Serdica J. Computing **14** (2020), No 3–4, 113–144

Serdica
Journal of Computing

Bulgarian Academy of Sciences
Institute of Mathematics and Informatics

IMPROVING THE PRODUCTIVITY OF THE E-LEARNING PROCESS

Oleg Iliev

ABSTRACT. The e-Learning systems have become extremely popular over the last decade. Moreover, not a long time ago they were considered as an alternative way of representing learning content. However, suddenly due to the COVID-19 pandemic they turned out to be the only possible option for keeping the learning process uninterrupted. This makes any research on possible ways of optimizing the presented learning content and its better absorption by the learners, as this paper, extremely important. The paper considers the possibilities for improving the productivity of the e-Learning process in four main areas: (1) reusability of learning content; (2) personalized representation of learning content; (3) proper identification of participants in an e-Learning process; (4) opportunities for easy scaling of learning environments. As a result of the research, a set of methods and models for creating personalized learning materials has been developed. The learning materials are aligned

ACM Computing Classification System (1998): K.3.1, K.3.2.

Mathematics Subject Classification (2020): 97U50, 68U01.

Key words: e-learning, software development, thematic-oriented content, reusability of a learning content, personalized representation of a learning content, user identification.

This paper presents the principal results of the doctoral thesis "Methods and models for personalization of a thematic-oriented learning content" by Oleg Iliev (Laboratory of Telematics—BAS), successfully defended before a Scientific Jury on 25 January, 2021.

to the learner's preferred learning style and created from a thematic-oriented content. All the conceptual models were then implemented into a software environment, which provides an opportunity for their validation and verification and assessment of their effectiveness. The research also presents a so-called "concept for software scaling" from the perspective of an e-Learning environment and a novel software architecture to be used as a base of a system implementation.

1. Introduction. e-Learning environments provide an opportunity for everyone to become a participant in a learning process. He or she could be both a teacher or a student. Moreover, they often offer large databases of learning content. Although at first glance the benefits of this type of alternative to the standard way of learning – in a classroom, with one teacher and many students – seem much more than the disadvantages, it is the disadvantages and opportunities to overcome them that are of interest in this study. For example, because each of the participants in the learning process can be involved remotely, and also because each user of the learning systems can be both a teacher or a student, it is very important to provide a reliable way to verify the user's identity. At the same time, e-Learning environments are a kind of software solutions, which predisposes them to be limited by purely software principles. For example, if the software environments are not designed properly, even if they contain a lot of quality content and good opportunities for its presentation, they could reach the threshold of their development and usage, where it will not be possible to include new participants or new features.

Guided by the idea to overcome the software-related challenges of the e-Learning system, this study developed a special "concept for software scaling" for software systems, as well as a novel software architecture specifically designed to develop a e-Learning system. At the same time, the research work is not limited to the possibilities for software optimization, but uses the fact that in an e-Learning system the presentation of the learning process is provided by a software, so it could be flexible by its nature. The paper describes a number of models and methods for optimizing the presented learning content. The optimization was achieved by adapting two well-known models for managing learning processes – Bloom's Taxonomy [1] and Honey and Mumford's Learning Styles [2]. Based on these models, a new method for creating learning content has been designed. Thanks to the opportunities provided by the software technologies, such a model can be implemented using a variety of software techniques.

While Bloom's Taxonomy looks at the different cognitive processes that take place in learners as they acquire new knowledge, as well as ways to provoke these processes, Honey and Mumford's Learning Styles look at different types of learners and how they perceive learning content differently. A symbiosis between these two models has been proposed in this paper so that we can talk not about a process of acquiring knowledge, but about the ways to structure and create a learning content. As a result, a model has been designed. The model allows small pieces of learning content, called "information objects" according to Wagner's "Learning Content Model" [3], to be arranged in a certain order and thus not only to generate meaningful learning material, but also for the ordering of these pieces to be made in the most optimal way for each learning style. This provides an opportunity to create personalized materials according to the preferred learning style of the learners.

The developed conceptual models and methods for generating personalized learning content have been implemented in a software environment, through which the validation and verification of the models has been done, and their effectiveness has also been studied.

2. Overview of the Existing Methods and Models for Management and Analysis of Knowledge Acquisition Processes.

2.1. Bloom's Taxonomy. Bloom's Taxonomy is a hierarchy of cognitive skills in which higher levels of thinking include all cognitive skills from lower levels. It is designed to enable teachers to classify a learning goal and to define and rank learning objectives [1].

According to the Bloom's theory the goals and outcomes of learning are not the same. For example, memorizing scientific facts, no matter how important they are, is at a lower level than the ability to analyze or evaluate. To apply a concept, you must first understand it. To evaluate a process, you must first analyze it. Each subsequent level is upgraded over the previous one. This structuring enables learners to learn the lesson in many ways and to perceive information in different ways.

Teaching the lesson in this way allows learners to assimilate information step by step according to their learning styles and individual abilities and to progress towards their level.

Over the years the understanding of the features of the model grows, so the Bloom's Taxonomy has been further modified (Fig. 1). The update consists of several seemingly small but important changes in terminology and structure. With the update, verbs have been added to the model, as well as key questions

that will provoke some kind of mental activity in the learner in order to achieve a specific learning goal [4].

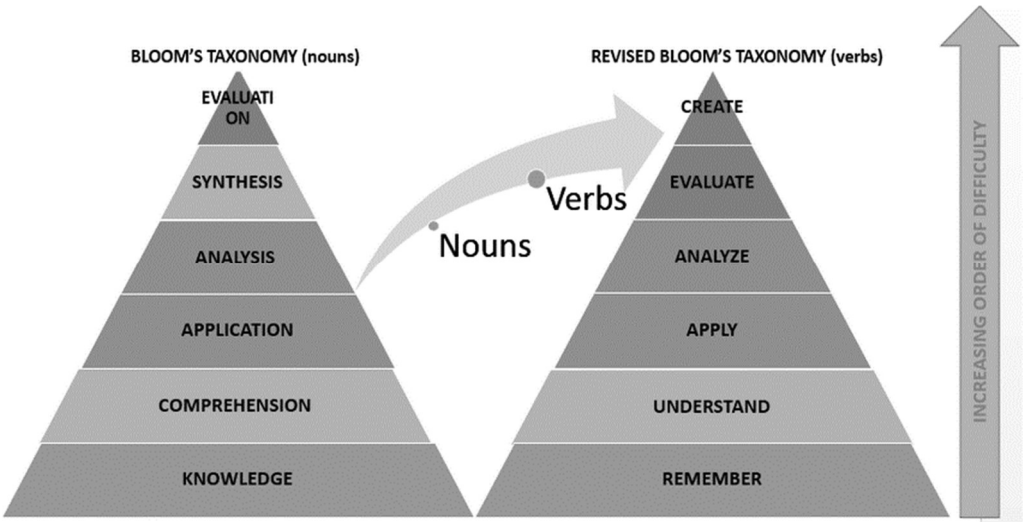

Fig. 1. Comparison between the original and the updated version of Bloom's Taxonomy

In Bulgaria, the taxonomy is strongly represented in education. An example of work on the topic is the article by Hristina Kostadinova, Georgi Totkov, Mariana Raykova, who developed a method for "Automated test generation" [5].

2.2. Learning Styles – Learning by Experience. Learning styles are characterized by different methods of assimilation, organization and understanding of information by individuals. They do not deal with the individual abilities or the level of intelligence of the learners, but aim to turn complex tasks into easy-looking ones, simply by adapting the method of presenting knowledge.

Research and practice in the field of learning show that learning can be improved when the learning process adapts to different learning styles of learners.

In Bulgaria, there are various researches on the application of the learning cycles according to Kolb [6] and Honey and Mumford. For example, Yuri Klisaron developed a "scale for assessing the learning style" in his work from 2013 [7].

Although Honey and Mumford's [2] theory is very close to Kolb's one, there are some differences between the two models. For example, Honey and Mumford found that Kolb's Learning Styles Inventory (LSI) was not always accurate [8]. Kolb takes for granted the objective determination of learning styles by the people themselves – everyone should determine the appropriate learning style for himself. At the same time, Honey and Mumford are proposing a so-called Learning Styles Questionnaire (LSQ), as they believe that people have never thought about how they perceive new knowledge. With the help of this questionnaire, people can determine accurately the appropriate learning style for them [9].

The learning cycle, according to Honey and Mumford, is divided into four quadrants (Fig. 2). Depending on the learning style – which quadrant the learner falls into, the learner should start from a different stage of the cycle, but the idea is that he or she will always go through all four quadrants [10].

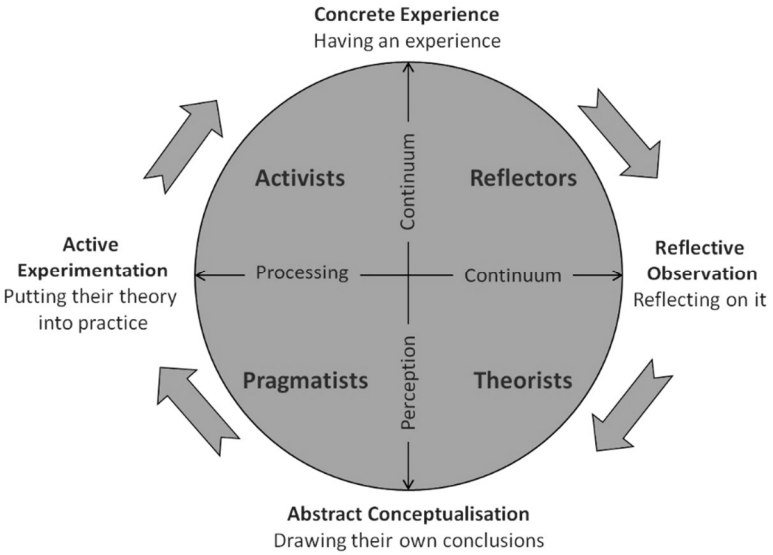

Fig. 2. Honey and Mumford's Learning Cycle

In addition to the typical approach to individual learning, Honey and Mumford propose solutions for how the learner should interact with others, thus making the learning process cooperative between individuals.

Quadrants defining Honey and Mumford's learning styles:

1. Activists (Concrete Experience). Activists are people who learn

by doing something. They love new experiences and prefer to take action before considering the aftermath. This type of people begins the learning cycle from the Concrete Experience stage.
They learn best when:

- Learning goes through a new experience, problem or opportunity
- "They are thrown into the deep"
- Work with other people, solve problems together, play games
- Have the opportunity to lead a group

They learn worst when:

- Listen to lectures or have to read long descriptions
- Read, write and think by themselves
- Analyze and interpret a lot of data
- Precise instructions follow

2. Reflectors (Reflective Observation). Reflectors learn by watching and thinking about what is happening. They like to consider the experience in detail. They are usually more careful than activists. While for activists the experience comes first and the evaluation comes second, for the reflectors the experience should be short and then given enough time to think. This type of people starts the learning cycle from the stage of Reflective Observation.

Reflectors consider all possible situations and consequences before making a decision. They spend their time listening and observing and are usually careful and cautious.
They learn best when:

- They have the opportunity to stand behind and watch first
- They are given time to think and reflect before commenting or acting
- Have the opportunity to rethink what happened
- Perform tasks that do not include deadlines

They learn worst when:

- They are forced to lead a group
- Take action without preparation
- They are pressed by deadlines

3. Theorists (Abstract Conceptualization). Theorists like to understand the theory on which actions are based. They need models, concepts and facts to learn. They like to analyze and synthesize, they feel

uncomfortable in the presence of subjective judgments. It is typical for this type of people to use their observations and experiences in logical, conceptual frameworks. They want to know how and why something happens a certain way. They want to get all the details first and only then move on. In this line of thinking, they are the complete opposite of activists who cannot delay the start of a task. This type of people begins the learning cycle from the Abstract Conceptualization stage.

They learn best when:

- The action is lined with ideas and concepts that form a model, system or theory
- In a structured situation with a clear task
- They have a chance to ask questions and do research
- Understanding of a complex situation is required

They learn worst when:

- They are in a situation that requires emotions and feelings
- Actions are unstructured and ambiguous
- They are required to take action without knowing the basic principles and concepts on which the action is based

4. Pragmatists (Active Experimentation). Pragmatists like to try new theoretical ideas, but prefer to actions into a simulated environment before moving on to the actual action. They are experimenters. They tend to be very practical people who can make a connection between theory and practice, but they want to be sure by experimenting that their ideas are right before taking on the task. They love open discussions, with no clear end. They are impatient when starting an action. This type of people starts the learning cycle from the Active Experimentation stage.

They learn best when:

- There is a clear and direct link between the topic and the current need
- They are presented with knowledge with a clear practical focus
- They can try things and get feedback from an expert
- Can copy an example or emulate a role model

2.3. Wagner's Learning Content Model. The learning content model illustrates the concept of assembling content into higher-level objects. Learning objects are composed of information objects that are built in a hierarchical structure and thus form collections of information to create

courses or entire curricula. This model describes the granularity of learning objects and presents its extreme importance when it comes to reusing content.

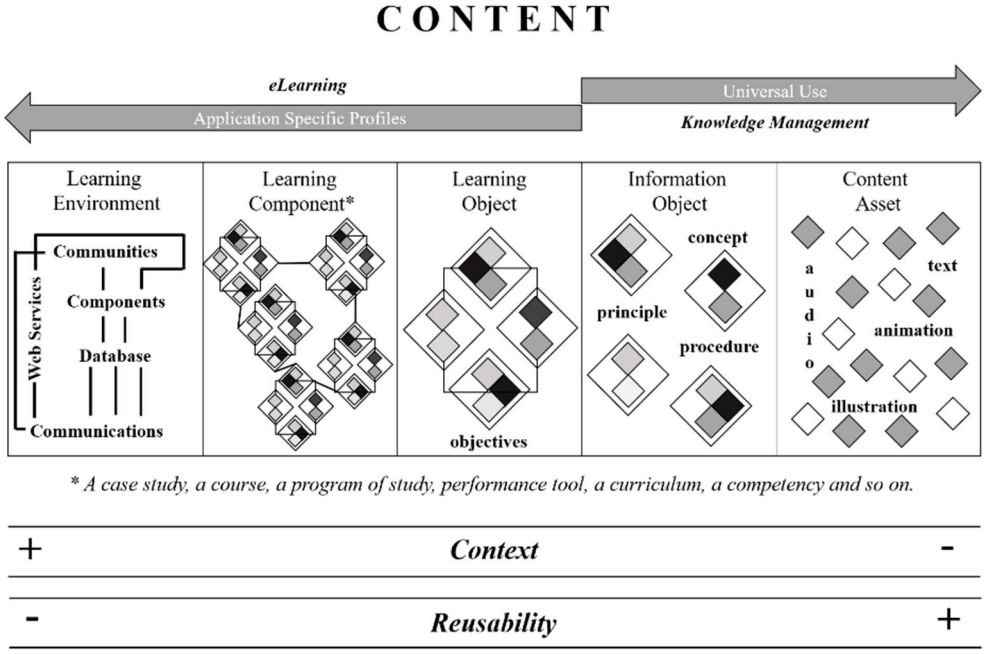

Fig. 3. Wagner's Learning Content Model

According to Wagner [9], the main components illustrated in Fig. 3 of the learning content model are the following.

- **Content Asset:** Content assets include raw media, such as images, clippings, audio and video clips, and more.
- **Information Object:** A text passage, web page, and others that focus on a single piece of information. Such a piece can explain a concept, illustrate a principle, or describe a process.
- **Learning Object:** In the learning content model, the learning object is a collection of information objects that are assembled together in order to meet one learning goal.
- **Learning Component:** The learning component is a basic concept for things like lessons or courses that are related in order to meet multiple

learning objectives at a higher level. They are a combination of several learning objects.

- **Learning Environment:** The learning environment is a combination of learning content and technology with which the learner interacts. The combination of learning components with communication tools and/or other functionalities aimed at providing online learning experience can be aggregated in a learning environment, such as a learning management system (LMS), e.g.

It is generally accepted that there is a connection between the size of the educational object and the possibility of its reuse. This can also be seen at Fig. 3. Well-granulated learning objects and components have the potential to be reused and assembled into new learning objects, while whole courses are not suitable for use in different contexts.

The learning objects should be granulated into small independent pieces that can be used alone or in combination with other materials in order to form higher level objects and meet the needs of the user. The fundamental idea of the learning objects is for the lesson designer to create small components that can be reused many times in different learning contexts. Many publications claim that reuse not only saves money and time to the teachers, but also improve the quality of learning materials [11] [12].

As with LEGO blocks, the idea is to create something small that can be complete on its own, but also easily combined with other components [13]. Learning objects should follow the rule that each unit should do only one thing and minimize the connection with other units. There is a general consensus that the learning object should be **Reused, Accessible, Compatible, Sustainable** [14].

2.4. Process of Perceiving New Knowledge According to the Cognitive Processes (Bloom's Taxonomy) by People with Different Learning Styles (Kolb's Learning Styles). In his paper, James Gallagher [16] suggest that there is a correlation between Kolb's Learning styles and Bloom's Taxonomy. He argues that different learning styles should go through all levels of the learning process defined by Bloom, but in different sequences. The principle of traversing the levels in a clockwise sequence is used, and a specific level suitable for the different learning styles is used to start the cycle (Fig. 4).

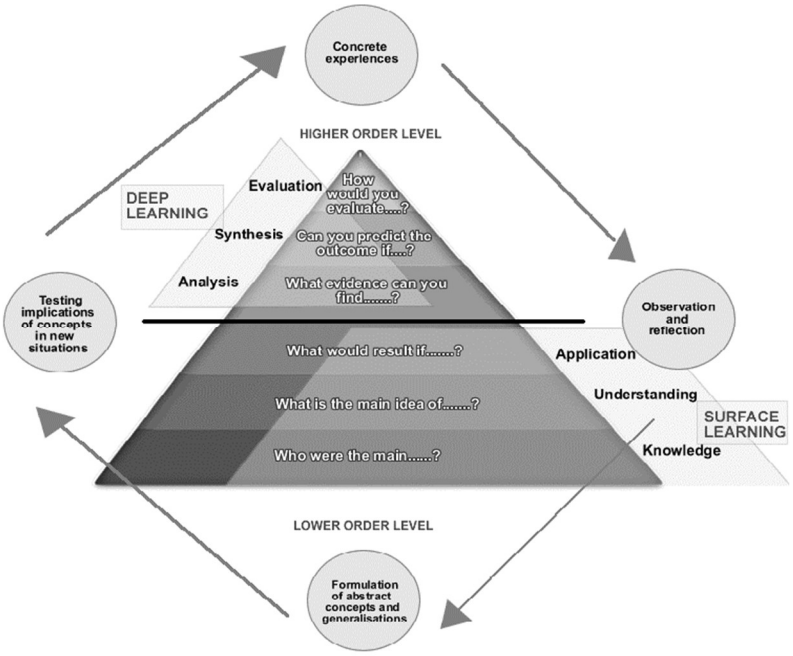

Fig. 4. Correlation between Kolb's Learning styles and Bloom's Taxonomy

3. Overview of the Types of Software Architectures and Mechanisms for Designing Software Environments, Including Those for e-Learning Environments. The process of designing a software environment, and in particular a web-based software environment, which is more common to be used for the e-Learning environments, has a number of characteristics. These are the various design mechanisms that should be considered before proceeding with the actual design of a software system:

- Version control
- Tight coupling vs. Loose coupling
- SOLID principles
    - Single Responsibility Principle
    - Open-Closed Principle
    - Liskov Substitution Principle
    - Interface Segregation Principle
    - Dependency Inversion Principle

- Inversion of Control
- Multi-tier architecture
- Logging enabled
- Caching – Database calls, Service calls, HTTP requests and responses
- Continues Integration / Continues Delivery (CI/CD)
- Switching from vertical scaling to horizontal scaling with load balancer

On the other hand, the modern software development knows three main types of software architectures:

- Monolithic Software Architecture
- Service Oriented Architecture (SOA)
- Microservice Architecture

Using the right principles and tools to work is only part of the condition for designing a reliable system that is conducive to expansion, reliability and easy maintenance. These are the most common problems of the modern e-Learning systems – even if they contain a lot of quality content and good opportunities for its presentation, they could reach the threshold of their development and usage, where it will not be possible to include new participants or new features. In this case, it is not possible to include new participants in the learning process/users or new functionalities into the system itself.

4. "Concept for Designing and Software Scaling" from the Perspective of a e-Learning Environment. The *"Concept of software scaling"*, as it is called in the paper, is part of the study based on the various software architectures, tools and principles for designing an e-Learning environment. It is valid for any web-based software and can serve as a guideline to be used in the design and planning of the stages of a system evolution. It aims to determine the appropriate software architectures to be used for each stage of evolution and the set of prerequisites that must be met before progressing to the next stage of the evolution.

The "concept of software scaling". presented in Fig. 5, defines a taxonomy of 7 steps describing the evolution of a software system, as well as the mandatory elements that must be available in order to move smoothly to each subsequent step.

A common base of mandatory components that must be available is defined for all stages of software system evolution. They must be implemented in the system in its initial stage of evolution, but remain a mandatory part of

each of the next stages. These components are closely related to each other and often the performance of one of the components requires the presence of another component. For example, following the SOLID principles for software development will provide an opportunity to achieve loosely coupled components and the ability to have the system well tested due to the interface representation of business logic [16]. On the other hand, the use of software interfaces will enable the use of Inversion of Control (IoC). Last but not least, the provisioning of a multi-layered architecture and the REST architecture of the application ensures easy migration to the service architecture, caching and separation of individual components of the system [16, 17, 18, 19].

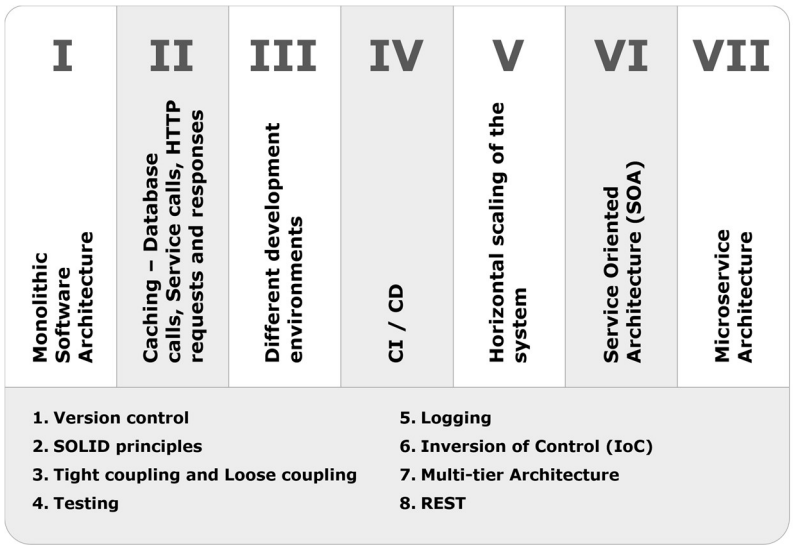

Fig. 5. Stages of software system evolution – Concept for Software Scaling

I. Monolithic Architecture – the natural start of the development of any system is to create its Monolithic Architecture. At this initial moment the domain described by the business logic of the system is very narrow

II. Caching – once the learning system has reached a certain maturity, the load on the database it uses to represent the repository of learning content increases. It is expected that the system will often have to make calls to the database, so different caching mechanisms should be inspected. The process starts with caching of the results returned from the database, but then could evolve to caching at the HTTP level. In

other words, we could start caching the requests to the endpoints of software system or caching the calls of the software system itself to any HTTP service used by it. The use of REST architecture, as described in the base layer of the taxonomy, makes achieving this stage fairly easy

III. **Different Environments** – the first three stages of system evolution are expected be activated almost immediately after its creation. The stage in which more than one environment is used is almost a prerequisite for any software solution. The current research identifies 4 mandatory development environments – (1) DEV environment (Development environment shared between the team/s of developers), (2) Staging environment (Determine whether the developed features meets the requirements), (3) Canary environment (Used for testing on production servers and using the production database, without compromising the user experience with the system), (4) PROD (Production environment)

IV. **CI/CD** – a full regression testing of the new functionalities will be provided by automatically starting the tests – part of the system. Moreover, thanks to the CD process, we could achieve an automatic deployment and delivery of every new version of the system to each of the environments described in stage III – Different environments

V. **Horizontal Scaling** – the e-Learning system should initially be expanded mainly vertically, updating the parameters of the servers on which it is hosted, as providing "horizontal scaling" requires additional resources – human and technical. At the same time, in case of reaching a higher load, this will make the simple vertical scaling unprofitable and even impossible. Therefore, the possibility of horizontal scaling should be provided by adding a Load Balancer. This is going to make the system resilient with more than one instance of the system available. It is very important before progressing to this stage to check how the Caching stage is designed as the use of more than one instance of the same system may have an impact on the caching mechanisms. For example, the system should not only cache locally on the server where the instance is hosted, but eventually to use some sort of distributed cache

VI. **Service Oriented Architecture (SOA)** – the constant analysis of performance logs, that should be implemented on the base stage, should outline the actual load on the system. This is going to show us when it is time to separate one or more parts of the monolithic application in a

separate, independent system, using SOA. The aim is not to scale the whole system horizontally, as this would be unprofitable, but only the most loaded part of it

VII. **Microservice Architecture** – this stage requires a large resource of people and technical infrastructure. He we talk about scaling the team, not just scaling the system itself, so it is important that it be conceptually envisaged in the development of the learning system

5. Component-based Software Architecture. The last stage of the "concept of software scaling" developed in the current research describes the Microservice Architecture. At the same time, this type of architecture requires the expansion of the software support team itself – human resources and an infrastructure to support the software (each new service requires a server to be hosted, and hence many network connections, databases and more).

As part of the current research a new type of flexible software architecture was designed [20]. The architecture aims to provide an easy support for larger monolithic applications, as well as flexible way to deploy the latest versions of the software. Nevertheless, thanks to the suggested architecture each individual part of the software system could be maintained and developed individually. As it sounds, these are most of the benefits usually provided by the Microservice architectures. However, in contrast to the microservises, which require a complex infrastructure and a set of teams to support them, the goal of the suggested architecture is to be supported easily and not that different than the regular, well known, Monolithic architecture. This novel architecture is called *"component-based architecture"*.

As can be seen from Fig. 6, the latest stage of the system evolution – Microservice architectures, according to the Concept of Software Scaling, is now replaced by the Component-based Architecture, developed as part of the current paper.

Breaking big things into smaller components and concentrating on them has always aroused the interest of engineers. In their designs, the engineers aim to provide easy maintenance of small replaceable parts that, when combined together, have high quality and performance. Software development is no exception to this principle of operation. A fundamental concept behind object-oriented programming is the creation of parts of programing code that group semantically related logic and can be maintained

independently of other parts of the system. This is clearly modeled in the SOLID programming principles [16].

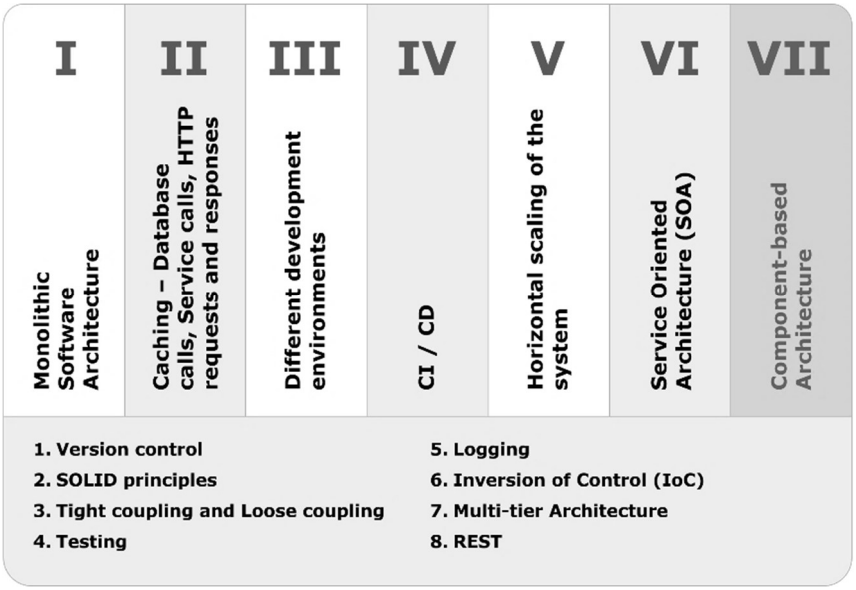

Fig. 6. Stages of software system evolution using Component-based Architecture – Concept for Software Scaling

Component-based software engineering considers the principles for ensuring the reusability of the individual components of a system, while ensuring the necessary separation of concerns. The component based-software architecture is based on this software development model. It aims to look beyond the well-known design patterns and to provide loosely coupled components that are completely autonomous and independent of each other [20].

5.1. The Building Parts of Component-based Software Architecture. In his paper, James Gallagher [16] suggest that there is a correlation between Kolb's Learning styles and Bloom's Taxonomy. He argues that different learning styles should go through all levels of the learning process defined by Bloom, but in different sequences. The principle of traversing the levels in a clockwise sequence is used, and a specific level suitable for the different learning styles is used to start the cycle.

The main building part of the component-based software architecture are the components. They must be independent of each other, be able to be

maintained and developed absolutely individually, and also be able to work autonomously. The components are combined logically according to a specific definition by a component orchestrator and thus ensures the completeness of the software system.

This paper describes the types of "components" required to create an e-Learning system. According to the model, the system has 5 types of components (Fig. 7):

1. **Components that do not offer any visual part:** Types of software libraries that aim to contain shared business logic between individual components
2. **Components with a visual interface:** Small building block parts of a web-based solution – modal windows, components for presenting a learning object (text, image, etc.), headers, footers and more
3. **Components aggregators of other components:** With their help it is possible to annotate the components and create meaningful and valid content. They also could be considered as components with visual interface
4. **List of system definitions, referred in the paper as Manifest:** Defining the components of which the software environment is composed
5. **Orchestrator / composer of components, referred in the paper as Bootstrapper:** Aiming only to load a set of components according to the list of system definitions

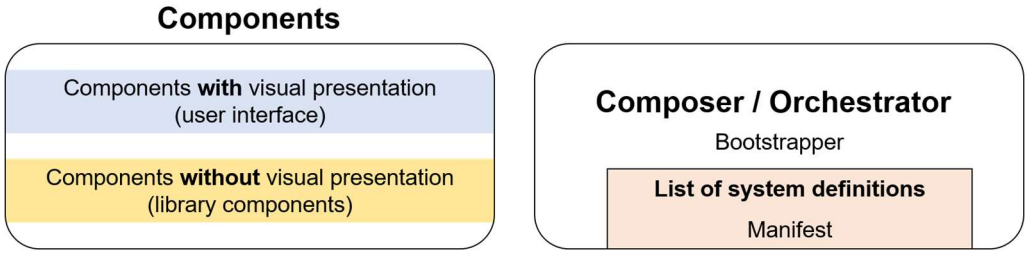

Fig. 7. Component-based software architecture – building blocks

The list of system definitions presents a list of all used by the system components. Adding a new component in the system and thus extending its functionalities is extremely simple and requires no more than updating the

definition file. In this way, *Bootstrapper* automatically loads each subsequent component (Fig. 8).

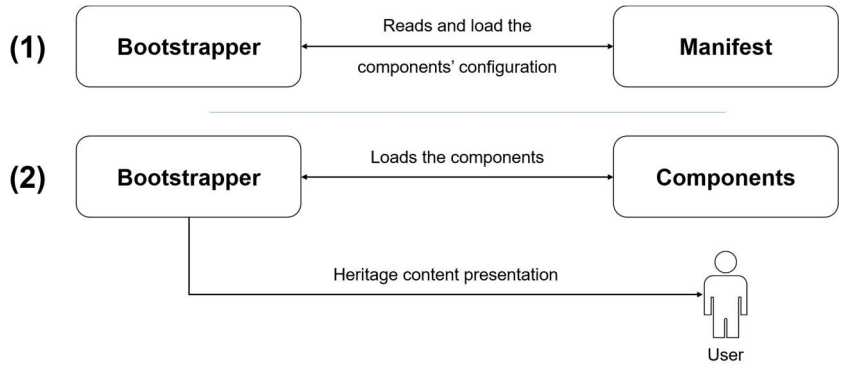

Fig. 8. Component management in component-based software architecture

5.2. e-Learning System Designed Using "Components". The use of component-oriented software architecture to design an e-Learning system on the one hand provides a number of advantages for the system itself – each individual "component" of the system could be managed and maintained separately. For example, the creation of a component for presenting a learning object, e.g. video, could be maintained and developed independently of another component for the presentation of test questions, for example. On the other hand, this architecture works in full sync with the developed models for generating personalized learning materials – it provides an opportunity to gather small and independent components, which together could they generate meaningful content for people with different learning styles. At the same time, thanks to its design, this architecture allows easy scaling of the system and its continuity, even if there is a release of a new software version.

This publication proposes the use of three main types of components to support a complete e-Learning system (Fig. 9).

- **User Interface (UI) Components:** All types of components that can represent learning objects, such as text, questions, exercise, example, video, image, etc.
- **Page Components:** They aim to aggregate several UI Components and then to present the result in the meaningful and well formatted content.

- **Fundamental Components:** Components not having visual interface, but some business logic of fundamental use that could be shared between multiple components (equivalent of software libraries).

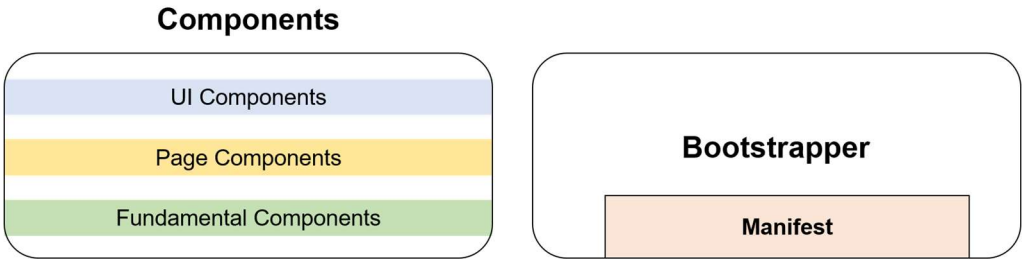

Fig. 9. Component design of an environment for the presentation of heritage content

The flexibility provided by the UI Components and subsequently their aggregation by the Page Components allows the easy extension and maintenance of the system. For example, a new UI Component could be created to support a GeoGebra plug-in as part of a Math related learning material. It then only needs to be added and configured in the Manifest.

The definition of the contextual connection between the individual components (the use of one component in another) can be represented by a relational database describing a tree structured relationship between "root" (Page Components) and "leaves" (UI components) (Fig. 10). For example, a Page Component may support video presentation (Video Component) and GeoGebra plug-in (GeoGebra Component).

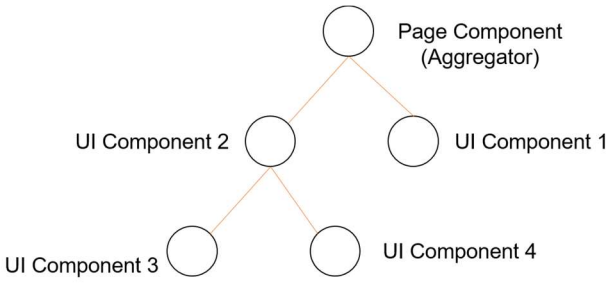

Fig. 10. Tree structured relationship between "root" (Page Components) and "leaves" (UI components)

Each component is versioned and the version is then represented in its description and is part of the *Manifest*, which in turn also has its own description – Manifest ID (Fig. 11).

```
{
  "ManifestId": "Manifest-1",
  "Components": [
    {
      "Name": "HomePageComponent",
      "Version": "1.0.0"
    },
    {
      "Name": "LibraryComponent",
      "Version": "1.0.0"
    },
    {
      "Name": "ImageComponent",
      "Version": "1.0.0"
    },
    {
      "Name": "VideoComponent",
      "Version": "1.0.0"
    }
  ]
}
```

Fig. 11. Model of a Manifest

Each new version of a component is being deployed to the web server in off-state, but it is being "activated" with a change of the Manifest, where its version should be referenced (Fig. 12).

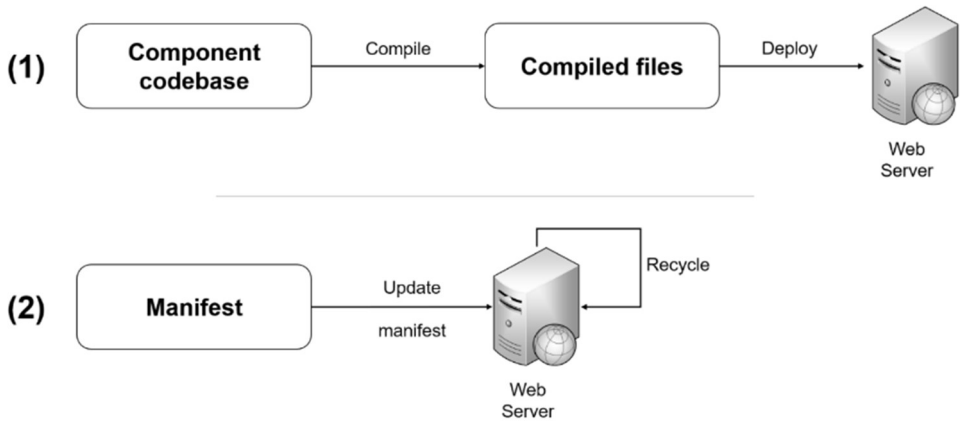

Fig. 12. Development life cycle of a component and its way for activation

The advantages of this model for software architecture are many, but some of the most important are: (1) ability to independently support each part of the system; (2) flexibility to constantly extend the system; (3) the accelerated software development life cycle. Unlike the software applications that use a monolithic architecture, in the component-based software architecture, changing a small part of the system does not require update and release of the entire system. For example, changing the image visualizations and styling is isolated to the Image Component only – the other components do not change at all. This accelerate the software development life cycle and increases its productivity. The delivery of the changed components is instantaneous, and their "activation" is possible by only changing the Manifest. In other words, activating but also deactivating delivered changes in components or introducing new components is a matter of seconds. Moreover, the time for testing and quality assurance of the changes made is also shortened, as the part of the system to be tested can be closely isolated to a single component without the need to test the whole system. Moreover, the components with a visual interface can be tested absolutely independently of the other components.

5.3. Framework to Manage e-Learning Content. The aggregation of the individual UI components must be easy and convenient in order to achieve a powerful Content Management System (CMS). The model designed in the paper considers the CMS from three perspectives (Fig. 13):

- **Template management of the individual Page Components:** The logic (potential relationship with UI Components) and layout (markup and styles) contained in the Page Component itself. An exact location of an individual UI Component or set of UI Components (children – without specified a single one) could be configured and styled.
- **Component relationship management:** Describes the relationship between the individual components (UI/Fundamental/Page Components), for example the relationship between the Page Component – Home page, with the UI Component to present images – Image Component
- **Component data management:** Standard functionality for any CMS. It is controlled by the database, where the metadata of the individual learning object is stored

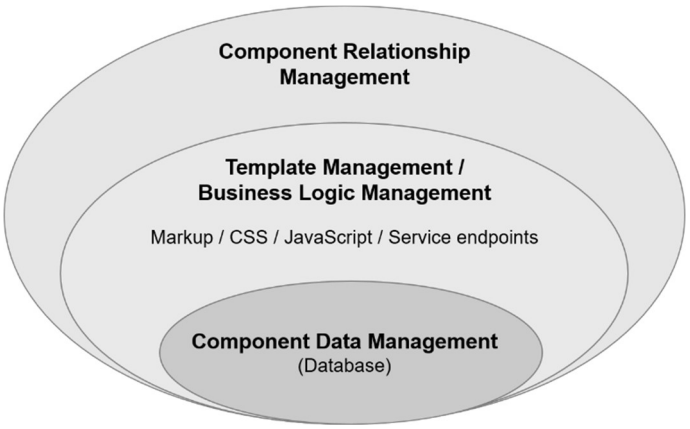

Fig. 13. CMS perspectives considered

6. Verification of User's Identity. The modern learning and content management systems offer a virtual environment in which the user (teacher or student) consumes learning materials, generates content, assesses his/her knowledge, and also has the opportunity to obtain a certificate attesting to his/her completed training course. In other words, these types of systems aim to completely replace the conventional method of offline training, whereby all participants in the learning process meet in person and the verification of their personalities becomes easy and direct (the learner is introduced to the trainees and both know each other), with one in which the connection between all participants in the process is virtual (authentication and verification of individual participants in the process is necessary for them to present themselves to the system and from there to the other participants). For this reason, it is absolutely necessary to ensure that consumer identity can be verified.

Even with the provision of a highly reliable service for a user authentication and authorization, the human factor runs the risk of compromising the process. Users can be expected to be careless when storing or using their passwords. This outlines the question, "how can we be sure that the user posing as a teacher is indeed one and not using the teacher's account maliciously?". The same question is applicable also for the students, which have the opportunity to receive a certificate as the end of a course.

According to the General Data Protection Regulation (GDPR) [21], binding to more than one user authentication method ensures that consumer identity is fully identified [22, 23]. In other words, the combination of a

username and a phone number is a sufficient way of verifying a user's identity. Following this idea, various services are now available over the Internet to enable third party verification of the user's identity instead using well known electronic signatures. For example, the DocUSign system provides the ability of having a fully legitimate and legally enforceable contracts signed remotely by the users. With LMS and LCMS it is possible to simultaneously associate email, username and phone number (Fig. 14).

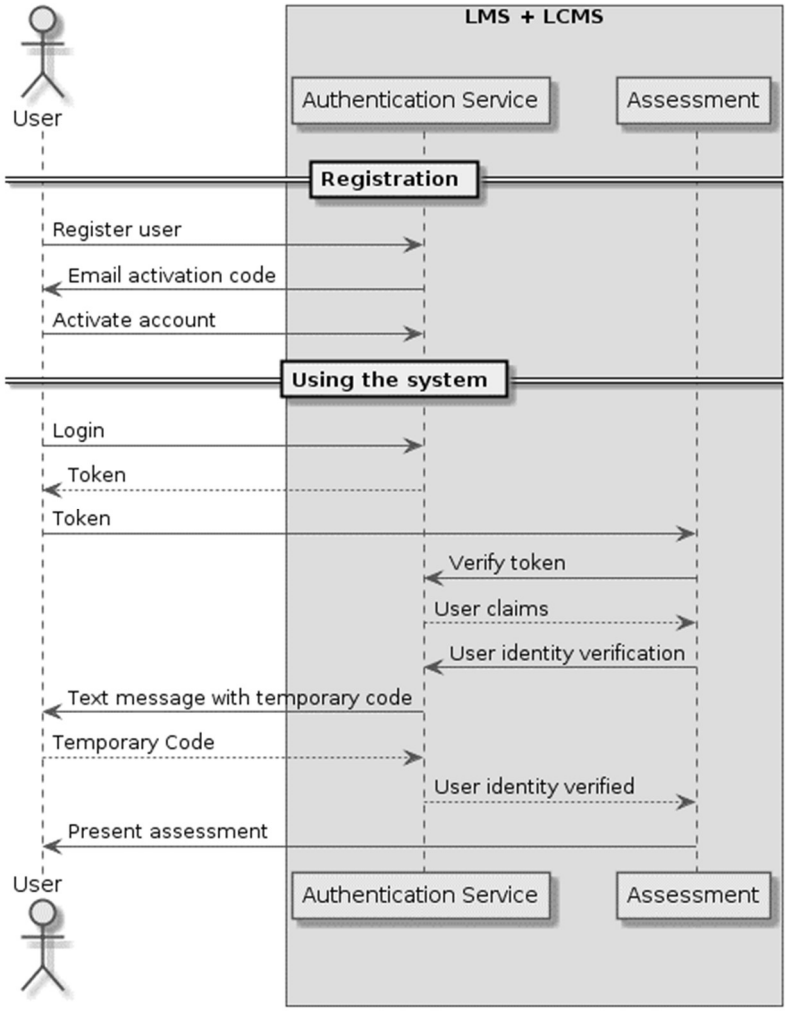

Fig. 14. User authentication and identity verification

During the registration process into the LMS or LCMS the user chooses a username that makes it unique to the system, then he/she is required to enter an email to use to communicate with him/him and confirm his/her registration by going to the address cited in an automatically generated email to activate the account. This provides the system with the ability to link the personal email address of the user with a unique identifier for the system in the face of the username. Further, if a code is sent to the user's phone number as part of a text message during any user's attempt to interact with the system, such as when starting an exam or when generating a learning content, we can be assured that the system provides verification of the user's identity. It is not possible for a malicious person to impersonate the user as even "stealing" the password is not enough to interact with the system.

7. The Structural Way for Binding a Learning Material with Personal Preferences of Learners. Dividing the learning material in such a way as to adapt its presentation with Bloom's Taxonomy would increase the productivity of learners in the process of acquiring new knowledge. This productivity could be further enhanced if the segregated learning material is rearranged to follow a specific learning style appropriate for each of the different types of learners. In other words, the sequence of goals to be set for learners must be linked to the most appropriate learning style for them. And while Bloom, Honey and Mumford models are mainly focused on the processes that take place in learner's mind while he/she acquire new knowledge, as well as what is the best way for the learners to gain that knowledge, this paper is focused on something different. Following the example of the model of James Gallagher, who provides an opportunity to correlate the learning cycle of Kolb to the Bloom's Taxonomy, this research presents a model that addresses the creation of learning content using concepts based on the Boom's Taxonomy and the Learning Styles of Honey and Mumford. In this case, the question is not what cognitive processes take place in student's minds and how they achieve each of the learning goals, but how to construct a learning material so as to provoke appropriate cognitive processes and set the learning goals by itself. Moreover, the material itself should be presented in an appropriate way for each learning style, according to Honey and Mumford [24] (Fig. 15).

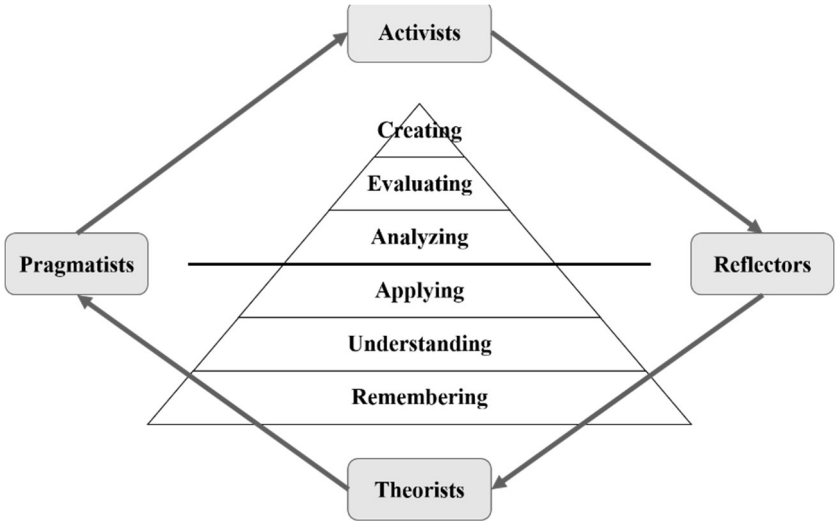

Fig. 15. Gallagher's adapted model illustrating the correlation between the Learning Style, described by Honey and Mumford, and the learning goals, described by the modified version of the Bloom's Taxonomy

The sequence of processes defined in Bloom's Taxonomy through which the learners with different learning styles must pass, according to Honey & Mumford, is graphically presented in Fig. 16.

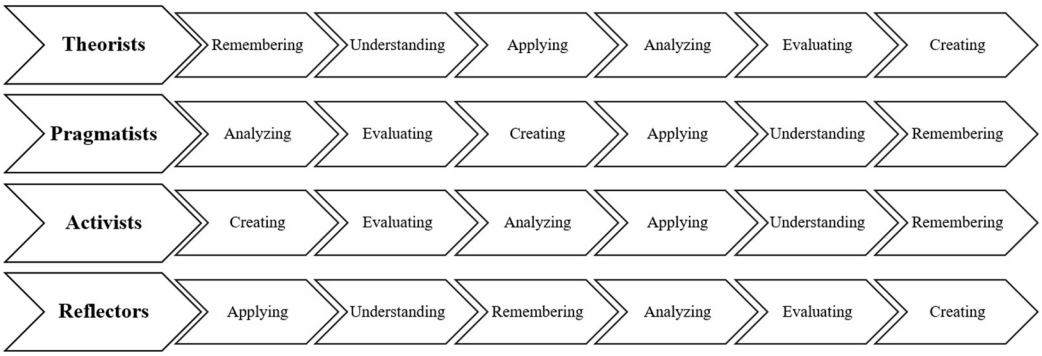

Fig. 16. The sequence of processes as defined in Bloom's taxonomy, through which people must pass for each learning style defined by Honey & Mumford's learning cycles

7.1. Sources of Learning Content for Creating Personalized Learning Materials. The model, developed as part of the study, aims to create personalized learning content personalized to each learning style, according to Honey and Mumford, and including separate sections describing each part of Bloom's Taxonomy and learning objective, which this part describes. This model could be digitally represented – as part of an e-Learning system that creates personalized learning materials. Moreover, it is envisaged that the learning objects will be small enough so that they can be reused in more than one learning material. In other words, the teacher could choose to create a new learning material, self-uniting the individual 6 components for it, and then leave the system, following the model, to create appropriate learning content for people with different preferences for learning styles. At the same time, the teacher has the opportunity to use already created learning objects to apply in the learning material he/she creates. This provides the e-Learning systems with extra flexibility, so instead of using the conventional way for creating learning content (in form of encoded files that cannot easily be indexed and reused), the learning content created with this model is easy to index, search and create new meaningful learning content [24].

In order to provide an opportunity to create learning content, the paper suggests a structure for describing learning materials with a set of metadata descriptors (Table 1). Thanks to these descriptors it is possible to achieve reusability of the learning objects. Moreover, the objects could be indexed and searched, so they could be used for automatic generation of learning materials [24].

Table 1 presents the structure of the metadata needed to describe all single pieces (learning objects) of the low-level information object [24].

Table 1. Structure of the metadata used to describe the individual learning objects associated with any learning material

Descriptor	**Field Type**	**Example**
Language	Language code selected from ISO 639-1	*en-US*
Learning goal	Value selected from a predefined list	*Definition*
Content	Text area field with the option of adding an image/video/audio or	*An object moves if it changes it position in time*

	an external reference	*compared with another object.* *The object is at rest (it is still) if it does not change its position relative to the orientation.*
Complexity level	Low/Normal/High	*Normal*
Level of education	Value selected from a predefined list	*VI-th grade*
Learning context	Value selected from a predefined list	*Human and Nature*

The final aggregation of all the individual learning objects to generate a meaningful and personalized learning content, could be done automatically by searching existing repositories with learning materials. In order to archive that a "smart system" that recognizes the content of information and classifies it according to the proposed structure should be created (Fig. 17).

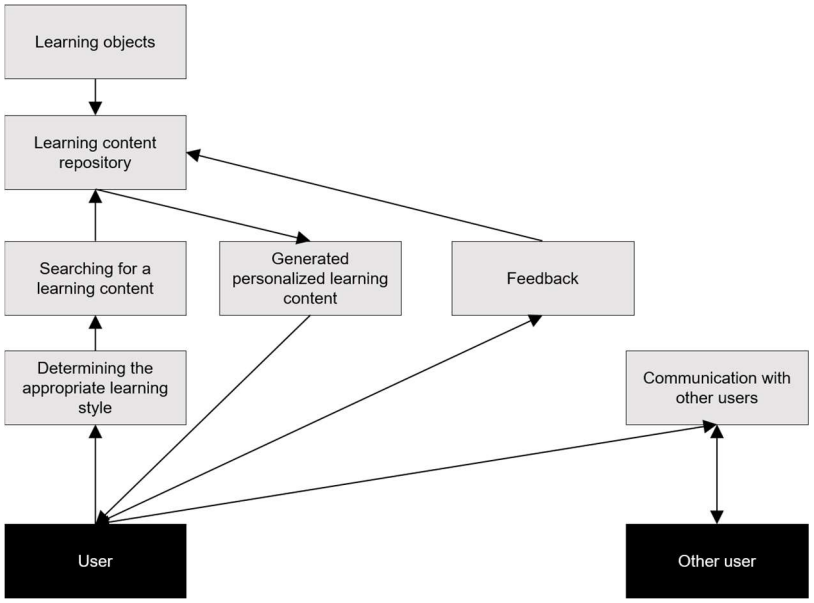

Fig. 17. Process of creation personalized learning content in an e-Learning system

8. Testing and Analysis of Results.

8.1. Opportunity to Collect Feedback and Enhance the Motivation of Both the Teacher and Student. The challenge in front of any new way of presenting learning content is to achieve a right level of motivation of both the teachers and students, so they are interested to use it. In other words, it is not enough that the model for presenting learning materials to be effective and well implemented into an e-Learning system, but the system must also be used by well-motivated users – teaches and students.

Motivation is of great importance for the productivity of learners in the process of acquiring new knowledge.

The developed model requires a special attention by the teachers to describe the learning objects they add. Without appropriate description of the learning objects it won't be possible to automatically generate personalized learning content by the e-Learning system. Therefore, it is important to pay attention to the level of motivation of the teachers themselves, who should be encouraged to create learning objects. This study suggests a potential solution to the challenges with the motivation of participants in the learning process – the use of the so-called Gamification [25]. This is done in two aspects:

1. **Gamification is used to enhance the motivation of the creators of learning content and learning objects** – according to this model, they should collect virtual reward, in correlation to the learning materials they create. Also, depending on the feedback left against the added learning content by teachers and students, the creators of content should be extra rewarded – people who generate better content need to receive the right level of acknowledgment
2. **Gamification is implemented in the learning process to motivate and evaluate the learners themselves** – according to this model, they should collect virtual rewards, in correlation to the amount of learning materials they covered and the results from the assessments they have taken. Moreover, the e-Learning system should provide real-time statistics on the most trained learners, with the highest scores, etc.

When such a mechanism is applied to an e-Learning system that aims to create personalized learning content through small and independent components, it could become a corrective to the content produced itself. For example, learning objects that didn't receive a positive feedback will not be used by the system in order to create new learning materials.

8.2. Validation and Verification of Methods and Models.

Creating a conceptual model to solve a problem is a complex process. Creating the model itself is not a one-time operation. Usually the models are adapted and modified, based on the analysis of the problem, as well as on the results of the conceptual validation of the created model [26]. The conceptual validation of thematic-oriented learning content methods and models is based on in-depth analyzes of the advantages and disadvantages of the Bloom Taxonomy and the benefits it brings to the learning process, as well as Honey and Mumford's learning styles. Subsequently, the effectiveness of the correlation between these two models is further analyzed and validated using experiments performed with learners.

After the creation of the conceptual model, a computerized model is created (implemented), which fully reflects the conceptual one and presents all its methods and goals (Fig. 18). This computerized model is created as part of an e-Learning system. This system also includes the repository to store structured learning content described in the paper.

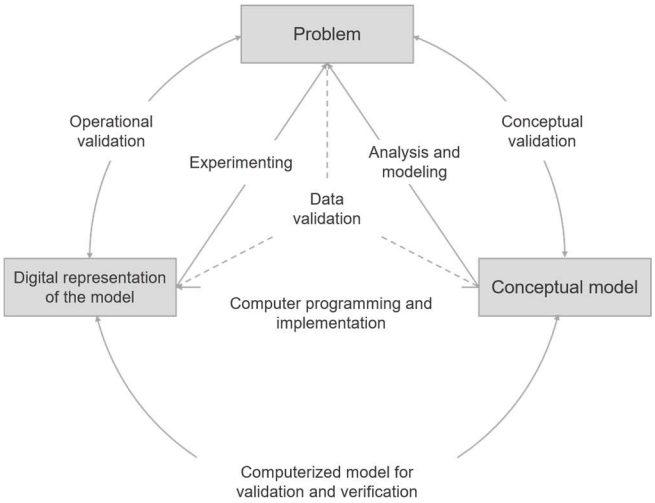

Fig. 18. Validation and verification of methods and models using their digital representation

The e-Learning system provides operational validation of the models and methods created as part of the study. In other words, we are no longer just talking about a concept, but a real validation of the results is obtained.

The transformation of the conceptual model into a computerized model provides an opportunity to create software approaches for validation and verification of models and methods. Of course, when we talk about computerized validation and verification, it cannot be abstract, but needs to be done with pre-generated input data to be used in random combinations, as well as a control set of data to be validated. The generated learning content is verified and validated using unit test and integration tests to verify if the learning objects were ordered properly to follow the models, described by the paper.

As the models developed as part of this study are not limited to creating customized learning content, but address problems in designing e-Learning systems and specifically proposed a new type of software architecture, the current study also evaluates the performance of the system when it loaded to a high level, using special load testing.

9. Conclusions. The possible ways to optimize the presented learning content into e-Learning system and its better absorption by learners is the subject of this research. The paper considers the possibilities for improving the productivity of the e-Learning process in four main areas: (1) reusability of a learning content; (2) personalized representation of a learning content; (3) proper identification of participants into an e-Learning process; (4) opportunities for easy scaling of learning environments. As a result of the research, a set of methods and models for creation of a personalized, according to the learner's preferred learning style, learning materials from a thematic-oriented content have been developed.

The research analyzes some existing methods and models for modeling of the processes of acquiring new knowledge. Bloom's Taxonomy, as well as Honey and Mumford's Learning Styles, are reviewed. As a result, conceptual models and methods for creating learning content have been developed, and unlike the research of Bloom, Honey and Mumford, it is not about thinking activities oriented to the minds of learners, but rather about combining principles from these existing models to achieve a successful creation of a personalized (to the students learning style) content.

The validation and verification of the conceptual models and methods is done by adopting them into a software environment.

The research also presents so called "concept for software scaling" from the perspective of an e-Learning environment and a novel software

architecture (Component-based software architecture) to be used as a base of a system implementation.

The evaluation of the value of the generated learning content was made using the method of A/B testing (active experimenting) with actual students.

REFERENCES

[1] Bloom B. Taxonomy of Educational Objectives. Book 1: Cognitive Domain. New York, David McKay, 1956.

[2] Honey P., A. Mumford. The Manual of Learning Styles. 3rd ed., Maidenhead, Peter Honey, 1992.

[3] Wagner E. D. Steps to Creating a Content Strategy for Your Organization. *The eLearning Developers' Journal*, October 29, 2002.

[4] Anderson L. W., D. R. Krathwohl, P. W. Airasian, K. Cruikshank, R. E. Mayer, P. R. Pintrich, J. Raths, M. C. Wittrock. Taxonomy for Learning, Teaching, and Assessing, A Revision of Bloom's Taxonomy of Educational Objectives. Pearson, 2001.

[5] Kostadinova H., G. Totkov, M. Raykova. Towards automated question generation according to Bloom's taxonomy. In: Proceedings of the Fortieth Jubilee Spring Conference of the Union of Bulgarian Mathematicians, Sofia, 2011, 413–422.

[6] Kolb D. A., R. E. Fry. Toward an Applied Theory of Experiential Learning. In: C. Cooper (ed.) Theories of group processes. New York, John Wiley & Sons, 1975, 33–57.

[7] Klisarov Yu. Izbor na skala za oceniavane na stila na uchene. *Profesionalno obrazovanie*, **1** (2013), 55–60. (in Bulgarian)

[8] Leaver B. Learning styles and learning strategies (Chapter 3). In: Leaver B., M. Ehrman, B. Shekhtman. Achieving Success in Second Language Acquisition. Cambrige University Press, 2005, 65–91.

[9] Buch K., S. Bartley. Learning style and training delivery mode preference. *Journal of Workplace Learning*, **14** (2002), No 1, 5–10.

[10] Zwanenberg N. V., L. J. Wilkinson, A. Anderson. Felder and Silverman's index of learning styles and Honey and Mumford's learning styles questionnaire: How do they compare and do they predict academic performance? *Educational Psychology*, **20** (2016), No 3, 365–380.

[11] Boyle T., J. Cook. Learning objects, pedagogy and reuse. In: Learning technology in transition. From individual enthusiasm to institutional implementation, Taylor & Francis, 2003, 31–44.

[12] Mason R., D. Rehak. Keeping the learning in learning objects. In: A. Littlejohn (ed). Reusing online resources: a sustainable approach to e-learning. Kogan Page, 2003, 20–34.

[13] Iliev O., R. Yoshinov. "Controlled self-study" in thematic educational community environment. In: Proceedings of the Forty-seventh Spring Conference of the Union of Bulgarian Mathematicians, Sofia, 2018, 200–213.

[14] Yoshinov R., O. Iliev. Reuse of content – a major problem of modern systems for storing educational content. In: Proceedings of the National Conference on "Education and Research in the Information Society", Plovdiv, 2018, 204–215.

[15] Gallagher J. G. The Business Case Study: A Suitable Candidate for Blended Learning? *Journal of Business Case Studies*, **2** (2006), No 4, 5–18.

[16] Joshi B. Beginning SOLID Principles and Design Patterns for ASP.NET Developers. Apress, 2016.

[17] Huizinga D., A. Kolawa. Automated Defect Prevention: Best Practices in Software Management. Wiley-IEEE Computer Society Press, 2007.

[18] Cervantes H., R. Kazman. Designing Software Architectures: A Practical Approach (The SEI Series in Software Engineering). Addison-Wesley Professional, 2016.

[19] Nadareishvili I., R. Mitra, M. McLarty, M. Amundsen. Microservice Architecture: Aligning Principles, Practices, and Culture. O'Reilly Media, 2016.

[20] Iliev O. "Hot releases" in learning management systems and learning content management systems. In: Proceedings of the Forty-ninth Spring

Conference of the Union of Bulgarian Mathematicians, Sofia, 2020, 137–143.

[21] Regulation (EU) 2016/679 of the European Parliament and of the Council of 27 April 2016 on the protection of natural persons with regard to the processing of personal data and on the free movement of such data, and repealing Directive 95/46/EC (General Data Protection Regulation). *EUR-Lex.* 4 May 2016. `https://eurlex.europa.eu/legal-content/en/TXT/?uri=CELEX%3A32016R0679`, 30 January 2022.

[22] Iliev O., R. Yoshinov, G. R. Tsochev. Verification of user identity and data security in the context of LMS and LCMS. In: Proceedings of the Forty-ninth Spring Conference of the Union of Bulgarian Mathematicians, Sofia, 2020, 144–151.

[23] Kabamba P. Know Your Customer (KYC) Policy: The Bank's Account Opening Procedures, Identity Verification Procedures and Customer Risk Assessment Procedures, 2016.

[24] Iliev O., R. Yoshinov. The Structural Way for Binding a Learning Material with Personal Preferences of Learners. *SPIIRAS Proceedings,* **5** (2018), No 60, 189–215.

[25] Robson K., K. Plangger, J. H. Kietzmann, I. McCarthy, L. Pitt. Is it all a game? Understanding the principles of gamification. *Business Horizons*, **58** (2015), No 4, 411–420.

[26] Sargent R. Validation and verification of simulation models. In: Proceedings of the 2010 Winter Simulation Conference, Baltimore, 2010, 166–183.

Oleg Iliev
Laboratory of Telematics
Bulgarian Academy of Sciences
Acad. G. Bonchev Str., Bl. 8
1113 Sofia, Bulgaria
e-mail: `ilievo@cc.bas.bg`

Received March 22, 2022
Final Accepted April 7, 2022