Informatics in Education, 2018, Vol. 17, No. 1, 1–20
© 2018 *Vilnius University*
DOI: 10.15388/infedu.2018.01

A Methodology to Define Learning Objects Granularity: A Case Study in Software Testing

Fabiane Barreto Vavassori BENITTI
Federal University of Santa Catarina (UFSC)
Florianópolis, Santa Catarina, Brazil
e-mail: fabiane.benitti@ufsc.br

Received: June 2017

Abstract. Learning Object (LO) is one of the main research topics in the e-learning community in the recent years. In this context, granularity is a key factor for LO reuse. This paper presents a methodology to define the learning objects granularity in the computing area as well as a case study in software testing. We carried out five experiments to evaluate the learning potential from the produced learning objects, as well as to demonstrate the possibility of LO reuse. The results show that LOs promote the understanding and application of the concepts. In addition, the set of LOs identified through the proposed methodology allowed its reuse in different contexts.

Keywords: learning object, methodology, reuse, granularity, software testing.

1. Introduction

While the reputation of testing in the software industry varies from mediocre to belittled, the ACM and the IEEE-CS (ACM/IEEE Computer Society, 2013) situate program testing as a relevant part of the core of undergraduate Software Engineering education (Salzer, Haberman, & Yehezkel, 2012). The lack of qualified professionals may be one reason companies do not have mature testing processes (Myers, Badgett, & Sandler, 2012) (Dias Neto, Natali, Rocha, & Travassos, 2006), which occurs due to the little attention given to the subject in the curricula of the Computer Science and Software Engineering courses (Chen, 2004) (Stroustrup, 2010).

From the need to support the flexible software testing training to suit different teaching contexts – business training and undergraduate courses as well as consider various curricula, we discussed in this article a methodology for defining a set of reusable learning objects (LO), applied in software testing area. However, the proposed methodology can be extended to any computing area.

What is the reason for creating a methodology to define the set of learning objects? It is in order to enable the use of learning objects in different contexts, due to the fact that a key and difficult factor to balance regards to granularity (Wiley, 2001). Granularity refers to the object size, it can be understood as "the smallest portion of the object with all the essential information of a subject" (Braga, Dotta, Pimentel, & Stransk, 2012). A very large LO can derail its reusability because it contains a complexity of content that is suitable to a limited number of contexts. Therefore, granularity must be set to increase the reusability. Polsani (2006) highlights that the size of a LO is crucial to achieving success in its reusability and Moyse and Ally (2013) explain that "the finer the granularity, the greater the potential to reuse the LO in different situations; on the other hand, smaller LO tend to have less educational value and result in reduced and/ or lower-level learning."

This article discusses in section 2 a brief background on granularity and reusability of learning object. The following sections present the proposed methodology and the instantiation of methodology for software testing area. Subsequently we present some information about the production of learning objects, including illustrating them. Further, section 5 presents the evaluation of the LO developed. Finally, the conclusions are presented.

2. Granularity and Reusability of Learning Object

Learning Object is one of the main research topics in the e-learning community in recent years, and most researchers pay attention to the issue of Learning Object Reusability (Noor, Yusof, & Hashim, 2011), sharing a common approach to the composition of the virtual courses parts, whole or partially, they must be small, digital and capable of reuse (Flôres, Santos, & Tarouco, 2007) (Wiley, 2001).

Learning objects are defined as any entity, digital or non-digital, which can be used, reused or referenced during learning supported by technology (Learning Technology Standards Committee, 2002). The most obvious motivation is the economic interest of reusing learning material instead of repeatedly creating it and other motivations can be found in the pedagogical area, since that learner-centric teaching theories invite instructors to use a wide variety of didactic material (Noor, Yusof, & Hashim, 2011).

Instructional technology identified as "learning objects" leads to technology choice for the next generation of instructional design due to its potential for reuse, generativity, adaptability and scalability, Gibbons, Nelson and Richards (2000) claimed more than 10 years ago. Nowadays, although reuse is the reason why much of Learning Object Technologies exists, little is quantitatively known about the reuse process. A quantitative analysis performed by Ochoa pointed the amount of learning objects reused is around 20% (Ochoa, 2011).

In an empirical study of MERLOT repository (Heyer, 2005), it has been found that the majority of learning resources integrates several information objects and educational components in a fixed, immutable way, which implies that the degree of reusability is extremely low. Reusability requires the LO to be in a fine-grain form because raw media elements are often much easier to reuse than aggregate assemblies. In other words, as the LOs size decreases (lower granularity), its potential for reuse increases.

Granularisation refers to the size of learning objects and it is a necessary condition for learning objects to be shared and reused. The metric for granularity varies, including instructional time, amount of learning achieved, and amount of content covered (Moisey & Ally, 2013) (Schoonenboom, 2012). In this paper, we consider that learning objects should be sufficiently small to be reusable, but large enough to ensure that professors do not have to spend time aggregating large numbers of resources (Noor, Yusof, & Hashim, 2011), so we consider that the rule to be applied is (Polsani, 2006): how many ideas about a topic can stand on their own and can be reused in different contexts? The methodology presented here (section 3) proposes to assist in the answer to this question.

When we think about reuse of a LO it is important to define what the reuse context we consider. Schoonenboom (2012) proposed four scenarios for determining the size and reusability of LO:

- Sharing scenario: The extent to which LOs can be used in treating the same topic in the same domain with the intended audience. In this scenario, LO is smallest object that will be used separately by the intended audience.
- Flexibility scenario: aims at optimizing the flexibility with which the content of the repository can be recombined. In this scenario, LO is smallest object within a set that can be used in different sequences.
- Different domains scenario: seeks to maximize different topics and domains in which the content of the repository can be used.
- Cohesion scenario: each LO addresses one idea or one learning objective. In this scenario, LO is smallest object that addresses one learning objective.

In this article, we consider the discussion in the context of "sharing scenario", that is, aims at optimizing the sharing of materials between instructors who teach or develop courses on the same topics (in this case, software testing). Schoonenboom (2012) gives a definition of optimal LO size in the Sharing scenario, noting that pieces that will be used together by the audience should be put together. But how to define which parts should be used together? Our methodology proposes a way to help in this definition.

3. Proposed Methodology

To contribute to increase LO's reuse potential, we propose a methodology to define the granularity of a set of learning objects for a computer area. To specify the methodology, in Fig. 1, we used the BPMN notation (Business Process Model and Notation) (Object Management Group, 2015).

The methodology provides two main phases definitions:

- **Phase 1** aims to define the contents that must be addressed. At this phase the effort is focused on identifying reference curricula, both internationally recognized as defined by national institutions. However, there may be national curricula that are very generic in relation to the content being addressed. In this case, we suggested a survey on teaching plans. This first stage should result in a list with the main contents to be addressed. However, we must consider whether the study area has certifications and/or standards or relevant models in the area. If so, we suggest the contents listing refinement. Thus, the main result of this phase is a listing with the definition of the contents and their sources.

- **Phase 2** aims to define the granularity of each learning object. At this phase the effort is focused on mapping the level of learning expected for each content, and then based on the result, identify the learning objects, that is, the defini-

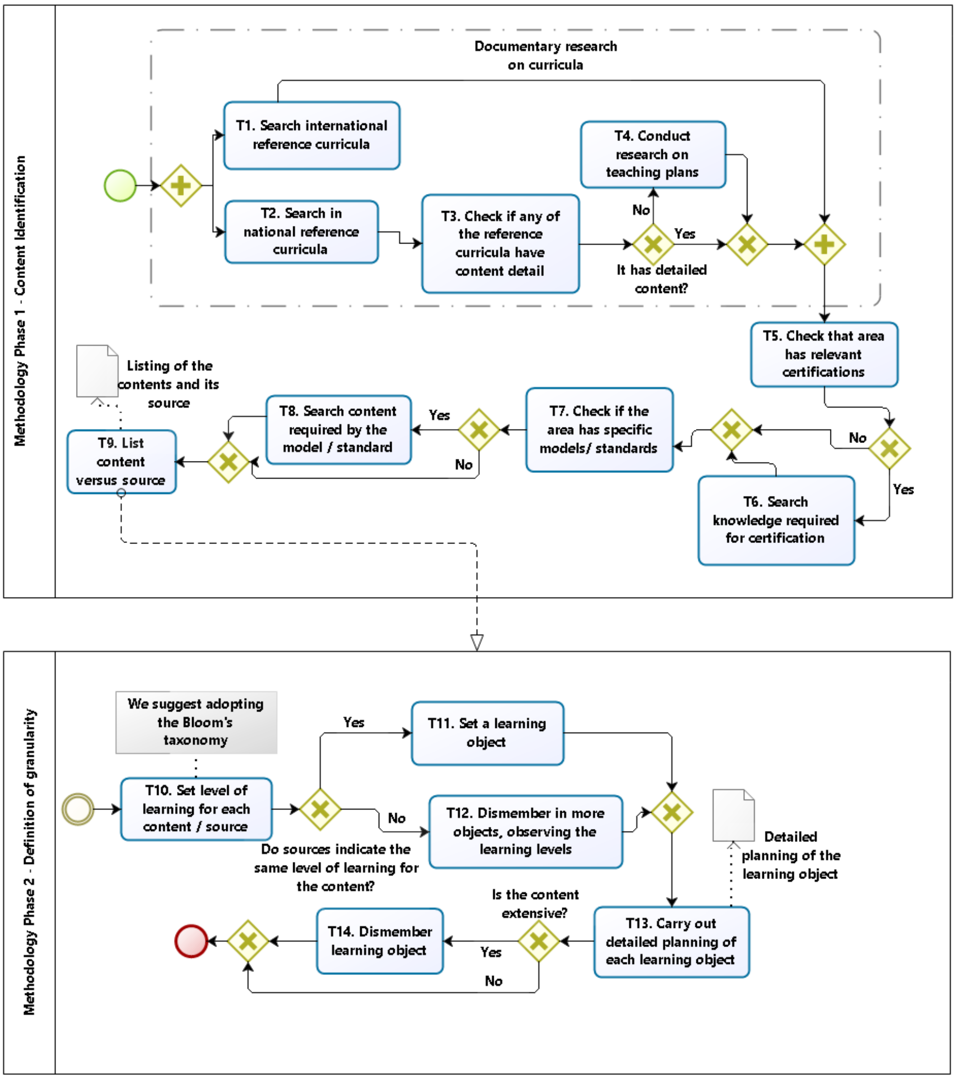

Methodology Phase 1 - Content Identification
Documentary research on curricula
T1. Search international reference curricula
T2. Search in national reference curricula
T3. Check if any of the reference curricula have content detail
It has detailed content?
No
Yes
T4. Conduct research on teaching plans
T5. Check that area has relevant certifications
No
Yes
T6. Search knowledge required for certification
T7. Check if the area has specific models/ standards
Yes
No
T8. Search content required by the model / standard
T9. List content versus source
Listing of the contents and its source
Methodology Phase 2 - Definition of granularity
We suggest adopting the Bloom's taxonomy
T10. Set level of learning for each content / source
Do sources indicate the same level of learning for the content?
Yes
No
T11. Set a learning object
T12. Dismember in more objects, observing the learning levels
T13. Carry out detailed planning of each learning object
Detailed planning of the learning object
Is the content extensive?
Yes
No
T14. Dismember learning object

Fig. 1. Proposed methodology.

tion of a learning object involves the content to be addressed and the level of learning expected. We suggest using Bloom's taxonomy to define the levels. Bloom's taxonomy is a hierarchical organization structure of educational objectives. The levels are: (i) Remembering: recognizing or recalling knowledge from memory.; (ii) Understanding: constructing meaning from different types of functions; (iii) Applying: carrying out or using a procedure through executing, or implementing.; (iv) Analyzing: breaking material or concepts into parts, determining how the parts relate or interrelate to one another or to an overall structure or purpose.; (v) Evaluating: making judgments based on criteria and standards through checking and critiquing; and (vi) Creating: putting elements together to form a coherent or functional whole; reorganizing elements into a new pattern or structure through generating, planning, or producing (Anderson & Krathwohl, 2001). It is important to highlight that if there is a source indicating a level of learning and another source requires another level, we suggest preparing two objects. For example, source A indicates the level of understanding and source B application level – in this case, two objects should be prepared, the first aimed at understanding the content and a second object focusing on the application, without duplicating content. Therefore, if you want to reach the application level both objects should be used. That done, there is a set of objects that must be detailed, that is, the content should be broken down and one must set the teaching approach. At this stage, it is noticed that the detailed content is too long, for example exceeding the period of 1 hour (Downes S. , 2003), at this specific situation it is recommended to split the content in more than one learning object.

3.1. *Methodology Application for Software Testing*

This section describes how the methodology was applied to define a set of learning objects for the software testing area.

3.1.1. *Phase 1 – Content Identification*

The documentary research involving international reference curricula considered (Fig. 1 – task T1):

- Computing Curricula 2005 (ACM; AIS; IEEE-CS, 2005): It has proposed undergraduate curriculum guidelines for five defined sub-disciplines of computing: Computer Science, Information Systems, Computer Engineering, Information Technology and Software Engineering. Software testing is covered in a discipline of Software Verification and Validation. The document presents concept for the activities of V & V, but does not detail the contents.
- SWEBOK – Guide to the Software Engineering Body of Knowledge (IEEE Computer Society, 2004): it defines a content structure for a course in software engineering, including software testing area.

For the “Task T2” (Search in national reference curricula – in this case Brazil):

- Curriculum Guidelines to courses in Computing and Information Technology Area (CEEInf, 1999): curriculum proposed by the Ministry of Education that includes software testing theme without detailing content.
- Computer Brazilian Society Reference Curriculum for undergraduate courses in Bachelor of Computer Science and Computer Engineering (Sociedade Brasileira da Computação, 2005): it refers to software testing area, but without detailing content and depth.

Whereas national reference did not specify content for the software testing area, we carried out a documentary research on teaching plans (Fig. 1 – task T4). As criteria to define what teaching plans analyze, we seeked institutions with well-ranked graduate courses, so we selected the undergraduate programs of such institutions. We evaluated the education plans of 18 courses in computing area, covering 28 teaching plans, which provided the software testing education. The details of this research can be found at Benitti and Albano (2012).

With respect to certifications analysis (Fig. 1 – task T5) it was identified more than 10 certifications related to software testing area. Due to the large number of certifications, the research (Fig. 1 – task T6) was restricted to CBTS (Brazilian Certificate of Software Testing) created by ALATS (Latin American Association of Software Testing) (ALATS, 2011).

Regarding the search for standards/models (Fig. 1 – task T7) it was considered the MPS.BR (Brazilian Software Process Improvement) a mobilizing program, long-term, coordinated by the Association of Brazilian Software Excellence Promotion (SOFTEX), which has the support of the Ministry of Science and Technology (MCT). We consider the Implementation Guide in organizations of the Testing Factory type (Fig. 1 – task T8) (SOFTEX – Associação para promoção da excelência do software brasileiro, 2009).

As a result of this research we have the Table 1, identifying content versus source (Fig. 1 – task T9). In order to facilitate the organization of content and identify intersections, the content has been broken down into large topics (identified from the sources), in which case, the topics are: Fundamentals, Levels, Techniques, Types, Metrics and Process and automation. Each cell describes what the source (column) points on the topic (line).

3.1.2. *Phase 2 – Definition of Granularity*

This phase starts by defining the desired level for learning content. In this sense, we consider the learning level already set by SWEBOK (IEEE Computer Society, 2004) and perform a compatibility analysis with the other sources. When it was identified that it would be necessary to address different levels of learning, the contents were split up. For example, concerning the testing techniques, LO2 is at the level of understanding and LO2.2, LO2.3 and LO2.4 objects are in application level (Table 2).

Table 1

Researched sources and corresponding content

	Swebok (IEEE Computer Society, 2004)	**National curriculum** (Benitti & Albano, 2012)	**MPS.BR** (SOFTEX – Associação para promoção da excelência do software brasileiro, 2009)	**Certification** (ALATS, 2011)

Fundamentals	Testing-related terminology;
	Keys issues;
	Relationships of testing to other activities.
		Test principles.
		Terminology, objectives and basic concepts test.
		Verification and validation (V & V).
			Validation and verification concept.	Principle and test concepts;
				Understanding the test process.

Levels	Unit testing;
	Integration testing;
	System testing.
		Unit, integration and systems testing;
		Alpha, beta and acceptance testing;
		Scope tests;
		V & V in the model lifecycle.
			Unit testing;
			Integration testing;
			System testing.
				Contextualizing the test in the development lifecycle.

Techniques	Specification-based;
	Code-based;
	Fault-based;
	Usage-based;
	Based on nature of application;
	Selecting and combining techniques.
		Generation of test cases;
		Testing techniques: the notion of criteria and coverage;
		White box testing;
		Black box testing;
		Tests based on UML models;
		Code-based testing.
			Generation of test cases.	Concepts and testing techniques;
				Choosing techniques and test tools;
				Techniques for development of test cases.

Types	Acceptance;
	Installation;
	Alpha and beta;
	Functional;
	Reliability;
	Regression;
	Performance;
	Stress;
	Back-to-back;
	Recovery;
	Configuration;
	Usability
		Test human interface;
		Web application testing;
		Test quality requirements;
		Performance tests;
		Regression testing.
			Test types (without specifying);
			Regression testing.
				– nothing mentioned –

Metrics	Evaluation of the program under test; Evaluation of the tests performed.	Nothing mentioned	Estimates.	Test estimate (Test Point Analysis)

Process and automation	Test activities.	Management of the testing process;
		Registration and issue tracking;
		Tests design;
		TDD;
		Test planning;
		Software test documentation and maintenance;
		Automation and testing tools.
			Defining life cycle model;
			Automation.
				Understanding the testing process;
				Quality factors.
				Test mass;
				Assurance x control quality;
				Test Plan;
				Roles and Responsibilities;
				Input and Output artifacts;
				Risk analysis techniques.

Table 2
Definition of the learning objects

ID	Content	Levels by Bloom's Taxonomy*						Sources
		i	ii	iii	iv	v	vi	
LO1	1. Software Testing Fundamentals 1.1. Software test concept 1.2. Terminology related to testing 1.3. Verification and validation of concept 1.4. Understanding the testing process	▨	▨					Swebok / National curriculum / Certification / MPS.BR
LO2	2. Testing Techniques Overview of testing techniques	▨	▨					Swebok
LO2.1	2.1. Technical white box 2.1.1 Control flow graph (CFG) 2.1.2 Cyclomatic complexity	▨	▨	▨				Swebok / National curriculum / Certification / MPS.BR
LO2.2	2.2. Technical white box: control flow based criteria 2.2.1. Command testing 2.2.2. Condition/decision testing 2.2.3. Paths testing	▨	▨	▨				Swebok / National curriculum / Certification / MPS.BR
LO2.3	2.3. Technical white box: complexity based criteria 2.3.1. McCabe's criteria (basic path)	▨	▨	▨				Swebok / National curriculum / Certification / MPS.BR
LO2.4	2.4. Technical white box: data flow based criteria 2.4.1. All-definitions 2.4.2. All-uses	▨	▨	▨				Swebok / National curriculum / Certification / MPS.BR
LO2.5	2.5. Technical black box: equivalence partitioning and boundary value analysis	▨	▨	▨				Swebok / National curriculum / Certification / MPS.BR
LO2.6	2.6. Technical black box: cause-effect graphing and decision table	▨	▨	▨				Swebok / National curriculum / Certification / MPS.BR
LO2.7	2.7. Technical black box: test based on use cases	▨	▨	▨				Swebok / National curriculum / Certification / MPS.BR
LO3.1	3. Unit testing 3.1. 3.1.1. Characteristics 3.1.2. Unit test of OOP 3.1.3. Employing the techniques in the unit test 3.1.4. Automation	▨	▨	▨				Swebok / National curriculum / Certification / MPS.BR
LO3.2	3.2. Integration testing 3.2.1. Characteristics 3.2.2. Drivers and stubs 3.2.3. Integration strategies	▨	▨	▨				Swebok / National curriculum / Certification / MPS.BR
LO3.3	3.3. System and Acceptance testing 3.3.1. System testing 3.3.1.1. Automation 3.3.2. Acceptance testing	▨	▨	▨				Swebok / National curriculum / Certification / MPS.BR

Continued on next page

Table 2 – continued from previous page

ID	Content	Levels by Bloom's Taxonomy*						Sources
		i	ii	iii	iv	v	vi	
LO4	4. Types test (Objectives of Testing) 4.1. Regression testing 4.2. Usability testing 4.3. Performance testing 4.3.1. Volume 4.3.2. Stress 4.3.3. Timing 4.3.4. Tools / Automation 4.4. Configuration testing 4.5. Reliability testing 4.6. Recovery testing 4.7. Installation testing 4.8. Security testing	▨	▨	▨				Swebok / National curriculum / MPS. BR
LO5	5. Test process Overview of test process	▨	▨					Swebok
LO5.1	5.1. Test activities 5.1.1. Test planning 5.1.2. Roles and responsibilities 5.1.3. Specification 5.1.4. Execution 5.1.5. Artifacts	▨	▨	▨				Swebok / National curriculum / Certification
LO5.2	5.2. Test management 5.2.1. Risk analysis techniques	▨	▨	▨				Certification / MPS. BR
LO5.3	5.3. Test metrics 5.3.1. Evaluation of the program under test 5.3.2. Evaluation of the tests performed 5.3.3. Test estimate	▨	▨	▨	▨			Swebok / Certification / MPS.BR
LO5.4	5.4. TDD (Test Driven Development)	▨	▨	▨				National curriculum

* (i) Remembering; (ii) Understanding; (iii) Applying;
(iv) Analyzing; (v) Evaluating;(vi) Creating

After that we started planning each learning object (Fig. 1 – Task T13). At this stage we used an approach based on instructional design matrix proposed by Filatro (2008). Through this matrix in a comprehensive way the objectives, roles, tools, content, activities, assessments and the necessary environments can be organized. However, for the design of learning objects an adaptation of the matrix was carried out as detailed in Table 3.

When detailed planning was elaborated, we observed that the outline was extensive to the contents regarding the test levels. In this case, content was split up. (Fig. 1 – Task T14) generating the LO3.1, LO3.2 and LO3.3 objects (Table 2).

Table 3
Detailing elements of each learning object

Element	Goal
Units	Each unit is a learning object, defined by: Identifier (LO <<number>>), title and learning level (just as Bloom's Taxonomy).
Prerequisites (added)	Proposed element to describe the previous knowledge that students must have to get better use of the learning object content.
Goals	What is expected from each unit. You should use verbs in line with the level of learning defined by Bloom's taxonomy.
Contents	The contents identified are structured in topics, i.e there is a content refinement. Approach: Describes how each topic should be addressed, examples: video, interactive element, image, text, etc. Tools: It points out the tools necessary to prepare the topic.
Evaluation	Addressed in the course of the topics or the end of the learning object in the form of exercises. Recommendation – exercises should be in line with the level of learning proposed for LO.
Further Reading and next steps (added)	Proposed element to display links to the deepening of study, as well as guidelines for other related content.

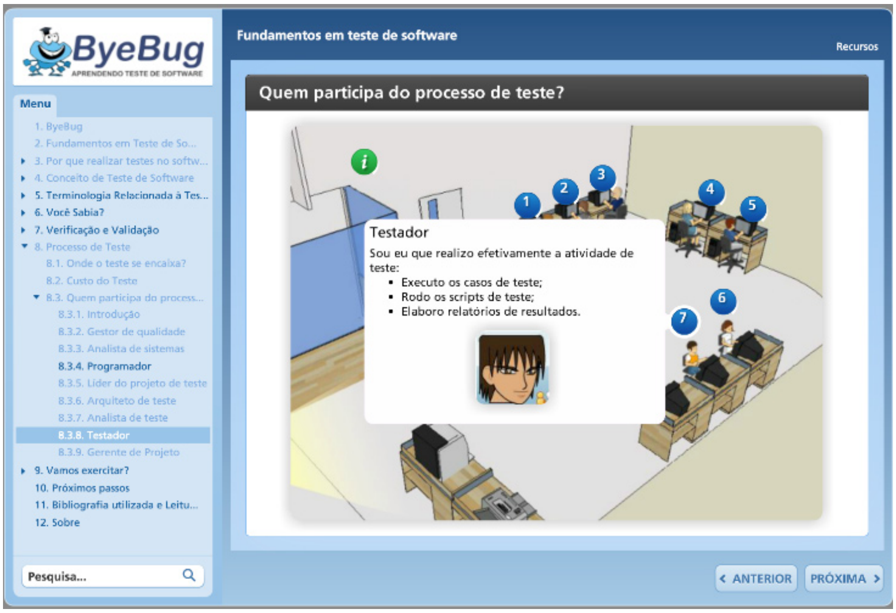
4. Production

This section illustrates how the planned learning objects (section 3) have been implemented. For the construction, Articulate Studio® was used as an authoring tool, helping to standardize. In addition, Articulate has tools for creating various types of questionnaires, narration and capturing screen actions to create video lessons. However, the main feature is that the tool allows the packaging of objects in the SCORM pattern (Advanced Distributed Learning, 2015). Thus, it is possible to distribute the object in a format compatible with the main LMS (Learning Management System) currently available. Fig. 2 and Fig. 3 illustrate the LOs produced in order to demonstrate the layout, resources and contents covered.

5. Evaluation

To evaluate the learning objects, students participated in experiments using the LO in some classes. We attempted to identify whether using the LO in a class was effective to achieve the goal of assisting in learning of software testing. In this context, we established the hypotheses in Table 4.

For the analysis of the hypotheses, five in-vivo experiments were performed, that is, the learning objects were applied in four different groups of universities and one group of IT professionals to assess the reusability in different contexts – classes for undergraduate students and training for professionals, and for different purposes – initial learning and improvement.

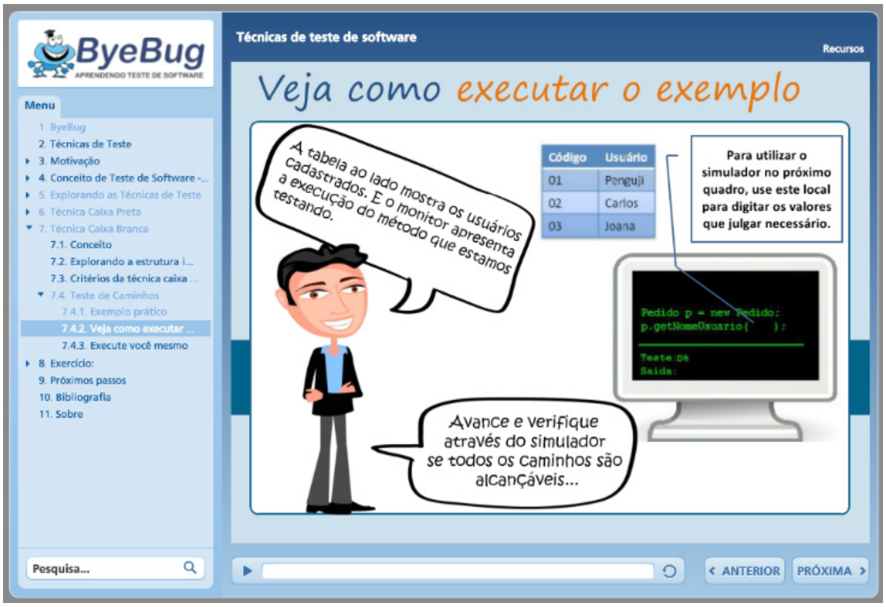

ByeBug
APRENDENDO TESTE DE SOFTWARE
Menu
1. ByeBug
2. Fundamentos em Teste de So...
3. Por que realizar testes no softw...
4. Conceito de Teste de Software
5. Terminologia Relacionada à Tes...
6. Você Sabia?
7. Verificação e Validação
8. Processo de Teste
8.1. Onde o teste se encaixa?
8.2. Custo do Teste
8.3. Quem participa do process...
8.3.1. Introdução
8.3.2. Gestor de qualidade
8.3.3. Analista de sistemas
8.3.4. Programador
8.3.5. Líder do projeto de teste
8.3.6. Arquiteto de teste
8.3.7. Analista de teste
8.3.8. Testador
8.3.9. Gerente de Projeto
9. Vamos exercitar?
10. Próximos passos
11. Bibliografia utilizada e Leitu...
12. Sobre
Pesquisa...
Fundamentos em teste de software
Recursos
ANTERIOR
PRÓXIMA
1
2
3
4
5
6
7

Quem participa do processo de teste?

Testador

Sou eu que realizo efetivamente a atividade de teste:
- Executo os casos de teste;
- Rodo os scripts de teste;
- Elaboro relatórios de resultados.

Fig. 2 (in Portuguese). Example of LO1 – Software Test Fundamentals (Topic: Who participates in the testing process? – The students clicking on a team member receives a description of their responsibilities).


ByeBug
APRENDENDO TESTE DE SOFTWARE
Menu
1. ByeBug
2. Técnicas de Teste
3. Motivação
4. Conceito de Teste de Software -...
5. Explorando as Técnicas de Teste
6. Técnica Caixa Preta
7. Técnica Caixa Branca
7.1. Conceito
7.2. Explorando a estrutura i...
7.3. Critérios da técnica caixa ...
7.4. Teste de Caminhos
7.4.1. Exemplo prático
7.4.2. Veja como executar ...
7.4.3. Execute você mesmo
8. Exercício:
9. Próximos passos
10. Bibliografia
11. Sobre
Pesquisa...
Técnicas de teste de software
Recursos
ANTERIOR
PRÓXIMA

Veja como executar o exemplo

A tabela ao lado mostra os usuários cadastrados. E o monitor apresenta a execução do método que estamos testando.

Código	Usuário
01	Penguji
02	Carlos
03	Joana

Para utilizar o simulador no próximo quadro, use este local para digitar os valores que julgar necessário.

```
Pedido p = new Pedido;
p.getNomeUsuario(    );

Teste:04
Saida:
```

Avance e verifique através do simulador se todos os caminhos são alcançáveis...

Fig. 3 (in Portuguese). Example LO2 – Test Techniques (Topic: White box – The student must include values to test a code that has been presented to him, receiving feedback from the coverage.)

Table 4

Hypotheses

Null hypotheses**	Alternative hypotheses
H01: Learning objects do not aid students to **learn** about software testing.	HA1: Learning objects aid students to learn about software testing.
H02: Learning objects do not aid the student a better **understanding** (Bloom's taxonomy) about software testing.	HA2: Learning objects aid the student a better understanding (Bloom's taxonomy) about software testing.
H03: Learning objects do not aid the student to **apply** (Bloom's taxonomy level) the concepts of software testing.	HA3: Learning objects aid the student to apply (Bloom's taxonomy level) the concepts of software testing.

** Trochim & Donnelly (2006) proposed that the hypothesis that supports the prediction is the alternative hypothesis, and the hypothesis that describes the remaining possible outcomes is the null hypothesis. They recommended using a notation like HA to represent the alternative hypothesis and H0 to represent the null case.

Table 5

Lesson plan of class #1 and equivalent learning objects

Content (defined by the Professor)	Equivalent LO (ID)
1. Fundamentals	LO1
1.1. Error, defect and failure	
1.2. Verification, validation and test	
1.3. Testing and debugging	
1.4. Testing process	
1.5. Stubs e Drivers	LO3.2
2. Functionality Test Levels	LO3.1
2.1. Unit testing	LO3.2
2.2. Integration testing	LO3.3
2.3. System testing	LO4
2.4. Acceptance testing	
2.5. Business cycle testing	
2.6. Regression testing	
3. Supplementary tests	LO4
3.1. Interface testing with user	
3.2. Performance testing	
3.3. Security testing	
3.4. Failure recovery testing	
3.5. Installation testing	
4. Structural testing	LO2
	LO2.1
4.1. Cyclomatic complexity	LO2.2
4.2. Control Flow Graph	
4.3. Independent paths	LO2.3
4.4. Test cases	
4.5. Multiple conditions	
4.6. Impossible paths	
4.7. Limitations	LO2.2
5. Functional Test	LO2
5.1. Equivalence partitioning	LO2.5
5.2. Boundary value analysis	

Table 6
Lesson plan of classes #2, #3, #4 and equivalent learning objects

Content (defined by the Professor)	Equivalent LO (ID)
1. Fundamentals	LO1
1.1. Software testing definition	
1.2. Error, defect and failure	
1.3. Verification and validation	
2. Testing Techniques	LO2
2.1. White box testing	LO2.1
2.1.1. data flow based criteria	LO2.2
2.1.2. complexity based criteria	LO2.3
2.2. Black box testing	LO2.5
2.2.1. Equivalence partitioning	
2.2.2. Boundary value analysis	
3. Test Levels	LO3.1
3.1. Unit testing	LO3.2
3.2. Integration testing	LO3.3
3.3. System testing	
3.4. Acceptance testing	
4. Types test	LO4
4.1. Regression testing	
4.2. Usability testing	
4.3. Performance testing	
4.4. Security testing	
5. Test process	LO5

The class #1 of Information Systems course at the Paranaense University (Brazil) was attended by 18 students and classes #2, #3 and #4 of Information Systems course at the Federal University of Santa Catarina (Brazil) was attended by 34, 17 and 27 students. Although the information system courses of the two universities are in-class mode, for the experiments, all the content was taught at a distance mode, based only on learning objects produced (as described in section 4). Table 5 presents the lesson plan defined by the class #1's Professor and Table 6 presents the lesson plan defined by the class #2, #3 and #4's Professor to address the issue of software testing and equivalence with the learning objects. Each related LO should involve, approximately, 1h of study.

The training for IT professionals was carried out in the distance format through the Moodle learning management system. The 40 vacancies offered for the course were filled. However, 27 professionals effectively started the course and only 8 concluded (Massive Open Online Courses retention rates overall are typically very low – figures of 10% retention are widely cited Hone & Ghada (2016)). In order to complete the course the participants should get an average of 6.0, each topic contemplating exercises weighing 1.0 and the final evaluation had weight 3.0. Table 7 presents the lesson plan defined by the Professor to the training for IT professionals to address the issue of software testing and equivalence with the learning objects.

Table 7

Lesson plan of training to IT professionals and equivalent learning objects

Content (defined by the teacher)	Equivalent LO (ID)
1. Fundamentals	LO1
1.1. Software testing definition	
1.2. Error, defect and failure	
1.3. Verification and validation	
1.4. Introduction to the test process	
2. Testing Techniques	LO2
2.1. White box testing	LO2.1
2.1.1. complexity based criteria	LO2.3
2.2. Black box testing	LO2.5
2.2.1. Equivalence partitioning	
2.2.2. Boundary value analysis	
3. Test Levels	LO3.1
3.1. Unit testing	LO3.2
3.2. Integration testing	LO3.3
3.3. System testing	
3.4. Acceptance testing	
4. Types test	LO4
4.1. Regression testing	
4.2. Usability testing	
4.3. Performance testing	
4.4. Security testing	
4.5. Reliability test	
4.6. Configuration test	
5. Test process	LO5
5.1. Overview of a traditional testing process	LO5.4
5.2. TDD	

All experiments involved a pretest and a posttest. The posttest contained some questions equal to the pretest and others similar to the same level of knowledge. The tests always involved more than one question for each level of learning proposed. For example, LO1 has the "Understanding" level of the Bloom taxonomy, so there were 3 questions about subject's LO1 at this level both in the pretest and in the posttest. Since LO2.1 has the "Application" level, there were 2 questions at the understanding level and 2 questions at the application level for this object, not only at the pretest but also at the posttest. Thus, each evaluation involved, approximately, 37 objective or open questions.

The experiments consisted of a first moment of a clarification about the research that would be carried out and the completion of a profile questionnaire. Subsequently, the students responded to the pretest to map the participants' initial knowledge. In the sequence, all the participants had the classes related to the subject of software testing carried out through the learning objects in the Moodle, that is, without Professor's exposure. Finally, the participants performed the posttest. Fig. 4 presents the results of the undergraduate students' tests, considering all the questions. In Fig. 5 and Fig. 6 we have the result of the questions of the level of "Understanding" and "Applying", for undergraduate students.

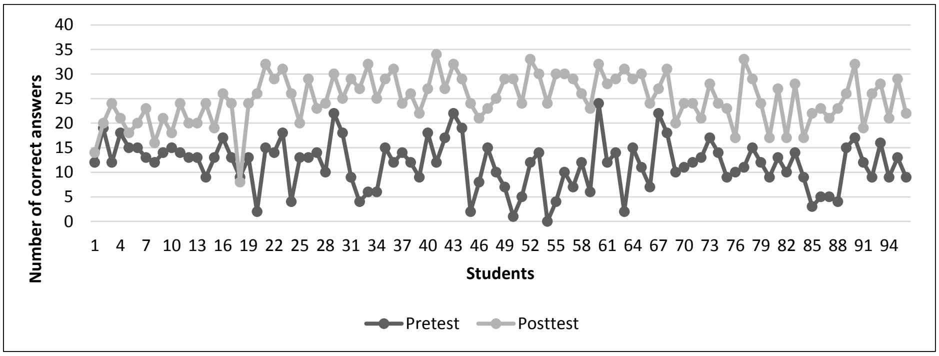


Fig. 4. Performance considering all the questions to undergraduate students.

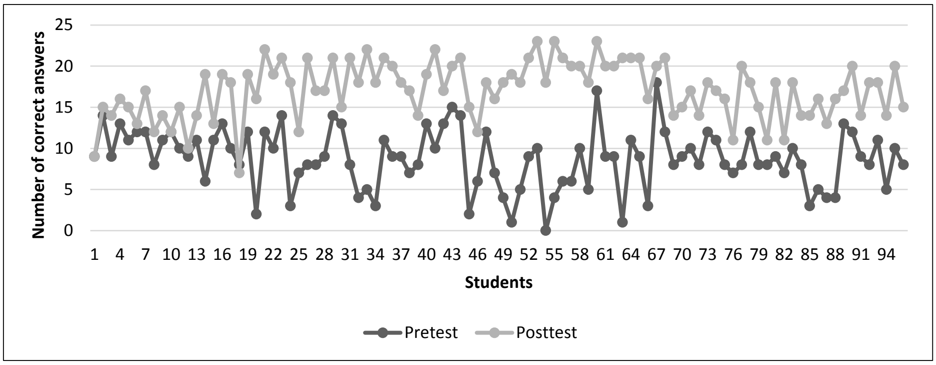


Fig. 5. Performance considering the «Understanding» level of the questions to undergraduate students.

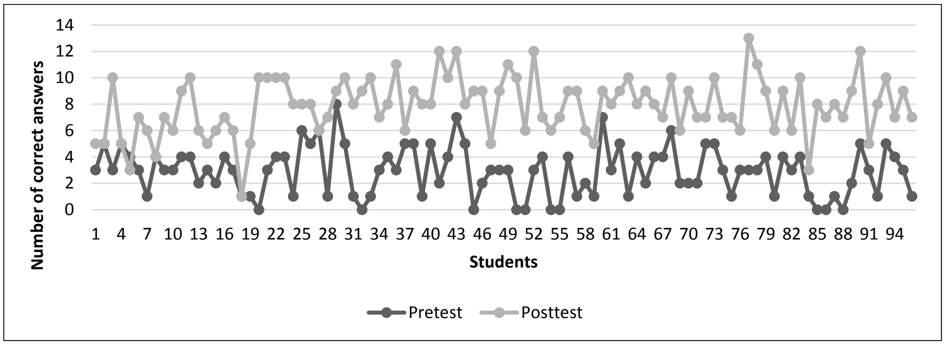


Fig. 6. Performance considering the application level of the questions to undergraduate students.

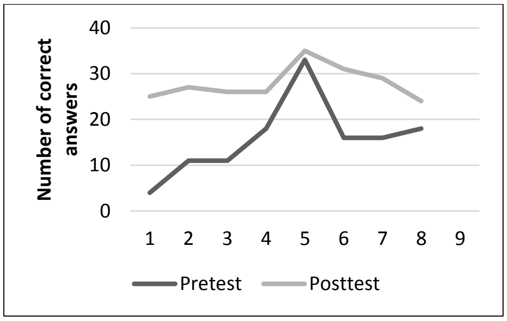


Fig. 7. Performance considering all the questions to IT professionals.

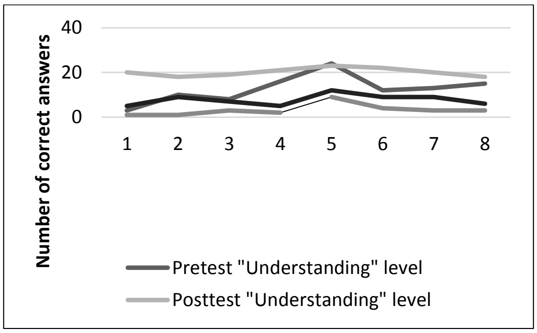


Fig. 8. Performance considering the "Understanding" and "Applying" levels of the questions to IT professionals.

Table 8
Statistical result of object learning

		Overall (HA1)	"Understanding" level (HA2)	"Applying" level (HA3)
Class #1 (n=18)	Z	-3.6582	-3.3869	-3.233
	p-value***	**0.00013**	**0.00035**	**0.00062**
Class #2 (n=34)	Z	-5.0862	-5.0862	-5.0119
	p-value***	**0**	**0**	**0**
Class #3 (n=17)	Z	-3.6214	-3.6214	-3.6214
	p-value***	**0.00015**	**0.00015**	**0.00015**
Class #4 (n=27)	Z	-4.5407	-4.5407	-4.5407
	p-value***	**0**	**0**	**0**
Training IT professionals (n=8)	W	0	1	0
	p-value	The critical value of W for N = 8 at $p \leq 0.01$ is 1. Therefore, the result is significant at $p \leq 0.01$.	The critical value of W for N = 8 at $p \leq 0.01$ is 1. Therefore, the result is significant at $p \leq 0.01$.	The critical value of W for N = 8 at $p \leq 0.01$ is 1. Therefore, the result is significant at $p \leq 0.01$.
All experiments (n=104)	Z	-8.848	-8.7477	-8.6315
	p-value***	**0**	**0**	**0**

*** result is significant at $p \leq 0.01$.

The graphs in Fig. 7 and Fig. 8 show the results of the IT professionals who completed the training.

Wilcoxon signed-rank test was used for the statistical analysis of the experiments results, being this one indicated to compare two samples which is the case of the evaluation carried out. Table 8 shows the comparison of the pretest results with the posttest of the experiments, observing: (i) the overall result considering the performance in all questions (HA1); (ii) the result considering only the questions of understanding level (HA2); (iii) the outcome of application-level issues (HA3).

From the result obtained, we can observe that all hypotheses in all experiments the value of p-value is lower than the level of significance adopted ($p <= 0.01$). Therefore, the result allows to reject H01, H02 and H03 and to accept HA1, HA2 and HA3, that is, it can be concluded that for the sample considered, the learning objects allow the student to learn about software testing, considering both levels of understanding and application.

Regarding threats to validity, we had the fact that the students are not present at the time of the assessments application. This threat was minimized in undergraduate classes because the content was addressed in a curricular way, the pretests and posttest were answered individually in the classroom. Regarding the internal threat to the validity of the results, the participants can study other materials than LOs. However, a question about this situation was included in the posttest and there was no report. As an external threat to the evaluation, there is the possibility of the participant having prior knowledge or experience in the subject studied. Based on the profile results and pretest questionnaire, this aspect can be dealt with – there is no need to exclude any participants.

In all the experiments the students studied the subject completely from the learning objects and they could have the Professor's support to solve any questions. In a perception survey of the activity, answered freely by the participants, we obtained the results shown in Fig. 9 and Fig. 10. Most students considered motivating to study through the

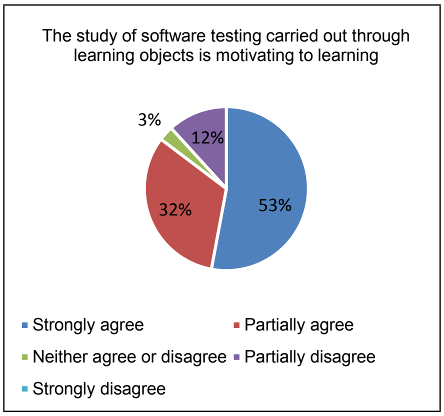


Fig. 9. Perception of participants regarding motivation.

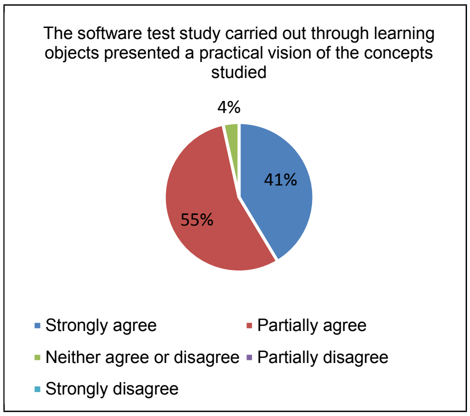


Fig. 10. Perception of participants regarding practical vision

learning objects, as well as considered that the objects presented a practical view of the concepts studied.

To date, three professors successfully use learning objects in their classrooms. Equivalences between the lesson plan and the LOs were possible in all cases – it can be seen in Table 5, Table 6 and Table 7 that all content provided by professors can be met by learning objects. However, in some cases the contents of the LOs exceeded what was requested, for example, content regarding the test types – but the professors considered the objects content to be appropriate and did not oppose to use them.

6. Conclusion

This paper presented a methodology to assist in the definition of content and granularity of learning objects. The methodology allowed to base the definition of the granularity of the objects, allowing the reuse in different contexts – courses of distinguished universities and professors, as well as a training for IT professionals. This context of reuse is shared by Downes (2001) for whom the reuse "focuses on sharing between institutions and on sharing materials on particular topics between those who teach these topics". In this view, the estimated use by the intended audience is the key factor that determines LO size. This principle of common use states that pieces that will always be used together by the intended audience should be kept together. Only if components will be used separately should they be developed into separate LOs (Schoonenboom, 2012). In this sense, the proposed methodology supports the definition of the granularity, guiding the production of the objects, because the content to be approached in each LO was well delimited.

The viability of the methodology was presented through a case study in software testing area, presenting the planning, production and evaluation of a set of learning objects. Five experiments, in different contexts, were carried out, involving 104 students/professionals. Statistical tests allow us to verify that the learning objects developed favored learning, both at the level of understanding and application. Besides, most students considered motivating to study through learning objects, as well as considered that the objects presented a practical view of the concepts studied.

As future work, we intend to reapply the methodology in another area of knowledge, aiming to validate its effectiveness and/or refine it.

Acknowledgements

This work was supported by the CNPq (Conselho Nacional de Desenvolvimento Científico e Tecnológico – www.cnpq.br), an entity of the Brazilian government focused on scientific and technological development.

References

ACM/IEEE Computer Society. (2013). *Computer Science Curricula 2013.* DOI:10.1145/2534860

ACM, AIS, IEEE-CS. (2005). *Computing Curricula 2005*. Retrieved from `http://www.acm.org/education/curric_vols/CC2005-March06Final.pdf`

Advanced Distributed Learning. (2015, 10 23). SCORM. Alexandria, VA, USA. Retrieved from `http://www.adlnet.gov/adl-research/scorm/`

ALATS. (2011). *Manual do Candidato à Certificação Brasileira de Teste de Software – CBTS.* Retrieved from ALATS (Associação Latino-Americana de Teste de Software): `http://www.alats.org.br/portal/images/cbts/manual_candidatos_cbts.pdf`

Anderson, L., Krathwohl, D. (2001). *Bloom's taxonomy revised.* Retrieved from `http://thesecondprinciple.com/teaching-essentials/beyond-bloom-cognitive-taxono-my-revised`

Benitti, F.B., Albano, E. (2012). Teste de Software: o que e como ensinar? *XXX Congresso da Sociedade Brasileira de Computação.* Curitiba.

Braga, J., Dotta, S., Pimentel, E., Stransk, B. (2012). Desafios para o desenvolvimento de objetos de aprendizagem reutilizáveis e de qualidade. *DesafIE – Workshop de Desafios da Computação aplicados à Educação.* Curitiba.

CEEInf. (1999). *Diretrizes Curriculares de Cursos da Área de Computação e Informática.* Retrieved from Comissão de Especialistas de Ensino de Computação e Informática. Secretaria de Educação Superior do MEC (SESu/MEC). `ftp://ftp.inf.ufrgs.br/pub/mec/diretrizes.doc`

Chen, T. (2004). Experience with teaching black-box testing in a Computer Science/Software Engineering Curriculum. *IEEE Transactions on Education,* 47(1).

Dias Neto, A., Natali, A., Rocha, A., Travassos, G. (2006). Caracterização do estado da prática das atividades de teste em um cenário de desenvolvimento de software brasileiro. *Simpósio Brasileiro de Qualidade de Software*, 27–41.

Downes, S. (2001). Learning Objects: resources for distance education worldwide. *International Review of Research in Open and Distributed Learning,* 2. DOI: `http://dx.doi.org/10.19173/irrodl.v2i1.32`

Downes, S. (2003). Design and reusability of learning objects in an academic context: A new economy of education. *USDLA Journal,* 17.

Filatro, A. (2008). *Design Instrucional na Prática.* . Prentice-Hall.

Flôres, M., Santos, L., Tarouco, L. (2007). Objeto de aprendizagem: teoria instrutiva apoiada por computador. *Porto Alegre*, *Revista Novas Tecnologias na Educação*, 6(2).

Gibbons, A., Nelson, J., Richards, R. (2000). The nature and origin of instructional objects. . In: D.A. WILEY, *The Instructional use of Learning Objects.* Bloomington: Association for Educational Communications and Technology.

Heyer, S. (2005). An analysis of learning resources using a cognitive process taxonomy. *Interactive Computer-Aided Learning Conference.*

Hone, K., Ghada, G. (2016). Exploring the factors affecting MOOC retention: a survey study. *Computers & Education,* 98, 157–168. DOI: `http://dx.doi.org/10.1016/j.compedu.2016.03.016`

IEEE Computer Society. (2004). *SWEBOK – Guide to the Software Engineering Body of Knowledge, 2004 Version.* Retrieved from `http://www.computer.org/portal/web/swebok`

Learning Technology Standards Committee. (2002). *Draft Standard for Learning Object Metadata, IEEE 1484.12.1-2002. Institute of Electrical and Electronics Engineers, Inc. LTSC.* Retrieved from Learning technology standards committee website.

Moisey, S., Ally, M. (2013). Realizing the promise of learning objects. In: M.G. Moore, *Handbook of Distance Education.* New York: Taylor & Francis.

Myers, G., Badgett, T., Sandler, C. (2012). *The art of Software Testing.* (3 ed.). Hoboken: John Wiley & Sons.

Noor, S., Yusof, N., Hashim, S. (2011). Creating granular learning object towards reusability of Learning Object in e-learning context. *International Conference on Electrical Engineering and Informatics (ICEEI).* DOI:10.1109/ICEEI.2011.6021645

Object Management Group. (2015). *Business Process Model and Notation.* Retrieved from Object Management Group: `http://www.bpmn.org/`

Ochoa, X. (2011). Learnometrics: metrics for learning objects. In: *1st International Conference on Learning*

Analytics and Knowledge . New York: ACM. DOI: 10.1145/2090116.2090117

Polsani, P.R. (2006, feb). Use and abuse of reusable learning objects. *Journal of Digital Information, 3*. Retrieved from `https://journals.tdl.org/jodi/index.php/jodi/article/view/89/88`

Salzer, H., Haberman, B., Yehezkel, C. (2012). The scientific method and software testing integrated into the same lesson. In: *17th ACM annual conference on Innovation and technology in computer science education*. New York: ACM, 385. DOI:10.1145/2325296.2325402

Schoonenboom, J. (2012). Four scenarios for determining the size and reusability of learning objects. *Australasian Journal of Educational Technology,* 28(2), 249–265.

Sociedade Brasileira da Computação. (2005). *Currículo de Referência da SBC para Cursos de Graduação em Bacharelado em Ciência da Computação e Engenharia da Computação.* Retrieved from `http://www.sbc.org.br/documentos-da-sbc/category/131-curriculos-de-referencia`

SOFTEX – Associação para promoção da excelência do software brasileiro. (2009). *MPS.BR – Melhoria de Processo do Software Brasileiro: Guia de Implementação – Parte 10: Implementação do MR-MPS em organizações do tipo Fábrica de Teste.* Retrieved from `http://www.softex.org.br`

Stroustrup, B. (2010). Viewpoint: What should we teach new software developers? Why?. *Commun. ACM.* 53(1), 40–42. DOI: `http://dx.doi.org/`10.1145/1629175.1629192

Trochim, W., Donnelly, J. (2006). *Research Methods Knowledge Base* (3 ed.). Mason, OH: Thomson.

Wiley, D. (2001). Connecting learning objects to instructional design theory: a definition, a metaphor, and a taxonomy. In: D.A. WILEY, *The Instructional Use of Learning Objects.*

F.B.V. Benitti is Professor in the Department of Informatics and Statistics (INE) at the Federal University of Santa Catarina (UFSC). Her primary research area relates to software engineering and computer education.