CaVa^{DSL}: virtual Learning Spaces Formal Specification

Ricardo Giuliani Martini[1] and Pedro Rangel Henriques[1]

1 **Centro Algoritmi/Dep. de Informática, Universidade do Minho, Campus de Gualtar, Braga, Portugal**
rgm@algoritmi.uminho.pt, prh@di.uminho.pt

Abstract

In the context of Cultural Heritage, *memory institutions* build exhibition rooms to expose their assets and disseminate knowledge through these learning spaces. CaVa project aims at facilitating the process of learning spaces construction on the web to implement virtual museums. This paper presents CaVa^{DSL}, an external Domain-Specific Language (DSL) designed to specify virtual Learning Spaces enabling their automatic generation. To introduce CaVa^{DSL} language, the paper runs a case study, *Museu Virtual Interativo da Fotografia* (MVIF), presenting the specification for the museum's exhibiting rooms and their final layout. The process that analyzes and transforms the formal specification into the virtual Learning Spaces is briefly described to present the core engine of CaVa platform.

1998 ACM Subject Classification F.4.2 Grammars and Other Rewriting Systems, F.4.3 Formal Languages

Keywords and phrases Domain-Specific Languages, Formal Specification, Context Free Grammars, Virtual Museums, Virtual Learning Spaces

Digital Object Identifier 10.4230/OASIcs.xxx.yyy.p

1 Introduction

To preserve objects that belong to the assets of *memory institutions*, like museums or archives, and to disseminate the enclosed knowledge (the so-called *cultural heritage*), people is resorting more and more to the digital format and computer-based systems. In particular, Internet is an appealing vehicle for imparting knowledge; the related technology has changed the manner of reading, thinking, writing, and learning [7]. According to [5], the rapid and continuing advances in information and communication technologies are changing the ways people share, use, develop and process information. In this context a new concept emerged years ago and is gaining popularity: *virtual museum* [2]; CaVa project, that is discussed along this paper, was born to automatically create those museums. Associated with it, we decided to call *virtual Learning Space* (vLS) the website containing the information related with the museum belongings (its digital objects) that will be seen and explored by the visitor so that he can learn through it. These new web Learning Spaces are the *exhibition rooms* of the traditional (physical) museums [4].

This paper presents an approach to automatically build these virtual Learning Spaces formally specified in a specially tailored language (CaVa^{DSL}) that resorts to a domain ontology. That Domain Specific Language was designed aiming at being easily used by museum Curators and other non-programmers, experts in Cultural Heritage areas—so, as we will illustrate with a running-example along the paper, CaVa^{DSL} has a light syntax and is

7th Symposium on Languages, Applications and Technologies (SLATE 2018).
Editors: Billy Editor and Bill Editors; pp. 1–9

OpenAccess Series in Informatics
Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

focussed on the process of planning and mounting museum exhibits. The ontology schema[1], at the core of CaVa system and underlying approach, is defined with two purposes: first, to give formal semantics to the digital object repository, using an high-level and abstract schema independent of the final use; second, to describe the information that must be displayed in the envisaged virtual Learning Space.

Our main goal in this paper is to discussed as a properly designed language—based on an appropriate vocabulary (defined for a specific knowledge domain)—can effectively facilite the mission of people working in *memory institutions*. From specifications written according to our ontological approach in CaVa^{DSL} language, the system we developed, CaVa [1, 3, 4], is able to automate the creation of web-based virtual exhibition rooms (vLS) to display cultural, material or immaterial, objects.

This paper is organized as follows. The next Section 2 introduces our case study: *Museu Virtual Interativo da Fotografia* (MVIF for short). Section 3, the main one, introduces CaVa^{DSL} language, and Subsection 3.1 continues its presentation outlining the specification of MVIF virtual LS with CaVa^{DSL}. In order to process the formal specification written in CaVa^{DSL} and generate the MVIF exhibition rooms, Section 4 describes CaVa^{gen}, a set of web-application generators. Section 5 presents the result of rendering the files generated by CaVa^{gen}, the desired MVIF. Finally, Section 6 concludes and gives the directions for future work.

2 MVIF at a glance

MVIF is a virtual museum developed by Ricardo Ravanello. The museum exposes the techniques, equipments, ideas and characters of the history of photography [8]. In the work underlying this paper, the idea is not to reproduce the museum faithfully [2], but just to use MVIF specification and implementation to illustrate our proposal presenting a concrete example of a CaVa^{DSL} museum description. To store the MVIF assets, a relational database was built. Some tables of that database can be seen at the MVIF website[3] clicking on the left side menu "MVIF DB TABLES".

To reproduce the MVIF museum in CaVa, it is necessary to describe the context domain in which it is enclosed. For that and according to the CaVa ontological basis, we designed an ontology describing the MVIF concepts and relations; a part of it will be presented. Figure 1 shows the two main concepts related to the MVIF domain, `Technique` and `Period`, and four subclasses of the later. Notice that those two concepts abstract precisely the data enclosed in the two tables shown in the museum's link referred above). Four attributes (datatype properties) for the concept `Period` are also displayed.

[1] Ontology defined at the abstract level of concepts and relations, without instances.
[2] Actually the museum's web site available is the original one, by Ravanello, and not a web site generated by CaVa environment.
[3] Available at: `http://www4.di.uminho.pt/~gepl/cava/MVIF/`

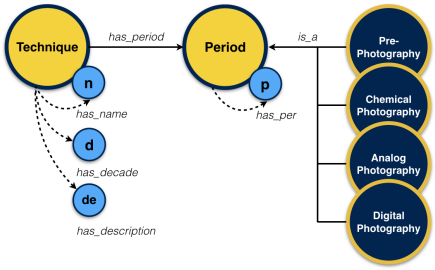

Figure 1 A portion of the "Museu Virtual Interativo da Fotografia" ontology

To relate the abstract concepts of the MVIF ontology with their instances stored in the relational database structure, it is necessary a *mapping* between the two—that mapping will create a pavement to enable querying the database using the abstract vocabulary established by the ontology. For this purpose, the Ontop[4] framework based on the Ontology-Based Data Access (OBDA) axioms was used. The mapping for this concrete case is available at the MVIF website on the left side menu "MVIF MAPPING".

3 CaVa^{DSL} Language

Aiming at an easy generation of virtual Learning Spaces for the use of the person in charge of archives or museums, an external DSL was designed. CaVa^{DSL} was developed to describe, in an abstract level, virtual exhibition rooms in the museum Curator's perspective, giving the Curator the possibility to specify the virtual LS based on a domain ontology vocabulary. CaVa^{DSL} focuses on the exhibition rooms, so the main component of the language is a list of exhibitions. The syntax of CaVa^{DSL} is similar to that of the JSON (JavaScript Object Notation) language, because it is based on the 'key-value pair' format and it is easy for humans to read and write. The structure of CaVa^{DSL} is split into four main blocks that specify: the main configuration, the header, the content, and the footer of the virtual Learning Space, as specified by the derivation rule `p0`.

```
[p0] cava:  mainConfig header content footer ;
```

The details of that language will be presented in the next subsection using MVIF as a running example.

3.1 MVIF specified in CaVa^{DSL}

This section presents the specification of the MVIF museum organized according to the four main blocks of CaVa^{DSL}.

The *mainconfig* defines the virtual Learning Space title and main description, as well as, other components (e.g., carousel of images) related to the entire LS (not only about a specific page, like an exhibition, for example); this configuration is written according to the production rule `p1` of Listing 1:

Listing 1 Production rule p1: mainConfig

```
1  [p1] mainConfig:  'mainconfig' '[' learningSpaceTitle learningSpaceAbout? learningSpaceCarousel? ']' ;
```

[4] Accessible at: `http://ontop.inf.unibz.it/`

Listing 2 is a part of the CaVa^{DSL} MVIF specification written according to the derivation rule p1.

Listing 2 Example applying the production rule p1: mainConfig

```
mainconfig [
    LS title: "Museu Virtual Interativo da Fotografia",
    about [
        p: "O museu virtual da fotografia se propõe a organizar as informações históricas dentro de temáticas
        curatoriais apresentadas em cronologia.",
        p: "Partimos de uma convicção: não  é possível compreender a importância, as possibilidades, o presente e o
        futuro da fotografia sem compreende as ideias e os conceitos, os equipamentos e as tecnologias, as práticas e
        os usos, os personagens e suas trajet órias pelos quais e por onde a fotografia passou para chegar onde
        chegou nos dias de hoje.",
    ]
    carousel [
        interval: 5,
        images [
            caption: "Author - Ricardo Ravanello", src: "imagem-capa-mvif.png", active,
            caption: "Fornalha de um Hammam - Marrocos, 2015 (Ricardo Ravanello)", src: "imagem2-mvif.jpg",
        ]
    ]
]
```

The non-terminal grammar symbol *menu* defines the main menu of the virtual Learning Space that is composed of: brand, background and foreground colors; behaviour (fixed menu or moving following the page scroll); type of the menu links (simple or dropdown), containing the label and the link. Production rule p2 in Listing 3 formalizes this composition.

Listing 3 Production rule p2: menu

```
[p2] header:  'menu' '[' (optionHeader)+ ']' ;
     optionHeader:  brand | backgroundColor | fontColor | behaviourStat | items ;
```

The part of the CaVa^{DSL} MVIF specification in Listing 4 is written according to the derivation rule p2.

Listing 4 Example applying the production rule p2: menu

```
menu [
    brand: "Museu Virtual Interativo da Fotografia",
    background color: green,
    foreground color: white,
    behavior: fixed,
    options [
        label: "Exibições", dropdown [
            dropdown label: "Permanentes", url: "permanentes",
            dropdown label: "Temporárias", url: "temporarias",
            dropdown label: "Especiais", url: "especiais",
            dropdown label: "Futuras", url: "futuras",
        ]
        # "Temáticas" dropdown menu
        label: "Sobre", url: "sobre_mvif", extension: php,
    ]
]
```

Production rule p3 in Listing 5 states that the non-terminal symbol *exhibitions* (a part of the content block and the main component of CaVa^{DSL}) is a list of exhibition rooms. Each exhibition is composed of: title, short description and icon; additional info with a title and a description; behaviour (collapsed or expanded); exhibition type (permanent, temporary[5], future, or special); query operator—`all` (search for all occurrences of a determined ontology concept declared and returns the set of instances); or `one` (search for only one object (first instance) that corresponds to the conditional parameter and the ontology concept)—or SPARQL query (specified according to the production rule p5).

Listing 5 Production rule p3: exhibitions

```
[p3] exhibitions:  'exhibitions' '[' (exhibition)+ ']' ;
                   exhibition:  'exhibition' '[' (optionExhibition)+ ']' ;
                   optionExhibition:  exhibitionTitle | exhibitionShortDescription | exhibitionIcon
                   | exhibitionBehaviour | exhibitionAdditionalInfo | exhibitionType | exhibitionNotification | (queryOperators | sparql) ;
```

[5] If the type is set up to 'temporary', it is possible to configure a notification to the visitor based on the exhibition expiration date;

The `queryOperators` non-terminal symbol defines the operator that shall be used to query the database repository using the ontology vocabulary. The production rule `p4` in Listing 6 formalizes the alternatives and the components of such operators, crucial to extract the information to exhibit in each museum's room.

Listing 6 Production rule p4: queryOperators

```
[p4] queryOperators:  all | one ;
     all:  CONCEPT '->' 'all' '(' parametersAll ')' labelsOptions ;
     parametersAll:  listName ',' mappingOrTriplesFileName ',' ontologyFileName ;
     listName:  TEXT ;
     mappingOrTriplesFileName:  TEXT ;
     ontologyFileName:  TEXT ;
     labelsOptions:  '[' (labelsExhibitionRoom)+ ']' ;
     labelsExhibitionRoom:  elem (',' elem)* ;
     elem:  TEXT ;
```

The rule `p5` in Listing 7 defines the non-terminal symbol `sparql` that offers another operator to write a query resorting directly to the SPARQL query language.

Listing 7 Production rule p5: sparql

```
[p5] sparql:  'SPARQL' '[' sparqlStatement ']' '[' labelsOptions ']' ;
```

The non-terminal `sparqlStatement` corresponds to a declaration based on the SPARQL grammar.

To demonstrate how the production rules `p3` and `p4` are applied, an MVIF specification example is presented in Listing 8. Notice that the terminal symbol `CONCEPT` used in production rule `p4` (Listing 6) must denote a concept belonging to the ontology.

Listing 8 Example applying the production rule p3 and p4: exhibitions and queryOperators

```
exhibitions [
        exhibition [
            title: "Técnicas da Fotografia",
            short description: "Dividimos a história da fotografia em três grandes períodos. Essa classificação se
            justifica não apenas pela mudança dos suportes ou das técnicas, mas também pelo fato de que é possível
            diferenciar drasticamente todo o sistema em torno da fotografia em cada um dos três momentos.",
            icon: "camera-retro",
            additional info [
                title: "1672-2010",
                description: "Período",
            ]
            behavior: expanded,
            type: permanent,
            Technique->all("Técnicas", "mvif.obda", "http://semanticweb.org/
            rgm/2017/mvif/")[headerOfEachElement:"Técnica", "Década", "Descrição", "Período"],
        ]
]
```

At last, the non-terminal symbol *footer* is used to specify an area at the bottom of the LS, containing: images and date; company or developer name; behaviour (fixed footer or moving according to the page scroll); style (simple footer with the data above mentioned or extended with an array of links with title, subtitle, URL, icon, etc.[6]).

This is formalized by production rule `p6` in Listing 9.

Listing 9 Production rule p6: footer

```
[p6] footer:  'footer' '[' (optionFooter)+ ']' ;
     optionFooter:  footerImage | footerFormatDate | footerDeveloper | footerBehavior | footerStyle ;
```

The fourth part of the CaVa^{DSL} MVIF specification, written according to the derivation rule `p6`, is presented in Listing 10 to illustrate its use.

[6] The extended footer is good for addresses, social network links and other important information related to the virtual Learning Space.

Listing 10 Example applying the production rule p6: footer

```
footer [
    images [
        image: "cava_logo.png",
        alignment: right,
    ]
    format date: "Y",
    developer [
        name: "Ricardo Martini",
        alignment: left,
    ]
    behavior: fixed,
    style: condensed,
]
```

Notice that each block and component specification starts with the left bracket "`[`" and closes with the right bracket "`]`", always embracing pairs consisting of plain text or built-in terms.

4 CaVa^{gen}

CaVa^{gen} is a component of CaVa system as fully documented in [3, 4]. As can be seen in the block diagrams shown in the previously referred website at `http://www4.di.uminho.pt/~gepl/cava/MVIF/`, this CaVa component consist of four processors, namely CaVa^{structure}, CaVa^{queries} CaVa^{queriesTriple} (the laternative to deal with triples datastorage, instead of mappings), and CaVa^{run}[7].

The first one is responsible for the generation of the static content of the virtual LS. Moreover, CaVa^{structure} has the task of executing the CaVa^{queries} (or CaVa^{queriesTriple}) processor when, at least, one query operator is set up in the CaVa^{DSL} Specification. The second and third modules have the duty of assembling the ontology queries based on the query operator(s) specified in the CaVa^{DSL} description. They shall handle the mapping or other intermediate file that links the ontology to the database (e.g., an OBDA mapping file, a Terse RDF Triple (turtle) file, etc). The last one, CaVa^{run}, is responsible for executing the queries mounted by CaVa^{queries} (or CaVa^{queriesTriple}) processor and generates the queries results file to be used by the LS Scripts.

The purpose of CaVa^{gen} processors is to get the LS Specification (CaVa^{DSL}) as input and transform it into several scripts possibly written in more than one web language (e.g., HTML, PHP, JS, template engines, CSS, etc.) and other kind of documents (e.g., state files[8]), that together make up multiple web pages, i.e., the complete virtual Learning Space.

CaVa^{gen} processors have two objectives: (1) parsing the CaVa^{DSL} specification of a virtual LS that was manually written by the enduser[9] according to CaVa^{grammar}; (2) generating/setting up the LS scripts component of CaVa^{render}, that comprises the configuration and script files necessary to be rendered by the web browser[10].

Figure 2 shows a simple CaVa^{gen} workflow involving the three steps that shall be followed to achieve the main goal of this work: Specification of a virtual Learning Space in CaVa^{DSL}; Automatic generation and assembly of queries; Automatic generation of static and dynamic content of LS Scripts.

---

[7] Due to space constraints, we can't include here the block diagrams that depict the system architecture; this is way we direct the Reader to the project website and papers.

[8] State files contain data to be used by the processors of CaVa system in order to retrieve important information to proceed with the execution and generation of the virtual LS.

[9] the Museum Curator or another expert in cultura heritage information.

[10] To produce the desired the virtual Learning Space.

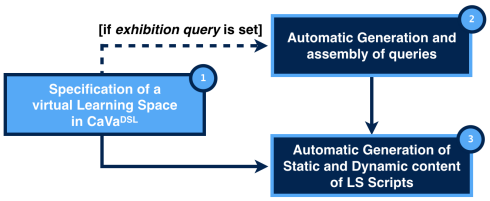

Figure 2 CaVa workflow: from the CaVa$^{\text{DSL}}$ Specification to the virtual LS automatic generation in three steps

From step 1, if there is at least one query operator specified at any exhibition room of the CaVa$^{\text{DSL}}$ specification, step 2 is executed, followed by step 3; otherwise, after step 1, step 3 is executed (excluding step 2), generating only the static content of the LS. After executing these steps, the web browser (comprised in CaVa$^{\text{render}}$) will receive the necessary script files aiming at interpreting and rendering the final virtual Learning Space specified.

5 Generating and Rendering MVIF virtual LS with CaVa$^{\text{gen}}$

Based on the process explained in Section 4, CaVa$^{\text{gen}}$ analyzes the MVIF specification, written in CaVa$^{\text{DSL}}$ (as discussed in Section 3), and transforms it into several scripts that together constitute the envisage exhibition rooms of that virtual museum. Figures 3 to 6 illustrate the effect of rendering, via an web browser, the generated MVIF program and data files. For the sake of space, only some generated web pages will be shown (namely mainconfig, menu, and exhibitions); more details are available from the website `http://www4.di.uminho.pt/~gepl/cava/MVIF/`.

Figure 3 shows the MVIF *entrance all* (actually the homepage) after rendering the files generated from the `main configuration` part of the CaVa$^{\text{DSL}}$ specification. Notice in Figure 3 the main components (*LS title*, *about* and *carousel*) for the museum's web pages created according to that specification.

Figure 3 Main configuration rendered

Based on the `menu` part of the MVIF specification (CaVa$^{\text{DSL}}$ code shown in Listing 3), the museum's menus are rendered as illustrated in Figure 4. According to that specification, the bar on the top of the museum's window, exhibits the LS name (`Museu Virtual Interativo da Fotografia`), and three menu options: two buttons corresponding to dropdown lists (named `Exibições` and `Temáticas`), and a simple button named `Sobre`.

Figure 4 MVIF Menu rendered

The exhibitions running in the museum are group in four type. As only one, of *permanent* type, was specified in our the running example, it will be the only available in the menu button `Exibições` under the option `Permanentes`. Figure 5 shows the exhibition room rendered using the files generated from the `exhibitions` part of the MVIF CaVa^{DSL} specification.

Figure 5 Exhibitions list rendered

In Figure 5 it is possible to see the exhibition attributes specified as *visible*: `title`, `short description`, `icon` (camera-retro), `additional info` (1672-2010 Período) and `behavior` (expanded, denoted by the chevron-up icon in the top right corner).

The final exhibition room rendered, containing the data retrieved from the knowledge repository by the query performed, can de seen in Figure 6. All the details of the webpage shown were rigorously defined in the MVIF specification listed previously (see Section 3.

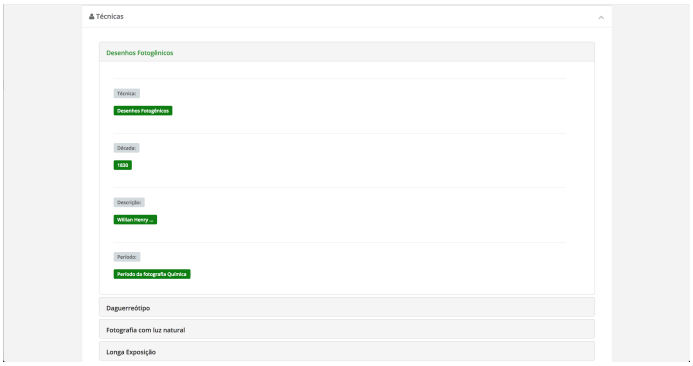

Figure 6 Permanent exhibition room rendered

6 Conclusion

In this paper CaVa^{DSL} language was presented. This language was specifically designed to allow the curators of *memory institutions* to describe rigorously virtual Learning Spaces aiming at their automatic generation. CaVa^{gen} [3], the engine developed to process CaVa^{DSL} specifications, was also introduced briefly in the paper.

The language itself was specified by a typical context-free grammar written in ANTLR notation [6]. Its complex processor was generated automatically by the compilers-generator system ANTLR from a translation grammar defined by a set of Java Listeners written over

the CFG referred above. So the CaVa$^{\text{DSL}}$ development process is not worthwhile for further discussion in the present context. The objective of the paper was, on one hand, to introduce the language design tuned for an easy usage in cultural environments, and on the other hand, to emphasize the power of automatic generation of programs, describing a complex application domain.

Although not discussed in the paper, CaVa$^{\text{DSL}}$ was applied to three different Cultural Heritage scenarios, specifying and creating virtual Learning Spaces complying with their needs as defined by the respective ontologies [2].

To proceed, we intend to extend CaVa$^{\text{DSL}}$ to incorporate more powerful elements, like more sophisticated query operators, variables or functions, that will increase the language expressiveness and improve its usability. We also consider important to investigate the possibility to use YAML or other host language, instead of building it from the scratch; this approach could enable the reuse of some YAML features for free. Furthermore, we plan to work soon on the improvement of: the generated User Interface; the portability of the system; and the system performance. The most important and sensible future work direction is the design and conduction of experiments to assess the system's usability and collect end-user feedback. This is a crucial step to properly move CaVa project onwards.

Acknowledgements This work has been supported by COMPETE: POCI-01-0145-FEDER-007043 and FCT – Fundação para a Ciência e Tecnologia within the Project Scope: UID/CEC/00319/2013.

References

1 Ricardo G. Martini, Cristiana Araujo, Pedro Rangel Henriques, and Maria Joao Varanda Pereira. *Trends and Advances in Information Systems and Technologies*, chapter CaVa: An Example of the Automatic Generation of Virtual Learning Spaces. Springer International Publishing, Switzerland, 2018.

2 Ricardo Giuliani Martini. *Formal Description and Automatic Generation of Learning Spaces based on Ontologies*. PhD thesis, University of Minho, Mar 2018. (submitted–to be discussed in 2018.).

3 Ricardo Giuliani Martini and Pedro Rangel Henriques. Automatic Generation of Virtual Learning Spaces Driven by CaVa^{DSL}: An Experience Report. *SIGPLAN Not.*, 52(12):233–245, October 2017.

4 Ricardo Giuliani Martini, Giovani Rubert Librelotto, and Pedro Rangel Henriques. Formal Description and Automatic Generation of Learning Spaces Based on Ontologies. *Procedia Computer Science*, 96:235 – 244, 2016. Knowledge-Based and Intelligent Information & Engineering Systems: Proceedings of the 20th International Conference KES-2016.

5 MCEECDYA. *Melbourne Declaration on Educational Goals for Young Australians*. Ministerial Council for Education, Early Childhood Development and Youth Affairs, 2008.

6 Terence Parr. *The Definitive ANTLR 4 Reference*. Pragmatic Bookshelf, 2nd edition, 2013.

7 H. Rainie, J.Q. Anderson, and S. Fox. *Challenges and Opportunities: The Future of the Internet*. The Future of the Internet. Cambria Press, 2010.

8 Ricardo Brisólla Ravanello. *Narrativa para bens culturais: tecnologias e aplicabilidades da fotografia digital expandida em museus virtuais*. PhD thesis, Universidade do Minho, Campus Gualtar, Braga, Portugal, 3 2018.