SERIF:
A Semantic ExeRcise Interchange Format

Ben De Meester, Hajar Ghaem Sigarchian, Tom De Nies, Ruben Verborgh, Frank Salliau, Erik Mannens, and Rik Van de Walle

Ghent University – iMinds – Multimedia Lab, Belgium
`{firstname.lastname}@ugent.be`

Abstract. More and more users obtain new knowledge using e-learning systems, and often assess their understanding of this new knowledge using corresponding assessment items. However, the distribution of content items and assessment items in a learning object is tightly bound. To publish assessment items, independently of the corresponding content items, it is required to wrap these assessment items into separate learning objects, which introduces a large overhead. Moreover, current learning objects are closely coupled with their execution environment. A stand-alone and lightweight format to describe assessment items is needed. This way their publication is facilitated and their discoverability can be increased. This paper proposes some important features for such a format and introduces SERIF: a Semantic ExeRcise Interchange Format, whose underlying data model is based on the QTI data model, and that can be distributed into different RDF serializations. SERIF was applied successfully in three proof-of-concept applications, where we assessed how SERIF is decoupled from the execution environment and extendable to other content and interaction types. In future work, we will exploit its defined semantics for automatic discovery and combination of assessment items.

Keywords: Assessment Item, JSON-LD, Learning Object, Linked Data

1 Introduction

Apart from print-based learning, more and more users obtain new knowledge on the Web using *e-learning* systems [5]. This knowledge is distributed by e-learning systems using *learning objects*. Learning objects are small, self-contained, and re-usable units of learning that comprise, amongst others, *content items* (used to instruct new knowledge, e.g., a mathematical theory) and *assessment items* (used to assess the learner's understanding of the new knowledge, e.g., mathematical equations that need to be solved by the learner) [8].

E-learning increases the need for available learning objects drastically. To cope with this increasing need, it becomes more and more important to efficiently and effectively discover and (re-)use available learning objects on the Web. To optimize reuse, individual content items and assessment items should be distributed separately. However, current learning object formats are built to exchange content items and assessment items together, leading to the following disadvantages:

- The exchange of individual content items or assessment items would require **wrapping those items in a learning object**, which introduces a large overhead. This is especially true for assessment items, as assessment items generally require more user interaction, but comprise less content.
- Execution environments need to support the **rendering of both the content items and the assessment items**. These environments thus become very complex as content items can be very diverse (e.g., a history lesson vs. a digital chemistry lab can impose very different technical requirements for the execution environment).
- Since these execution environments are already very complex, current learning object formats are very rigid and **do not allow any extensions**, neither in terms of content types (e.g., plain text vs. LaTeX), nor in terms of interaction types (e.g., multiple choice vs. a digital drawing canvas).
- Due to this complexity, these learning objects can thus only be used by the supporting execution environments, creating a **vendor lock-in**. This decreases its discoverability and reuse.

In this paper, we introduce *SERIF*, a Semantic ExeRcise Interchange Format. SERIF is a lightweight exchange format for describing assessment items that are not restricted in terms of content types and interaction types. Its underlying data model is semantically specified, which means SERIF inhibits the advantages of semantic technologies, mostly in terms of discoverability and reuse, but it is also backwards compatible with non-semantic agents through use of, e.g., JSON-LD as serialization format. This enables the development of e-learning systems that can automatically combine relevant assessment items (i.e., *mashup* systems).

After presenting the state of the art in Section 2, we propose, in Section 3, a list of desired features for an assessment item data model to solve aforementioned problems. In Section 4, we introduce the assessment item format SERIF. In Section 5, we evaluate whether SERIF complies with the proposed features, and we give concluding remarks in Section 6.

2 State of the Art

In Figure 1, we provide a high-level overview of a current e-learning system and the learner's interaction with it. A learner interacts with the client-side view of a Learning Management System (LMS), and fetches a learning object. This learning object is fetched, and rendered using the execution environment. All actions that the learner does with respect to the learning object are tracked inside the learning object, and passed to the execution environment. These activities need to be tracked by the learning object itself, as the content of a learning object can be so diverse, that the execution environment cannot cope with all possible activities for all possible content. The execution environment then sends these activity statements to the Learning Record Store (LRS). In the remainder of this section, we give an overview of the currently most prevalent technologies. Afterwards, we take a look into existing assessment item formats, and learning object metadata.

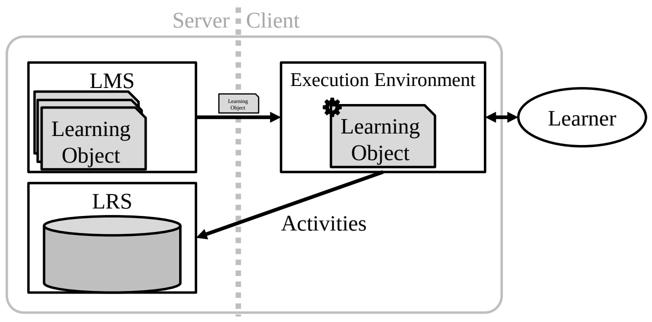


Fig. 1: Overview of a current e-learning system: learning objects are hosted on an LMS, sent to the learner. The execution environment renders the learning object. Learner activities are intercepted by the learning object, passed to the execution environment, and sent to an LRS.

LMS The Learning Management System (LMS) is where all learning objects are stored and managed [6]. The LMS is also responsible for other administrative tasks, e.g., user management. Two prevalent LMSs are Moodle[1] and Blackboard[2]. Given the many responsibilities of an LMS, creating and installing such a system introduces a lot of effort, which increases the costs for instructors [7].

Learning Object A learning object is a self-contained and re-usable unit of learning that comprises, amongst others, content items and assessment items. The first learning object format was specified by the Aviation Industry CBT Committee (AICC), which has been discontinued since December 2014. After that, two competing standard formats emerged. On the one hand, the Advanced Distributed Learning initiative (ADL) has specified the Shareable Content Object Reference Model (SCORM) [3], the de facto learning object standard. SCORM defines the functional requirements of the learning object, and thus, also the functional requirement of the execution environment (dubbed the SCORM driver) that renders the learning object. This couples the specification of the learning object with the specification of the execution environment. On the other hand, IMS Global has defined the Common Cartridge (CC) specification [2]. CC is very similar to SCORM. However, while SCORM only defines the functional requirement of the learning object, CC also defines the data model

[1] `https://moodle.org/`
[2] `http://www.blackboard.com/`

and format of the content items and the assessment items. Both specifications result in a very coupled, complex system that locks in users, and does not allow any extensions.

Execution environment The execution environment is the client-side application where a learner interacts with learning objects (note that the execution environment is generally incorporated in the front-end of the LMS). The execution environment fetches learning objects from the LMS, and renders them for the learner. After it is rendered, the learning object catches the activities of the learner with respect to the items in the learning object, e.g., solving a multiple choice assessment, or watching an explanatory movie. SCORM and CC specify which API calls need to be executed when a certain learning activity is performed. The SCORM driver needs to expose these API calls to the learning object, after which the SCORM driver sends the activities intercepted by the learning object to the LRS. Whilst these processing requirements for learning objects may be necessary for heterogeneous content items, this is not necessary for assessment items, as assessment items are a lot more structured.

LRS The Learning Record Store (LRS) is the database where all activities are stored that the learner performs in respect to the learning object. As such, the storage of learning activities is decoupled from the storage of the learning objects. Whereas originally the format to describe learning activities was integrated into SCORM, this has been decoupled with the advent of the Tin Can API[3]. The Tin Can API is built on top of Activity Streams [9], to publish learning activities without the need for an LMS or a SCORM Driver. The Tin Can API allows any data logging in the form of the triple actor-verb-action, i.e., "I – did – this". By decoupling the format of learning activities from the rest of the learning object, Tin Can API is a lot more flexible, and as such, more applicable in general.

Assessment item formats As mentioned before, IMS Global specifies within its learning object model (i.e., CC) also the data model and format of a content item and its corresponding assessment item. The Question and Test Interoperability specification (QTI)[4] by IMS Global is a data model in XML format that specifies the structure of assessment items, how to visualize them, and how to process the learners actions. QTI is also used outside CC. It is a very coupled specification with very little room to make adjustments. The visualization of QTI questions is handled with a custom interpretation of HTML. A subset of the current HTML-tags can be used, but for example, the QTI `<p>`-tag has a custom definition, and not the official definition by W3C. The latest version of the QTI specification is from August 2012, making this quite outdated compared to current Web Standards.

[3] `http://tincanapi.com/`
[4] `http://www.imsglobal.org/question/`

The most prevalent assessment item formats that are not coupled with an execution environment are GIFT[5], Aiken[6], and the importing format as described by Blackboard[7]. These formats are very lightweight, however, also very limited in capabilities. They are specified in a custom format in plain text. The support for multimedia is limited and cannot be extended, while only a limited set of assessment item interaction types are supported. These formats are usually used to exchange simple multiple choice questions between similar LMSs, e.g., two Moodle instances.

LOM the Learning Object Metadata specification (LOM) [1] is the prevalent specification to add metadata to learning objects. LOM is an IEEE standard that defines the structure of the metadata to annotate and classify learning objects. Most LMSs and learning object formats, such as Moodle and SCORM, support LOM. LOM has been converted to an ontology[8], which, combined with its widespread usage, makes this specification a good candidate to also annotate individual assessment items.

3 Proposed Features for the Data Model

Building upon and extending the state of the art, we propose the following features to achieve a generic data model for describing assessment items. Such a data model that describes assessment items needs to...

be decoupled from the execution environment To simplify the requirements for both the data model and the execution environment, no processing instructions should be specified inside the assessment item data model. This decouples the data model from the execution environment, and eases interoperability. This is a similar technique that has been used in, for example, the digital publishing domain, where the standard that defines the format of digital publications – EPUB 3 – only specifies its structure and media type of the content (namely, HTML). The rendering and processing of the content is being handled purely by the e-reader. Thus, the data model should only consist of the content and structure of the assessment item. This way, the same assessment item can be processed and presented whether the execution environment is a mobile app, a JAVA desktop application, or a Web application.

support multiple media types Plain text is not enough to visualize all types of assessments. Taking advantage of Web 2.0, interactions should be possible on other media types such as images, video, and audio. Restricting the supported media types would not be robust against new emerging media types.

[5] `https://docs.moodle.org/29/en/GIFT_format`
[6] `https://docs.moodle.org/29/en/Aiken_format`
[7] `https://help.blackboard.com/en-us/Learn/9.1_SP_10_and_SP_11/Instructor/070_Tests_Surveys_Pools/106_Uploading_Questions#file_format`
[8] `http://data.opendiscoveryspace.eu/lom_ontology_ods.owl`

Thus, the data model should allow specifying multiple media types, and it should also be possible to extend these supported media types.

support multiple interaction types With the advent of touch-based devices, the amount of interaction types has been increased (e.g., drag-and-drop, or drawing on a canvas). The possible interaction types should thus not be restricted. This does not imply that every execution environment should be able to render any kind of interaction type incorporated in a assessment item, just that the data model should not define which interaction types can be used. Multiple interaction types should be allowed, and the supported interaction types should be extendable.

enable discoverability and reuse To prevent vendor lock-in, the assessment items should be described in a way that they can be automatically discovered and exchanged easily between execution platforms. Then, it becomes possible to generate ad-hoc assessments.

4 SERIF

We introduce SERIF: a Semantic ExeRcise Interchange Format[9]. SERIF's underlying data model is based on the QTI data model. However, there are three notable differences between SERIF and QTI.

Decoupled Model Whereas QTI contains extensive means of describing ways of processing, evaluating, and scoring assessment items, SERIF is only used to model the content of an assessment item. By decoupling the content from the execution, the conformance requirements for the execution environment are also lowered, and the interoperability of the assessment items is increased.

Consistent visualization Although to its users, the QTI `<itemBody>` element appears to support HTML elements, these are actually elements in the QTI namespace, without any universally accepted guidelines on how to render them. By not defining any implicit namespace, but always requiring explicit MIME types (e.g., 'text/html'), SERIF does not exhibit this ambiguity when it comes to rendering. Also, this way, the visualization and definition of the content is separated better. MIME types are not restricted to a certain technology stack, which improves the interoperability of SERIF.

Specified semantics Whereas the QTI model is closely coupled with its XML serialization, we propose a generic underlying model with specified semantics. Based on these semantics, content can be described in the Resource Description Framework (RDF) [4]. RDF is a specification to describe data as triples (subject–predicate–object). It is the open standard as defined by W3C for representing data in a machine-interpretable format. Being a very generic and flexible model, data described in RDF is intrinsically modular,

[9] The full specification is available at `http://edutab.test.iminds.be/specs/serif/`, we also provide an ontology at `http://semweb.mmlab.be/ns/serif` and a context document at `http://edutab.test.iminds.be/specs/serif/context.jsonld` for a possible JSON-LD serialization.

distributable, and easily discoverable. Multiple RDF serializations can be defined that keep consistent machine-interpretable meaning.

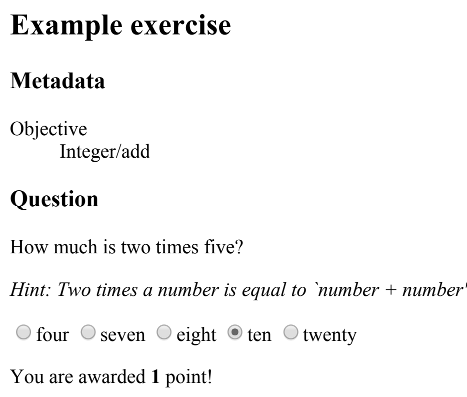
Compared to the current e-learning system as depicted in Figure 1, SERIF does not require new modules to be added, but lowers the requirements of assessment item execution environments, and allows for automatic discovery, instead of user-driven selection of relevant assessment items.

In Table 1, we give an overview of the hierarchy of the underlying data model of SERIF. In Figure 2, an exemplary multiple choice assessment item is shown. Each *AssessmentItem* contains zero or one *ItemBody* (that defines the question, e.g., "How much is two time five?"), zero or more *Items* (that define the interactions, e.g., multiple choice), and zero or more *InfoControls* (that define additional content, e.g., the hint that "Two times a number is equal to 'number + number"'). Each *Item* could have zero or more *Options* to choose from (e.g., 'four', 'seven', 'eight', 'ten', and 'twenty'), and zero or one *ScoreMaps* that map answer values to points (e.g., the value 'ten' is worth one point). All classes of the data model that contain a value (e.g., *ItemBody*, *Option*, and *InfoControl*), contain a `value` and a `value-mimetype`, where `value-mimetype` types the value. SERIF does not restrict values to one media type this way, and is robust against not yet existing MIME types. For example, the 'text/plain' question `How much is two times five?` can also be described in 'text/html' as `<p>How much<br/>is two times five?</p>`, to allow for different visualizations.

class	**description**
AssessmentItem	An assessment item, contains an *ItemBody*, a set of *Items*, and a set of *InfoControls*.
ItemBody	The actual content of the question.
Item	An interactivity for the user used to answer the question. Each *Item* has a certain interactivity type, and has optionally a set of *Properties*, a set of *Options*, and a *ScoreMap*. A *Property* is an additional configuration depending on the interactivity type of the *Item*[a], each *Option* is a possible answer, and the *ScoreMap* maps answer values to their respective scores.
InfoControl	Additional content of any kind. *Infocontrols* can be referenced in the *ItemBody* or in individual *Items* by referencing the `infoControlIdentifier`. The most used subclasses of *InfoControl* are *Hint* (i.e., supplementary content when no answer is yet given) and *Feedback* (i.e., additional content based on a given answer).

[a] For a full description of all properties of the basic interaction types we refer to the *QTI specification*

Table 1: Overview of the different classes in the data model, and their description.



Example exercise

Metadata

Objective
Integer/add

Question

How much is two times five?

Hint: Two times a number is equal to `number + number'

four seven eight [✓ ten] twenty

You are awarded **1** point!

Fig. 2: Visual example of a simple multiple choice assessment item

The metadata in SERIF is defined using key-value pairs as defined by LOM. Other metadata specifications could also be used, if needed. Therefore, we prefixed all LOM-specific key-value pairs (`lom:`). This way, no collisions are possible with other potential metadata schemas. As such, we can define LOM metadata on each individual assessment item, which was not possible in current learning object formats, as they define LOM metadata on the entire learning object, and not on the individual items (e.g., classifying the objective of the assessment item to be improving the skill of adding integers). This metadata (i.e., classification and keywords) can be used to discover appropriate assessment items automatically.

The example of Figure 2 is described in SERIF in Figure 3, in two possible serializations. Figure 3a shows the JSON-LD serialization [10]. JSON-LD has the advantages of JSON (i.e., compact, human-readable, easily handled by most programming languages), but retains the semantic interpretability via its `@context` document. Being based on JSON, it is easily processed by web clients, and objects are easily distributed. This also allows non-semantic systems to use the JSON-LD serialization as if it were JSON objects, with no prior knowledge needed about the Semantic Web. Figure 3b shows the semantic interpretation of the assessment item, by serializing the same exercise in Turtle, a human-readable serialization of RDF data. Turtle more closely resembles the RDF model (i.e., the format more closely resembles the triples), which could be beneficial for semantic agents, but might involve a learning curve for non-semantic developers. Intelligent agents could request assessment items based on their available metadata, which enables ad-hoc quiz generation applications.

```
{

"@context": "http://edutab.test.iminds.be/specs/serif/context.jsonld",


"identifier":                 "http://www.example.com/test1",
"type":                       "AssessmentItem",
"title":                      "Example exercise",
"lom:educationalLanguage":    "en",
"lom:educationalDifficulty":  "easy",
"lom:classification": [{
  "lom:classificationPurpose": "educational objective",
  "lom:taxonPath": {
    "lom:classificationTaxon": {"lom:taxonEntry": "Integer/Add"}
  }
}],
"itemBody": {
  "value": "<p>How much is two times five?<br />
           {{http://www.example.com/interaction1}}</p>",
  "value-mimetype":        "text/html"
},

"infoControl": [{
  "type": "Hint",
  "identifier": "http://www.example.com/infocontrol1",
  "value": "Two times a number is equal to 'number + number'"
}],
"item": [{
  "type":                  "ChoiceInteraction",
  "identifier":            "http://www.example.com/interaction1",
  "infoControlIdentifier": "http://www.example.com/infocontrol1",
  "properties": [
    { "key": "maxChoices", "value": 1       },
    { "key": "shuffle",    "value": false    }
  ],
  "scoreMap": [
    { "value": 10,         "score": 1       }
  ],
  "options": [
    { "label": "four",     "value": 4       },
    { "label": "seven",    "value": 7       },
    { "label": "eight",    "value": 8       },
    { "label": "ten",      "value": 10      },
    { "label": "twenty",   "value": 20      }]
}]}
```

(a) JSON-LD serialization

```
@prefix  serif: <http://semweb.mmlab.be/ns/serif#>.
@prefix  ex: <http://www.example.com/>.
@prefix  schema: <http://schema.org/>.
@prefix  lom: <http://data.opendiscoveryspace.eu/lom_ontology_ods.owl#>.
@prefix  xsd: <http://www.w3.org/2001/XMLSchema#>.

ex:test1 a                       serif:AssessmentItem;

  schema:name                    "Example exercise";
  lom:educationalLanguage        "en";
  lom:educationalDifficulty      "easy";
  lom:classification [
    lom:classificationPurpose   "educational objective";
    lom:taxonPath [
      lom:classificationTaxon [ lom:taxonEntry      "Integer/Add". ]
    ]
  ];
  serif:itemBody [
    schema:object """<p>How much is two times five?<br />
        {{http://www.example.com/interaction1}}</p>""";
    schema:contentType          "text/html"
  ];
  serif:item                     ex:interaction1.

ex:infocontrol1 a                serif:Hint;

  schema:object "Two times a number equals 'number + number'".


ex:interaction1 a                serif:ChoiceInteraction;

  serif:infoControl              ex:infocontrol1;
  serif:properties
    [ schema:name "maxChoices"; schema:object "1"^^xsd:integer     ],
    [ schema:name "shuffle";    schema:object "false"^^xsd:boolean ];

  serif:scoreMap
    [ serif:score               "1"^^xsd:decimal;
      schema:object             "10"^^xsd:integer                  ];
  serif:option
    [ schema:name "four";       schema:object "4"^^xsd:integer     ],
    [ schema:name "seven";      schema:object "7"^^xsd:integer     ],
    [ schema:name "eight";      schema:object "8"^^xsd:integer     ],
    [ schema:name "ten";        schema:object "10"^^xsd:integer    ],
    [ schema:name "twenty";     schema:object "20"^^xsd:integer    ].
```

(b) Turtle serialization

Fig. 3: A basic example showing an assessment item. Figure 3a and Figure 3b are aligned, and indicated from top to bottom with a vertical line are: the metadata, the body, a hint, and the interaction with five choices.

5 Evaluation

To evaluate SERIF, we will first evaluate SERIF functionally by comparing SERIF with the state of the art. Second, we will evaluate SERIF given the use case where SERIF was used as common format between two IT companies and four publishers.

In Table 2, you can find a functional comparison between the state of the art formats and SERIF, namely: (i) whether the format is coupled with the execution environment, (ii) how many content types are supported, (iii) how many interaction types are supported, (iv) whether the format is easily integrated, i.e., can easily be discovered and reused, and (v) the relative overhead the format introduces. Specialized content formats such as GIFT, Aiken, and the Blackboard format excel in terms of simplicity and have thus low overhead, and are integrated more easily. However, these formats are the least extendable with other interaction types, and do not allow other media types for the content of the assessment items. QTI introduces a lot more overhead, as QTI also models the processing instructions. It is also strongly coupled with the execution environment. CC comprises QTI, and as such has the same functional attributes as QTI, except that CC introduces even more overhead. SCORM is similar to CC, except that SCORM does not define the format of the content of the assessment items.

Format	**Coupled**	**Content**	**Interaction**	**Integration**	**Overhead**
SCORM	✓	1	∞	-	++++
CC	✓	1	++	-	++++
QTI	✓	1	++	-	+++
GIFT	✗	1	+	+	+
AIKEN	✗	1	+	+	+
Blackboard	✗	1	+	+	+
SERIF	✗	∞	∞	+++	++

Table 2: Functional comparison between current assessment item formats and SERIF. Introducing only slightly more overhead compared to specialized assessment item formats, SERIF is extensible in terms of content types and interaction types, and its defined semantics allows for multiple serializations thus improves ease of integration.

The proposed format has been successfully applied for the project "EduTab" by two different Flemish IT companies for three proof-of-concept assessment applications[10]. The content for these applications originated from four different publishers, each with different content formats. These different content formats where converted to SERIF to provide a uniform format for the three assessment applications. Via these proof-of-concepts, we were able to validate the different proposed features, namely, we can state that SERIF...

is decoupled from the execution environment Content and visualization was successfully decoupled thanks to SERIF, as two of the assessment applications were native iOS apps, whilst the latter one was an HTML5 application.

supports multiple media types The input of the four publishers had varying structure (from XML to CSV), whilst the input content ranged from LaTeX

[10] `https://www.iminds.be/en/projects/2014/03/20/edutab`

to HTML to plain text. For all types of input, a lossless conversion was required and achieved[11].

supports multiple interaction types The assessment items for the three applications ranged from (i) a single question using an advanced interaction type, namely, drawing on a tablet to input the solution, (ii) a series of simple multiple choice questions with hints, and (iii) a series of questions with varying interaction types, from simple to advanced. For all applications, SERIF has been used successfully. It was deemed flexible enough to cope with very varying content types, yet also simple enough to quickly implement the format in the applications.

has specified semantics Having a data model specification and ontology allows SERIF to be distributed using multiple RDF serializations that keep the machine-interpretable meaning (as shown in Figure 3). This improves ease of integration, aggregation with other sources, and discoverability, which eventually improves reuse.

The functional comparison hints towards SERIF's flexibility whilst remaining a simple syntax, and the use case proves how SERIF has all proposed features for an assessment item format.

6 Conclusions and Future Work

Users study more and more online, and learning objects are exchanged to provide new knowledge to these users. However, it is currently not possible to publish and discover individual assessment items. On the one hand, current learning object formats are tightly coupled with the execution environment, and introduce a large overhead to exchange individual assessment items. On the other hand, specialized formats are too restricted and cannot be extended to other interaction types, nor are they easily discovered and reused.

In this paper, we proposed some key features for an assessment item data model, i.e., a data model that (i) is decoupled from the execution environment, (ii) supports multiple media types, (iii) supports multiple interaction types, and (iv) has its semantics specified. Moreover, we introduced SERIF, a Semantic ExeRcise Interchange Format, that is built to comply with the proposed features of the data model. Being based on a data model with specified ontology, SERIF does not restrict its serialization format and is easily integrated with other data sources. Being datatype-agnostic, SERIF does not restrict the content of the exercises to be of any type, but allows for compatibility with future content types. As the content type body can be of any MIME type, and as the interaction types can be extended easily, SERIF allows for very generic and flexible usage. This has been verified by using it as the common format between three applications (built by two IT companies in Flanders), and four content providers.

[11] See `http://edutab.test.iminds.be/pub/data/serifconversions/` for a sample of input-output pairs

In future work, a long-term evaluation is needed based on user studies, to assess that the initial effort to comply to SERIF does not outweigh the advantages of having reusable and decoupled assessment items. We will also further research advanced use cases using SERIF, most importantly the automatic discovery and mash-up of relevant assessment item into an ad-hoc quiz generation application.

Acknowledgements The research activities described in this paper were funded by Ghent University, iMinds, the IWT Flanders, the FWO-Flanders, and the European Union, in the context of the project "EduTab".

References

1. IEEE standard for Learning Object Metadata. Tech. Rep. 1484.12.1-2002, IEEE (2009), `https://standards.ieee.org/findstds/standard/1484.12.1-2002.html`, accessed July 13th, 2015
2. IMS Global Common Cartridge profile. Final Specification 1.3, IMS Global (Jul 2015), `http://www.imsglobal.org/cc/ccv1p3/imscc_Overview-v1p3.html`, accessed July 13th, 2015
3. Bohl, O., Scheuhase, J., Sengler, R., Winand, U.: The Sharable Content Object Reference Model (SCORM) - a critical review. In: Werner, B. (ed.) Proceedings of the International Conference on Computers in Education. vol. 2, pp. 950–951. IEEE, IEEE Computer Society, Auckland, New Zealand (December 2002), `http://dx.doi.org/10.1109/CIE.2002.1186122`
4. Cyganiak, R., Wood, D., Lanthaler, M.: RDF 1.1: Concepts and abstract syntax. Tech. rep., World Wide Web Consortium (W3C) (Feb 2014), `http://www.w3.org/TR/rdf11-concepts/`, accessed July 14th, 2015
5. Mallon, D., Wang-Audia, W., Tauber, T.: The definitive buyer's guide to the global market for learning management solutions 2014. Industry study, Bersin by Deloitte (Aug 2014), `http://www.bersin.com/Practice/Detail.aspx?id=17858`, accessed July 15th, 2015
6. Ninoriya, S., Chawan, P., Meshram, B., VJTI, M.: CMS, LMS and LCMS for eLearning. IJCSI International Journal of Computer Science Issues 8(2), 644–647 (2011)
7. O'Loughlin, E.: Learning management systems userview | 2015. Tech. rep., Software Advice (Feb 2015), `http://www.softwareadvice.com/hr/userview/lms-report-2015/`, accessed July 15th, 2015
8. Reece, A.A.: A Reusable Learning Object Design Model for Elementary Mathematics. Dissertation.com, Boca Raton, Florida, USA (2009), `http://www.dissertation.com/book.php?method=ISBN&book=1599427214`
9. Snell, J.M.: Activity streams 2.0. W3c working draft, World Wide Web Consortium (W3C) (Jan 2015), `http://www.w3.org/TR/activitystreams-core/`, accessed July 13th, 2015
10. Sporny, M., Kellogg, G., Lanthaler, M.: Json-ld 1.0. Tech. rep., World Wide Web Consortium (W3C) (Jan 2014), `http://www.w3.org/TR/json-ld/`, accessed July 14th, 2015