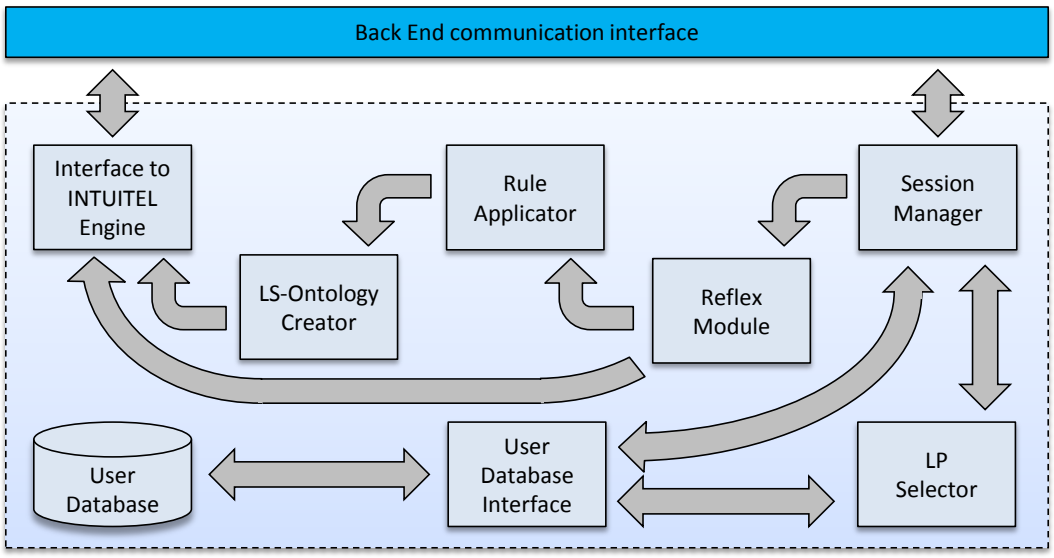


Figure 28: Back End communication interface in the LPM architecture

The Back End Communication Interface manages the technical aspects of the data exchange between the LPM and the other INTUITEL components.

It relies on a library (developed in context of task 3.3) developed by INTUITEL partner TIE Nederland b.v, which supports both operations modes of the communication layer (basic and advanced).

As INTUITEL operators can choose how to setup the system (i.e. one or multiple servers), they can also choose between the two versions for data transmission:

1) Basic: INTUITEL endpoints will communicate via REST web services only.
2) Advanced: Messages are managed by the TIE Smart Bridge (TSB) that combines REST and XMPP as communication technology. It will also offer additional features like group messages to support a more redundant and performant setup of INTUITEL endpoints.

For more information on that topic, please see working document 3.1, which describes these two versions and the communication layer in more detail.

Independent of the selected operation mode and server-structure the payload between the endpoints is identical. This has already been outlined in the INTUITEL Data Model (deliverable 1.1) for the messages between the Back End and the LMS. The data schema between the LPM and the

INTUITEL Engine or the SLOM metadata repository has not yet been defined and will follow when the respective endpoint development progresses. This will probably cover three types of messages:

1) The transmission of the LS-Ontology from the LMS to the INTUITEL Engine.
2) A message from the LMS to the INTUITEL Engine to specify which kind of TUG message should be created and what the respective information is.
3) Request and transfer of SLOM metadata.

All three have a direct relevance for the LPM and a specification for the payload format needs to be established in collaboration with the respective work packages and tasks.

### 9.7.2 Interface to INTUITEL Engine

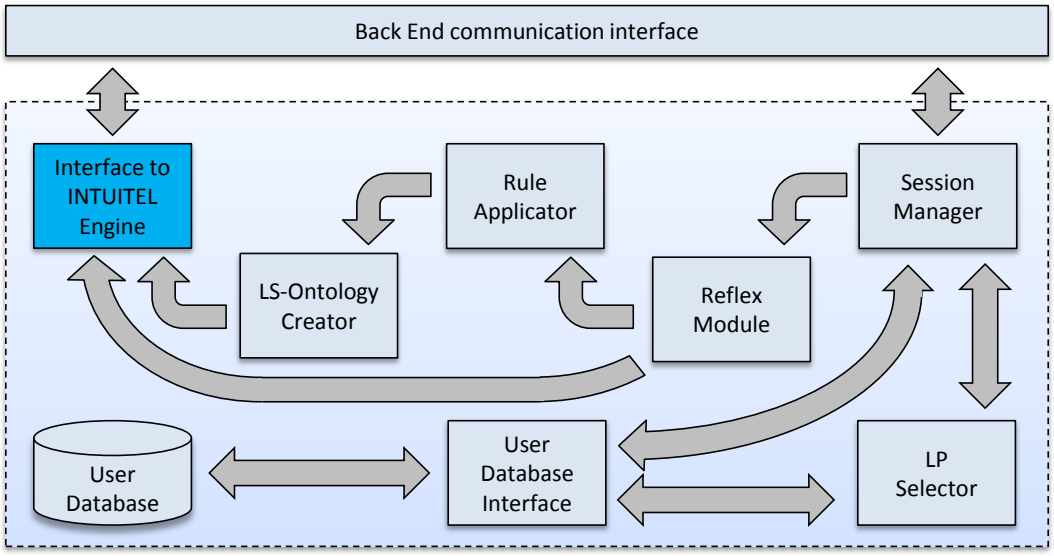


Figure 29: Interface to INTUITEL Engine in the LPM architecture

The 'Interface to INTUITEL Engine' (task 3.4) connects the LPM with the INTUITEL Engine. Its main purpose is the testing of the internal and external dependencies of the programmatic realization. This includes the verification of the connection between those two components and also the evaluation of the score extraction, messaging and recommendations on the LMS. It will therefore define a set of mock-modules to test the individual cases without having the need to use the respective counterparts. This makes it possible to test the different functionalities without the danger to include errors from modules that are currently not in the testing focus.

The secondary functionality of this interface is to forward the calculated information in an INTUITEL Engine conformant format to the Back End Communication Interface. The LS-Ontology will probably and the TUG-message-triggers will certainly have to be integrated into a specific format to transmit the information as payload, which consequently induces that one module needs to implement this functionality. A central place to form those messages is this very module. This also fits nicely to its main task of testing. Since similar methods are needed to create the testing payload, the know-how of creating them can be applied or reused here.

# 10 Summary

In this document, we described in detail the approach of the INTUITEL Back End to guide learners, but also the connection of the Back end with the other INTUITEL endpoints. This provides an introduction into the overall system that has been designed by the INTUITEL Consortium to combine the semantic description of Learning Pathways and metadata as factors to determine the didactically most suitable Learning Objects for individual learners. The interplay of well-known technologies (Java, OWL and machine based reasoning) forms a powerful yet easily extendable intelligent tutoring system.

The Learning Progress Model (LPM) is the central information mediating component which aggregates information from the Learning Management Systems (LMS), the data storage, the Semantic Learning Object Model (SLOM) and the Cognitive Models (CMs). It processes all that information to achieve the primary INTUITEL objective - learner-specific feedback and recommendations. The LPM reduces the complexity and dimensionality of the input data and transforms it to be usable in context of the reasoning process.

We therefore formulated a framework with an applicable set of transformation rules as basis for the feedback and recommendation process. This has been carried out in regard of the specification of the Back End and LPM concept, which introduces the technical approach of the INTUITEL system, which is also considered as the theoretical and semantic foundation of the LM ontology. Secondary aspects are concerned with the formulation of the Multidimensional Cognitive Space the learner is contextually located in, the actual conceptualization of the LPM and its internal information processing components as well as the description of the interplay of the LPM with the other INTUITEL components.

The core of the INTUITEL recommendation and feedback concept is to interpret the act of *teaching* (but not the process of *learning*) from a computer science perspective as a processing automaton with certain input values, intricate transformational mechanics and processed output.

To implement the LPM for its use in INTUITEL this document also describes the LPM from a more technical side. It summarizes the objectives of the individual components and thus allows an easier implementation of them in form of a joint LPM software module.

In this work, several concepts originating in information sciences have been transported into the domain of educational sciences. Cognitive Space, Rating Factors and Didactic Factors and their transformation may be considered a novel model of the Technology Enhanced Learning process. These concepts and ideas of the LPM are, in the view of the consortium, relevant beyond their direct implementation in the INTUITEL systems. It is therefore intended to publish them as major research accomplishments of the project.