Alessandra da Silva Gomes

“Web Metalaboratory”

“*Meta-Laboratório na Web*”

CAMPINAS
2013

UNICAMP

University of Campinas
Institute of Computing

Universidade Estadual de Campinas
Instituto de Computação

Alessandra da Silva Gomes

"Web Metalaboratory"

Supervisor: *Orientador(a):* **Prof. Dr. André Santanchè**

"Meta-Laboratório na Web"

MSc Dissertation presented to the Post Graduate Program of the Institute of Computing of the University of Campinas to obtain a Mestra degree in Computer Science.

Dissertação de Mestrado apresentada ao Programa de Pós-Graduação em Ciência da Computação do Instituto de Computação da Universidade Estadual de Campinas para obtenção do título de Mestra em Ciência da Computação.

THIS VOLUME CORRESPONDS TO THE FINAL VERSION OF THE DISSERTATION DEFENDED BY ALESSANDRA DA SILVA GOMES, UNDER THE SUPERVISION OF PROF. DR. ANDRÉ SANTANCHÈ.

ESTE EXEMPLAR CORRESPONDE À VERSÃO FINAL DA DISSERTAÇÃO DEFENDIDA POR ALESSANDRA DA SILVA GOMES, SOB ORIENTAÇÃO DE PROF. DR. ANDRÉ SANTANCHÈ.

André Santanchè.

Supervisor's signature / *Assinatura do Orientador(a)*

CAMPINAS
2013

Ficha catalográfica
Universidade Estadual de Campinas
Biblioteca do Instituto de Matemática, Estatística e Computação Científica
Maria Fabiana Bezerra Muller - CRB 8/6162

G585w

Gomes, Alessandra da Silva, 1982-
Web metalaboratory / Alessandra da Silva Gomes. – Campinas, SP : [s.n.], 2013.

Orientador: André Santanchè.
Dissertação (mestrado) – Universidade Estadual de Campinas, Instituto de Computação.

1. Laboratórios na Web. 2. Ambientes de laboratórios. 3. Tecnologia educacional. 4. Ensino à distância. I. Santanchè, André. II. Universidade Estadual de Campinas. Instituto de Computação. III. Título.

Informações para Biblioteca Digital

Título em outro idioma: Meta-laboratório na Web
Palavras-chave em inglês:
Web laboratories
Environmental laboratories
Educational technology
Distance education
Área de concentração: Ciência da Computação
Titulação: Mestra em Ciência da Computação
Banca examinadora:
André Santanchè [Orientador]
Ariadne Maria Brito Rizzoni Carvalho
Cesar Augusto Camillo Teixeira
Data de defesa: 28-06-2013
Programa de Pós-Graduação: Ciência da Computação

TERMO DE APROVAÇÃO

Dissertação Defendida e Aprovada em 28 de Junho de 2013, pela Banca examinadora composta pelos Professores Doutores:

[illegible]

Prof. Dr. Cesar Augusto Camillo Teixeira
DC / UFSCAR

[illegible]

Profª. Drª. Ariadne Maria Brito Rizzoni Carvalho
IC / UNICAMP

André Santanchè

Prof. Dr. André Santanchè
IC / UNICAMP

Institute of Computing /*Instituto de Computação*
University of Campinas /*Universidade Estadual de Campinas*

Web Metalaboratory

Alessandra da Silva Gomes[1]

June 28, 2013

[1]Financial support: CAPES scholarship (process 01-P-01965-2012 2 - 1) 2011–2012

Abstract

The amount of scientific data, services and on-line tools available on the Web offer an unprecedented opportunity to conceive new kinds of laboratories blending resources. Existing experimental and collected data can substantiate asynchronous laboratories. Combined with mashup enabled software, it is possible to produce hybrid laboratories to confront, for example, synthetic simulations with observations. This work addresses this opportunity in the Education context through our metalaboratory, an authoring environment to produce laboratories by combining building blocks encapsulated in components. We introduce here the lab composition patterns and the active Web templates as fundamental mechanisms to support a lab authoring task. These laboratories can be embedded and mashed-up in Web documents. This work shows practical experiments of producing Web virtual and hybrid laboratories.

Resumo

Os dados científicos, serviços e ferramentas on-line disponíveis na Web oferecem oportunidades sem precedentes de conceber nos tipos de laboratório mixando recursos. Dados experimentais e coletados podem substanciar laboratórios assíncronos. Combinados com software apto a mashup, é possível produzir laboratórios híbridos para confrontar, por exemplo, simulações sintéticas com observações. Este trabalho explora esta oportunidade no contexto da Educação através do nosso meta-laboratório, um ambiente de autoria para produzir laboratórios pela combinação de blocos de construção encapsulados em componentes. Introduzimos aqui os padrões de composição de laboratórios e os templates ativos para Web como mecanismos fundamentais para dar suporte à tarefa de autoria de laboratórios. Os laboratórios podem ser embutidos e mixados em documentos Web. Este trabalho mostra experimentos práticos da produção de laboratórios Web virtuais e híbridos.

Acknowledgements

I would like to thank my advisor, Dr. André Santanchè, for the orientation, patience and example of a teacher and researcher. I also want to thank him for encouraging me to realize this work.

To my mother Marilene, sister Dayane, brother Alessandro and nephew Pedro, for all the love, long talks and for being by my side along this journey. All their emotional support was fundamental to me.

To my friends, for always listening and encouraging me. For all the laughter, company, and for helping me to see life in different ways, making me a better person. To my LIS friends, for the example of dedication, knowledge sharing and for all the criticism, suggestions and compliments.

To all the teachers of the Institute of Computing for enriching my knowledge in Computer Science and to the employers for always helping me.

To the committee members, for the suggestions and improvements, making this a better work.

I would like to thank the financial support from Brazilian agencies. In particular, CAPES (01-P-01965-2012 2 - 1), the Microsoft Research FAPESP Virtual Institute (NavScales project), CNPq (MuZOO Project and PRONEX-FAPESP) and INCT in Web Science (CNPq 557.128/2009-9), as well as individual grants from CNPq.

Contents

List of Tables

List of Figures

Chapter 1

Introduction and Motivation

A scientific method can be seen as an analysis / synthesis cycle. The analysis comprises a method to observe phenomena in the real world, may involve experiments and results in a description or model to represent it, which is the beginning of the synthesis. The process becomes cyclic, since the model, which describes the world, will usually be confronted with the observations in the world.

There is a set of specialized tools that support the analysis and synthesis stages. In the analysis side, hardware and software allow to capture data about a phenomenon in the real world – such as sensors controlled by actuators – and software to send this information to be analysed by a computer. In the synthesis side, there is specialized software to build, describe and execute models, such as mathematical modelling or simulation software.

Our research led us to conclude that tools to support analysis tasks are conceived apart from those for synthesis. However, researches usually need to confront results and verify if a model is in accordance with a respective observed phenomenon in the real world. This segmentation of tools makes difficult the integration and comparison of results. This is the main motivation of this work. We propose an approach that offers the opportunity of confronting analysis and synthesis results through a Web laboratory perspective. This possibility was once idealized by Cramer and De Meyer [10], who proposed a hybrid environment where students have an opportunity of comparison between theory and experiments, learning how to apply a scientific method to study a phenomenon.

Our research focus in the Education context. Laboratories are essential to evolve scientific knowledge, but the acquisition of instruments and the maintenance of real laboratories for analysis can be very expensive. Two alternatives are shared: remote or asynchronous Web laboratories. Asynchronous labs is a notion we propose here, of a lab which exploits stored (asynchronous) data shared on the Web of experimental and collected data. The synthesis tools, on the other hand, can assume the form of virtual Web laboratories, also supporting experiments likewise, but built over abstract models. The confront can occur, for example, when remote / asynchronous and virtual experiments are executed side-by-side in the same environment.

Since labs may vary in configuration, across domains and specializations, we propose in this work to extend the notion of authoring, enabling users to create customized labs, by combining primitive building blocks. This is the essence of our metalaboratory.

Our metalaboratory was developed over Componere, a Web environment based on components aimed at authoring compositions over Web browsers [36]. By handling and combining visual software components, users can compose laboratories using the metalaboratory components primitives through direct manipulation. Working over the Web makes it possible to embed different types of resources and services inside components, to be combinable building blocks during the laboratory construction.

1.1 A Brief History

Started by our metalaboratory research idea of enriching multimedia authoring using data captured from the real world. Since sensors and educational kits as Arduino [32] became more accessible, we exploited them interacting with multimedia artifacts.

Part of this research involved studying a special type of system in the industrial context: supervisory systems. Their digital control panels can be developed using an authoring approach based on software components. Supervisory systems or SCADA (Supervisory Control and Data Acquisition) are systems that allow users to collect data and send instructions to different types of hardware devices [40]. These systems inspired our approach to built laboratories with the capacity of reproducing an experiment using data captured from the real world.

We have implemented and validated this initial proposal by building our first laboratory, a remote lab to capture and reproduce the motion of a ball in an inclined plane. It was built using Componere[36]. The motion was based on data captured and stored for asynchronous use, from an experiment executed in the real world. This work introduced the idea of an asynchronous laboratory, a special type of remote laboratory based on stored data coming from external resources in an asynchronous way.

The next step involved building over Componere a second laboratory in a different domain: biology. It was a virtual lab for taxonomic description of specimens. It was fully developed with software components and introduced the idea of description components, a set of components related to descriptive building blocks to be used during the execution of the laboratory.

These two previous experiences led us to propose a model to gather the common components used to built laboratories and an environment where these components will be used to experimentally build laboratories making concrete the Cramer and De Meyer idea [10]. It resulted in our proposition of the metalaboratory, a laboratory to build virtual, asynchronous and hybrid laboratories in an experimental way. We have built a third laboratory to validate the proposal: a hybrid laboratory to reproduce the motion of a virtual and a real damped pendulums. Data captured from the real pendulum was asynchronously reproduced.

Our metalaboratory presents three main contributions: a metalaboratory / laboratory family of components, laboratory composition patterns, and our active Web template approach. After observing that the process of laboratory authoring involves a common group of primitive elements, we defined a set of basic components, the **metalaboratory / laboratory family of components**, which is present in every laboratory composition. During the composition of laboratories, we observed that it is possible to identify different types of **laboratory patterns**, formed by a set of components with their respective connections. The **active Web templates** capture these patterns and enable reusing customizable composition structures.

A relevant contribution developed within this work was the asynchronous laboratory concept. It is based on the principle of storing data captured during the execution of an a real world experiment and use them to virtually reproduce the experiment – e.g., as an animation – several times, whenever it is necessary. Our proposal runs over the Web, we presented a Web asynchronous lab, but the concept can be applied to any type of external data used in an asynchronous way.

In order to conceive the main elements of an asynchronous lab architecture, we conducted experiments involving the complete cycle: real world setups with artifacts and sensors were built; data were captured and stored; asynchronous labs were built and tested based on these data. However, it is important to emphasize that asynchronous labs can be built over pre-existing data retrieved, for example, from the Web.

Our metalaboratory proposal and respective contributions were based on the experience of building three laboratories: the asynchronous mobile ball lab [17], the taxonomic virtual lab [20] and the damped pendulum hybrid lab [19]. This hybrid lab allows to confront experimental data coming from asynchronous labs with theoretical virtual simulations enabled by Web tools. As far as we know, this is the first initiative to produce such a metalaboratory: a laboratory for experimental laboratory authoring, enabling to combine asynchronous and virtual labs in a Web platform.

As our model is based on the experience of three laboratories of two different domains, we believe that an extension to different scenarios will enrich the proposal. The proposed environment offers a set of components to build laboratories. In the present stage, the environment does not offer support to built new components. This task requires programming skills to access the Componere framework and build new components, turning them available on the metalaboratory environment.

1.2 Main Goal and Contributions

The main goal of this work is to conceive a Web based metalaboratory approach for Web based laboratories authoring, and to design a metalaboratory environment.

1.3 Specific Goals

- To design and implement a metalaboratory / laboratory family of components, that systematizes the basic laboratory building blocks.
- To produce laboratory composition patterns, which capture common composition observed when components are composed to form laboratories.
- To conceive an active Web template approach to built laboratories in a metalaboratory environment, representing a generalization of the structure of a laboratory, to be used as an authoring method that guides the user during the construction of laboratories.
- To built a metalaboratory prototype to validate our model. This prototype is a Web environment to build laboratories through an authoring approach by direct manipulation. The prototype itself is based on components and was developed over Componere.
- To validate our proposal by creating practical laboratories by using our metalaboratory.

1.4 Dissertation Structure

The work is formed by a compilation of two published works, a paper submitted for publication and a technical report. We decided that the chapters will not follow the chronological order of publishing, as our third paper [19] presents a better overview of the complete work and will be, therefore, the next chapter.

The papers of Chapters 3 and 4 are practical experiments, which, on one hand, subsidized the Chapter 2, and, on the other hand, can be seen as practical applications.

Chapter 2

Contains the paper **Web Metalaboratory: Composition of Laboratories on the Web**. To be submitted.

This Chapter summarizes our complete proposal. It departs from the ideas presented in Chapter 3 and Chapter 4 to propose a general approach to build laboratories in an experimental way. To validate our proposal, we developed a hybrid laboratory to describe the motion of a damped pendulum.

Chapter 3

Contains the paper **Autoria Virtual Baseada em Dados do Mundo Real**. It was published in the *X Workshop of Tools and Applications (WebMedia - 2011)* [17].

This Chapter introduces the Componere Mundi, an extension of Componere to support compositions based on data captured from hardware devices, like sensors. It represents our first practical experiment of inserting new types of multimedia composition resources based on external data.

Chapter 4

Contains the paper **Web-Based Virtual Lab for Taxonomic Description**. It was published in the *XI Workshop of Tools and Applications (WebMedia - 2012)* [20].

This Chapter presents a proposal of a virtual laboratory based on visual components handled by direct manipulation. In this laboratory is proposed the idea of "description components", a new resource that allows authoring during the execution of the composed artifact. In order to validate our proposal, we have implemented a preliminary prototype with components to describe and to identify living beings. Based on a system called Varan-ID, an online determination system for monitor lizards, the virtual laboratory enables describing and identifying monitor lizards of the genus *Varanus*.

Chapter 5

Contains the technical report **Metalaboratory**. It will be published in the *Institute of Computing – UNICAMP, July 2013* [18].

This Chapter presents technical details of the metalaboratory proposal. It presents the complete metalaboratory model with its components family and detailed descriptions of Web composition patterns and active Web templates. It also presents a prototype of a metalaboratory environment and a practical case to validate it, an hybrid laboratory to describe the motion of an object in an inclined plane.

Chapter 6

Presents the conclusions of this work, our contributions and future work.

Appendix A

Presents the documentation that describes the classes presented in this document.

Chapter 2

Web Metalaboratory: Composition of Laboratories on the Web

2.1 Introduction and Motivation

While laboratories play a fundamental role in Science, we are facing the emergence of a new data-intensive paradigm, in which "In almost every laboratory, 'born digital' data proliferate" [5]. A deluge of this data is available on the Web, becoming possible to produce remote asynchronous[1] laboratories, settled on existing Web data. Not only data is available on the Web, but also tools and services, which can be in many cases "mashed-up". In this paper we explore this opportunity of producing labs, with its foundations on the Web on top of the Web, in the Education context.

We categorize resources – i.e., data, tools and services – in two groups according to their role in the scientific method cycle: analysis and synthesis. The cycle goes from the observation and experimentation in the real world (analysis) to the proposition and validation of models that describe them (synthesis).

In the analysis/synthesis cycle, the analysis can include experiments in laboratory or field observations. The synthesis usually involves models that describe the world, anchored in examples based on everyday observations. It is possible to go from the analysis to the synthesis, or vice-versa. There are several scenarios in which researchers wish to compare collected data (analysis) with results from models (synthesis). This is interesting, for example, to verify if a model is in accordance with a respective description of an observed phenomena in the real world. However, tools are segmented, making it difficult the integration and comparison of results.

Laboratories are essential to evolve the scientific knowledge. On one hand, the acqui-

[1]The term asynchronous here refers to the fact that data from remote labs are not retrieved as soon as they were produced, but afterwards from archives – in our case, shared through the Web.

sition of instruments and the maintenance of real laboratories can be very expensive. On the other hand, there is enough shared data coming from labs around the world, ready to be exploited. We consider here Web based laboratories to address this problem in many respects: remote labs can be shared through the Web; asynchronous remote labs can be produced on top of experimental and collected data stored and shared through the Web; online tools and services can be "mashed-up" to support lab operations.

This work explores this potential of the Web as a basis to build laboratories. We propose a metalaboratory approach that treats Web resources as combinable building blocks to build Web laboratories. In this way, experimental data coming from remote (synchronous and asynchronous) labs can be confronted with theoretical virtual simulations enabled by Web tools. As far as we know, this is the first initiative to produce such a metalaboratory: a laboratory to experimentally authoring another laboratory, enabling to combine remote and virtual labs in a Web platform.

Moreover, we contribute in this work with two techniques envisaged to support lab authoring, but which can expanded to other contexts: laboratory composition patterns and Active Web Templates.

The remaining of the paper is organized as follows: Section 2.2 presents the related work. Section **??** describes our metalaboratory proposal. Section 6 presents our and conclusions and future work.

2.2 Related Work

Related work comprises laboratories and metalaboratories (section 2.2.1), mashups (section **??**) and a framework called Componere (section 2.2.3).

2.2.1 Laboratories and Metalaboratories

Laboratories can be classified into three categories: real, virtual and remote [25][30]. Real laboratories are the classic one, with physical rooms with real equipment and infrastructure. By means of sensors, actuators, a communication infrastructure and a controlling software running in a computer, remote laboratories give access to physical resources of real laboratories. [12] proposes a remote lab to automate a manufacturing cell through a Programming Logical Controllers (PLC's). [4] presents a remote web-based lab that provides access to real instrumentation to debug and test experiments involving Digital Signal Processors (DSP) without physical and temporal restrictions.

Virtual laboratories can simulate the physical equipments behaviour and all the infrastructure of a real laboratory using computational techniques. They can also replicate experiments of a real laboratory using graphical interfaces offering interactive simulations.

[15] presents a virtual lab for Chemical Vapor Deposition to complement a real laboratory in the curriculum of a graduation course improving specific aspects of the theoretic knowledge. [31] presents the design, implementation and usage of a virtual laboratory for medical image analysis based on grids.

In general, laboratories are designed to provide specialized functionalities according to their context. We propose here a decomposition of such functionalities of remote and virtual labs in software primitives encapsulated in combinable components. Laboratories are designed and implemented as compositions of such components, enabling even to combine synthesis and analysis tools. The process to build a laboratory becomes experimental, making the composition tool a laboratory to create laboratories – i.e., a metalaboratory.

The term metalaboratory received different interpretations in related work, making it difficult to achieve an unified definition. The term is used in [23] to describe a tool for the composition of technologies and services aimed at teaching chemistry. It is defined as a cluster of laboratories distributed on the network that shares hardware, software, and knowledge. In [11] the term is used to define a tool for agent based modelling and simulation, i.e., a virtual laboratory perspective.

A group of related work adopt the interpretation of metalaboratory as an hybrid laboratory. [34] proposes a hybrid between remote and virtual laboratory to perform experiments and to compare the results in the engineering field. [2] presents the TriLab, a union of the three main types of laboratories, real, virtual and remote. It aims at teaching how to handle chemical equipment. [3] proposes a hybrid of a virtual and a real laboratory to teach optics, in which the experiments runs in parallel to enable comparative studies. As we will further detail, all these initiatives address specific scenarios, contrasting with our approach, which was designed to be flexible, addressing many contexts.

Cramer and De Meyer [10] introduced the proposal of interacting virtual laboratories with real laboratories to bridge theoretical models and experimental data. They extend the notion of virtual laboratory in order to embrace what we call here the remote laboratories, i.e., the connection with the real world. According to the authors, the development of a hybrid environment could provide to the students an opportunity of comparison between theory and experiments, offering the possibility of learning how to apply the scientific method to the study of a phenomenon. In 1997, they introduced three new terms: theory based virtual laboratory, experimentally based virtual laboratory and hybrid virtual laboratory. The first one, represents the space where the user can explore the theory of a phenomenon by manipulating parameters of an equation and investigating the respective consequences. The second one, starts from experimental measurements, captured and stored in digital form, to be accessed by a user interface. Finally, the third is defined as the combination of both to provide an environment to compare the theory against the experiments.

We will use the term metalaboratory in this work to refer to a laboratory to build laboratories. It is aligned with Cramer and De Meyer [10] perspective, as it can be applied to build hybrid laboratories. In order to achieve flexibility and generality, our approach to implement the metalaboratory is based on primitives represented as combinable software components, which can be specialized to a given domain.

2.2.2 Mashups on the Web

The Web increasingly proportionates an environment full of different types of information and services. While some users produce more content – e.g. videos, slides –, others want to reuse them in their applications. In this context a powerful reuse technique named mashup appeared. Mashup can be defined as an interactive Web technique able to combine resources from several origins. It combines "on the fly" content from different online data sources to build new services or applications [27]. By "on the fly" we mean that data come from different sources and are integrated in the client side in the moment the application is running.

Nevertheless, mashups are not easy to implement. The user needs programming skills to build them. In order to reach the end-user, mashup editors appeared over the last few years, with a general goal of empowering end-users to build custom Web applications [44]. The basic principle is to enable non-programmers to create functionally rich applications, by using pre-defined building blocks for: data sources, application logic and user interface. Some examples are projects like Yahoo Pipes (http://pipes.yahoo.com) and IFTTT (https://ifttt.com/). Because users can put together and customize their own mashups, the concept of personal environments to consume content became popular. They usually take a form of dashboards – e.g., NETVIBES (http://www.netvibes.com/) – in which the user selects components to: receive news, monitor data like stock market, write and view notes etc. Each user can organize the position, size and format of mashups.

More specialized solutions for specific domains - e.g., business and Education - have gained attention. In the business context: the development of the Enterprise Mashup Markup Language (EMML) by the Open Mashup Alliance - OMA (http://www. openmashup.org/); the mashup-based personalized environments [21] and the IBM Mashup Center (http://www.ibm.com/developerworks/lotus/products/mashups). In the Educational context, there are several proposals for Mash Up Personal Learning Environments (MUPPLE) [42]. This scenario involving a high diversity of trends requires a generic backing functionality applicable to the different mashup solutions. Our research contributes in this sense. It is based on the division of tasks between software developers, creating new component and adapting mashups to components, and user-authors customizing and plugging these components, providing an strategy to collaboratively explore skills.

Usually, strategies to compose mashups are flow-based, differentiating three kinds of components: data, application logic and user interface [44]. Our approach, on the other hand, compose lightweight components that unifies data, application logic and interface in a single model and can be applied to different contexts.

2.2.3 Componere

Componere is a Web authoring system based on components fully running on browsers [36]. It is based on the concept of multimedia authoring systems, which allow authors to create multimedia products by using a combination of multimedia primitives [7].

It offers to the authors the possibility of creating their own components library for specific domains or applications. Some libraries are for biodiversity [36], GIS, e-science [35] and education.

Figure 5.1 shows the authoring process in the Componere environment. It is organized in three stages: primitives building, authoring and execution. In the first stage, primitives are built as components and stored in a library. New components are built by software developers or end-users according to the type. Software developers are in charge of those components which require programming. They are based on the Digital Content Component (DCC) model [36].

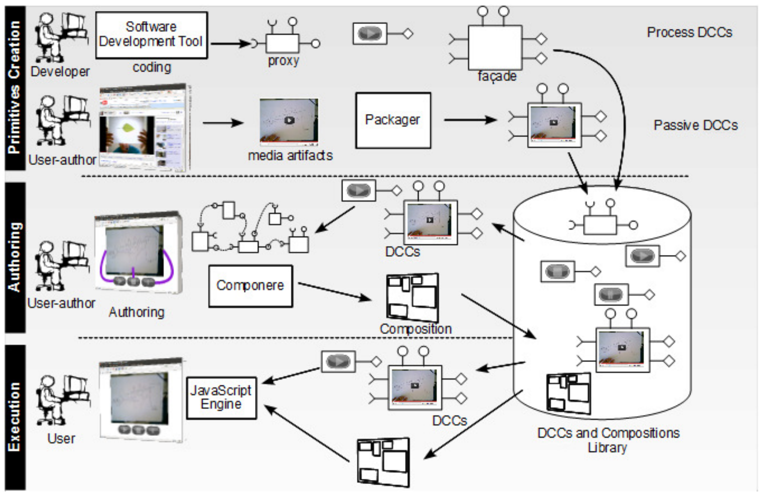


Figure 2.1: Graphical representation of Componere [35]

In the authoring stage, the user combines the available primitives to produce compositions. Components are the raw material for this stage, which involves customization and composition of components. The final product of this stage can also be stored in a library. Finally, in the last stage, the compositions are executed over a browser delivered "mashed-up" with pages.

2.3 Web Metalaboratory

Metalaboratory here refers to an environment to produce tailored laboratories. It is based on the composition of Web primitives embedded in components, which gives an experimental perspective to the environment. Authors can select, customize and combine components by direct manipulation, trying alternative scenarios – a laboratory to experimentally create laboratories: a metalaboratory.

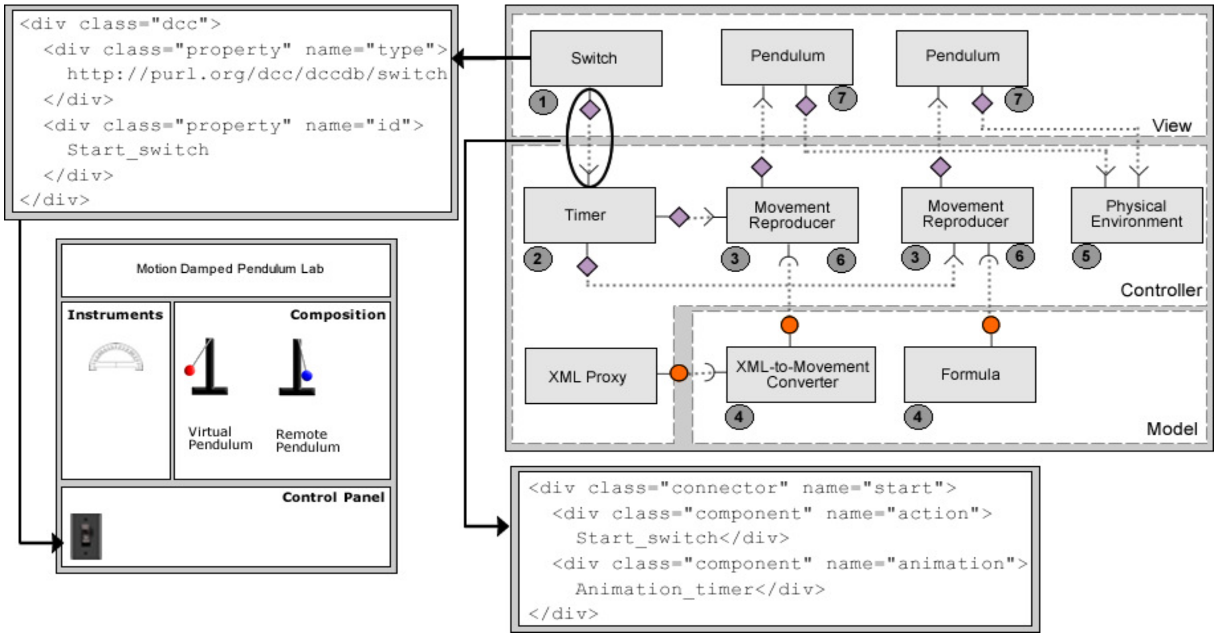
As presented before, [10] proposes an environment to confront theory and experimental knowledge in hybrid laboratories combining theory based virtual laboratories and experiments based remote laboratories. Our metalaboratory addresses this challenge, expanding it to comprise the construction of laboratories as an experimental process, which allows gathering knowledge regarding lab creation and evaluation.

We consider the Web an inherent part of the metalaboratory. While remote laboratories can be expensive and hard to scale up, experimental and collected data available on the Web, coming from third party laboratories, can take a role of asynchronous laboratories. For example, stored images from telescopes can be asynchronous telescopes, stored data captured from sensors can be reproduced. Moreover, there are several tools and services available on the Web, to be encapsulated in components boosting the lab capabilities.

The presentation of our metalaboratory will be based on a practical example, whose result is showed in Figure 2.2. It puts side-by-side two damped pendulums. The left one captures and reproduces the movement of a real world pendulum. The right one simulates a pendulum movement based on an equation. This example shows how our approach is able to blend in the same Web document different kinds of Web resources, subsidizing the confront of observations (analysis) and a theoretical model (synthesis).

On one hand, it illustrates our approach to exploit asynchronous labs. We settled up a hands-on experiment to capture, store and share data of a real damped pendulum on the Web in XML format. Part of the composition in Figure 2.2 retrieves these data and reproduces the pendulum movement. On the other hand, the example includes a simulation driven by a component software in a mashup fashion.

Our approach to conceive a metalaboratory departing from an authoring tool is based in three elements: a metalaboratory / laboratory Web components family, laboratory composition patterns and an Active Web template based technique. These three elements are contributions of this work. Passive Templates for authoring were introduced in our previous authoring tool. Here we transform it in an active approach and we expand it by providing templates mixing Web documents and component compositions. In the three following sections we detail these elements.



```
<div class="dcc">
  <div class="property" name="type">
    http://purl.org/dcc/dccdb/switch
  </div>
  <div class="property" name="id">
    Start_switch
  </div>
</div>
```

```
<div class="connector" name="start">
  <div class="component" name="action">
    Start_switch</div>
  <div class="component" name="animation">
    Animation_timer</div>
</div>
```

Figure 2.2: Composition schema of the damped pendulum lab

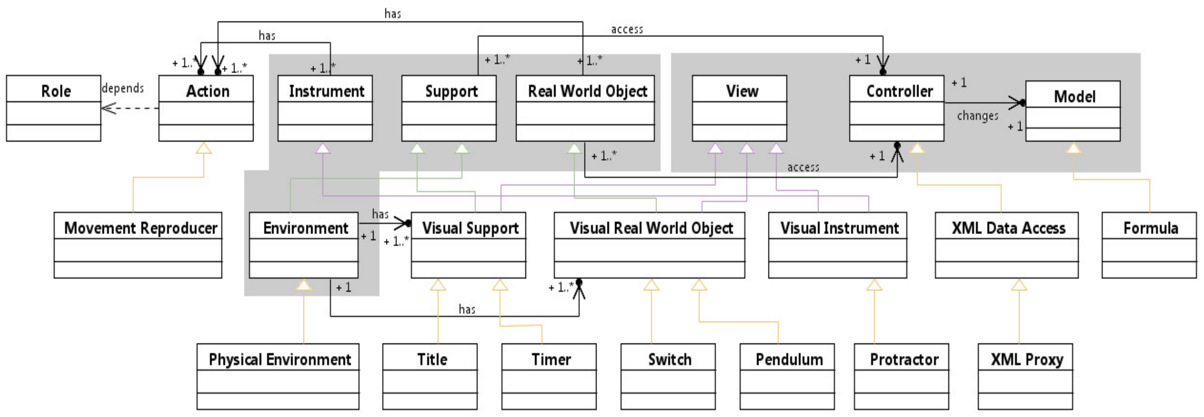


Figure 2.3: Classes of the damped pendulum lab components

2.3.1 Metalaboratory / Laboratory Web Components Family

A metalaboratory is an environment based on Web components to build laboratories through an authoring approach. Authoring in the hypermedia context involves to design and structure information in a particular way [16]. More specifically, multimedia authoring is related to the creation of multimedia artifacts by assembling, synchronizing, and adding interactivity to media from different sources [28]. In this work. authoring embraces multimedia and goes beyond, taking as raw material any resource available on the Web, encapsulated in a component model and able to be "mashed-up".

To offer an environment in which the user is free to compose laboratories through Web components, we start defining a metalaboratory model, by systematizing and reducing elements observed in virtual and remote laboratories to their primitives constituints. Figure 2.3 illustrates the main part of the model on a class diagram format. Since the metalaboratory is derived from the Componere environment, we adopt its Web components model – the Digital Content Component (DCC), see Section 5.2.2.

DCCs are organized in OWL classes according to their functionality. Each DCC class defines the semantics of a component in a taxonomy and the provided/required interfaces. Our family of metalaboratory components is systematized around a set of classes illustrated in Figure 2.3. We adopted an UML visual approach to represent OWL classes. There are two groups of classes (groups inside gray boxes in the figure): those related to the role of a component in a laboratory – **environment, support, instrument and real world object** – and those related to the component profile according to the Model-View-Controller (MVC) pattern. The classes were designed as complementary classifications. A component typically specializes one class of each group. Therefore, we have: visual instrument, real world model etc. A DCC belonging to a class means it complies to a set of expected interfaces, related to the role of the component and the way it interacts with the environment and other components.

Instrument DCCs are virtual equivalents of laboratory instruments. Authors can make them available in a laboratory to perform, for example, measurement tasks. **Support** components perform auxiliary control and management tasks. **Real world objects** are components that simulate or reproduce the behaviour of objects in the real world. **Environment** components virtually represent the space in which the laboratory runs. It performs tasks and provide information of the environment.

Figure 2.2 shows a diagram illustrating a component-based composition of our practical case. The diagram notation is inspired in UML, adapted to our composition context. The flow of execution starts when the user clicks on the **switch**, which represents a **visual real world object** component that is responsible for starting the execution of the experiment. It sends a notification to start the **timer**, a **non visual support** component that is responsible for producing event messages in a prefixed interval, to give rhythm to the pendulum motion. For each timer cycle, it sends a notification to the **movement reproducer** component, which collects data from source components and translate them into pendulum positions. Each **pendulum** component requests to its data source – i.e., **XML-to-movement converter** and **formula** – the value of the next pendulum position. The **XML-to-movement converter** is a model component responsible for retrieving data in XML format, converting them to the movement reproducer format. The **XML proxy** is a controller component responsible for accessing data stored on a remote XML data source. The damped pendulum laboratory is available at http://fluidweb.sourceforge.net, on the link “Metalaboratory”.

2.3.2 Laboratory Composition Patterns

Inside this composition, it is possible to identify patterns to compose laboratories. Figure 2.4 illustrates the behaviour reproducer / simulator lab pattern, which is proposed in this work. It is based in three tier, which can be executed by one or more components. Consider a lab able to show the movement of an object – as our example – the model tier provides data concerning this movement (behaviour) – it can be a data source or producer; the controller tier consumes data from the model and manages the reproduction of this movement in the view tier, which in turn provides a visual animated real world object, apt to display the movement.

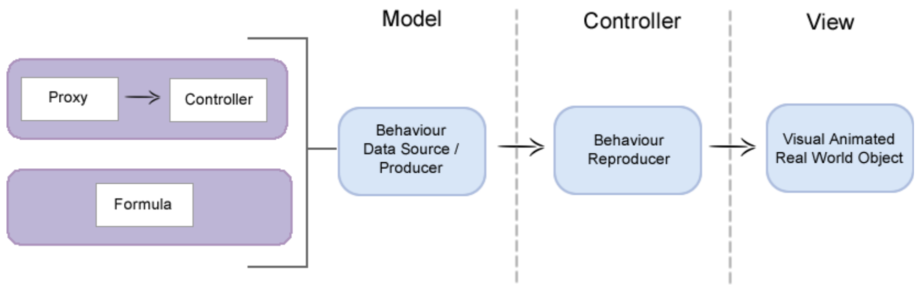

Model
Controller
View
Proxy
Controller
Formula
Behaviour Data Source / Producer
Behaviour Reproducer
Visual Animated Real World Object

Figure 2.4: Behaviour Reproducer / Simulator Lab Pattern

In the model tier we define two patterns specializations. The **Asynchronous Lab Pattern** combines a Proxy – a kind of DCC that bridges remote resources providing them through local interfaces [36] – with a DCC to convert the input format to the expected by the reproducer. The **Virtual Lab Pattern** is a DCC which produces simulated values – in our example a pendulum movement defined by an equation. These patterns are fundamental to our metalaboratory approach, as they generalize common practices. Other examples of lab patterns we produced: synchronization pattern to synchronize two comparable movements(e.g., pendulums); control panel pattern etc. In order to capture and reuse such patterns, we developed a Web template based methodology.

2.3.3 Active Web Templates

Even tough it is possible to start a new laboratory from scratch, our observations show that laboratories tend to follow patterns according to their specificity, and that it is important to reuse not only components but also the composition design. Thus, we developed a technique based on Active Web templates, which are generalizations of Web documents blended with DCC compositions. They capture both, composition patterns and the lab layouts, in a document.

The Componere approach to blend compositions with documents is based on representing these compositions by using HTML tags as part of Web documents [36], adopting Microformats [20]. Figure 2.2 illustrates the instantiation of a **switch** component and a connection between two components, repurposing `div` tags and `class` attributes. Whenever the composition is presented to the user in a browser, a client-side Javascript engine converts Microformats marks in instantiation and connections of Javascript DCCs.

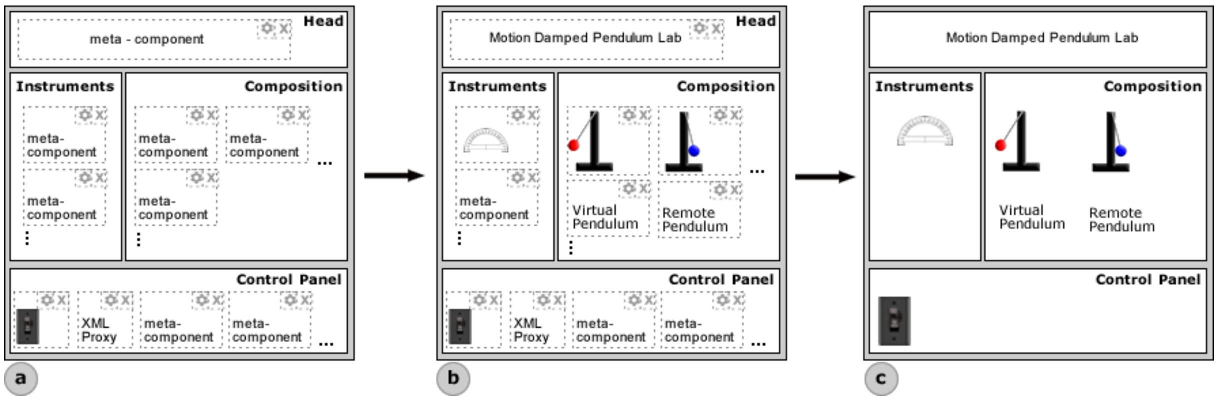


Figure 2.5: Metalaboratory Template

In order to generalize a composition + document structure, our Active Web Template defines configurable spots by replacing DCCs of the composition with meta-DCCs.

A meta-DCC is at the same time: (i) a surrogate of a future DCC and (ii) a software unit that will select and tailor this future DCC. As a surrogate (i), it ocupies in the Web document structure the specific spot where a future DCC will replace it. The software unit (ii) is triggered during the lab authoring, when a template is instantiated. Each meta-DCC becomes an active spot, containing an "editor component" specialized in selecting and tailoring DCCs to that specific part of the composition. Therefore, it is possible to have meta-DCCs specialized in: selecting data-sources, playing and customizing real world objects etc.

Figure 2.5 illustrates our three steps approach to produce labs based on Active Templates. In step (a) a template designer builds a template, usually by taking a pre-existing lab composition and generalizing it by replacing components by meta-DCCs. In step (b) an author starts by instantiating an Active Template. Whenever the template is executed, each meta-DCC is activated inside its position of the document. We call the template active since it is a runnable template auto-configurable. It contrasts with passive templates adopted by other approaches. The author customize the template through these meta-DCCs. According to their speciality, they allow authors to: select proper data sources, drag and position components in a canvas space etc. In the last step, an engine converts the template in a final Web document + composition. The template meta-DCCs give directions to the engine of the DCCs to be inserted and tailored.

2.4 Conclusion

In this paper we presented our Web metalaboratory approach to produce laboratories by combining Web building blocks. It explores the opportunity offered by several shared data sources providing experimental and captured data, to produce asynchronous remote laboratories. The model enables combining services and tools available on the Web, by encapsulating them in a component model. This work builds a metalaboratory system on top of a pre-existing authoring environment, adding four main contributions: (i) a metalaboratory / laboratory Web components family; (ii) a set of laboratory composition patterns; (iii) an Active Web template based technique, comprising the new meta-DCCs; (iv) practical implementations of laboratories built by using this tool. As far as we know, this is the first initiative to produce this kind of metalaboratory – a laboratory to experimental laboratory authoring – enabling to combine remote and virtual labs in a Web platform. As idealized in [3], our environment supports the comparison between theory and experimental knowledge, through a tool that combines properties of analysis and synthesis for Science teaching.

Even though we applied our Active Web Template technique to a lab authoring context, its underlying approach can be generalized to be applied to other composition contexts.

To validate our model, we built laboratories of different domains. In this paper we presented a physics virtual lab to study the motion of damped pendulums. In [20] we present a virtual biology laboratory for taxonomic descriptions and in [17] we present a remote physics lab to the motion of an object on an inclined plane.

Future work includes the expansion of the authoring tool to provide enhanced support for template creation; the definition of new template operators to enable more flexible customization; the expansion of our laboratory component family.

Chapter 3

Autoria Virtual Baseada em Dados do Mundo Real

3.1 Introdução

O conceito de autoria pode ter interpretações variadas a depender do contexto. No domínio da multimídia, autoria remete ao uso de ferramentas que possibilitam compor e sincronizar mídias em uma estrutura narrativa, com o objetivo de produzir apresentações, material de ensino etc. As ferramentas neste contexto atuam geralmente como integradoras de mídias produzidas externamente. Pode-se classificar tais mídias em dois grandes grupos: aquelas capturadas do mundo real por sensores – e.g., filmagens, fotografias, gravações – e aquelas sintetizadas.

No que tange às mídias capturadas – foco deste trabalho – as ferramentas de autoria usualmente dão ênfase àquelas relacionadas aos tipos fundamentais de mídia – vídeo, áudio e imagem – dada a natureza inerente ao trabalho multimídia. Entretanto, a popularização de sensores de muitas outras naturezas, aptos a ser integrados com computadores – e.g., movimento, luminosidade, acelerômetro - traz novas possibilidades para o contexto de autoria multimídia.

Por outro lado, no contexto de controle industrial, que exige a construção de sistemas que constantemente se relacionam com diversas naturezas de sensores, observa-se uma prática aplicada à construção de sistemas supervisórios, cuja abordagem é muito próxima ao princípio da autoria. Tais sistemas utilizam software para supervisionar as variáveis e dispositivos do sistema físico, conectados por meio de drivers específicos [33]. Utilizando ferramentas gráficas de composição por manipulação direta, com abordagem semelhante àquela usada pelos sistemas de autoria, o autor pode construir complexas interfaces visuais de monitoramento.

Inspirado na idéia de sistemas supervisórios, este trabalho apresenta o Componere

Mundi, que explora a combinação de composições multimídia na Web criadas por um sistema de autoria existente, o Componere, com dados capturados do mundo real por meio de sensores de diversas naturezas. O Componere é um sistema Web baseado em componentes e concebido para operar diretamente sobre navegadores [36]. Desta forma, este trabalho tem por objetivo integrar a ferramenta de autoria Componere com princípios de sistemas supervisórios, permitindo o acesso e apresentação de dados de ambiente, capturados por dispositivos de hardware. Tal extensão explora o modelo de componentes do Componere, que permite que os novos recursos se integrem e combinem com outros recursos de composição multimídia.

3.2 Arquitetura

A Figura 3.1 ilustra a arquitetura construída para a ferramenta deste trabalho. A arquitetura é dividida em dois grupos de ações: (i) acesso e entrega de dados de sensores; (ii) autoria. Cada um deles será descrito em subseções subsequentes.

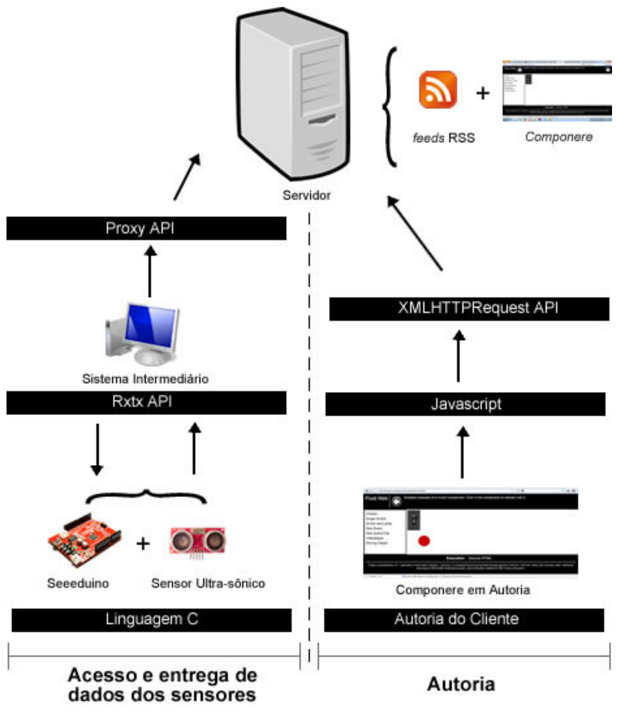


Figure 3.1: Arquitetura da ferramenta desenvolvida

3.2.1 Acesso e entrega de dados de sensores

Este grupo de ações está relacionado com as etapas de captura de dados de sensores, tratamento e distribuição dos através de feeds RSS. Isto envolveu a integração de dispositivos de hardware e módulos de software. Os dispositivos de hardware são responsáveis por capturar os dados do mundo real e entregar ao módulo de software, que é responsável por tratar estes dados e distribuí-los através de feeds RSS.

Para capturar dados do ambiente são utilizados sensores, que são dispositivos de medição capazes de transformar grandezas físicas, referentes a fenômenos ocorrendo em determinada região do espaço-tempo, em grandezas lógicas [38]. Estes sensores, unidos a algum circuito analógico/digital, podem enviar dados a um computador ou a outro dispositivo de hardware programável por meio de uma interface de comunicação (e.g., USB, RS232).

O software desenvolvido recebe os dados que foram enviados pelo dispositivo de hardware e os trata. Visto que os sensores estão sujeitos a ruídos, é feito um tratamento dos dados recebidos para se reduzir ao máximo a taxa de erros.

A arquitetura está sendo projetada para que possam ser usados diferentes filtros. Neste primeiro protótipo foi usado um filtro digital passa-baixas média móvel, que tem a função de reduzir as frequências altas, produzindo uma homogeneização geral dos dados de acordo com os valores definidos para a janela de convolução [9]. Dentre os diversos tipos de filtros digitais passabaixas, o filtro média móvel possui o objetivo de calcular a média de um número de pontos do sinal de entrada, produzindo novos pontos no sinal de saída em sistemas discretizados [1]. Optou-se pela utilização de um passa-baixas média móvel por sua simples implementação e rápido tempo de resposta quando aplicado a um conjunto de dados discretos [1].

Com os dados devidamente filtrados, é feita a geração de feeds. Foram adotados os RSS feeds pela facilidade de acesso e flexibilidade de uso através da Web. O sistema de assinatura e sindicância de feeds está amplamente disponível em ferramentas Web. Desta forma, o usuário poderá não apenas utilizar os dados para o software de autoria proposto, como também através de ferramentas Web.

Optou-se pelo o Really Simple Syndication (RSS) por ser o padrão de feeds mais difundido na Web. O software, por meio de uma API, armazena os dados filtrados em um arquivo RSS localizado no servidor. Os dados ficarão então disponíveis para posteriormente serem acessados.

3.2.2 Autoria

Os dados publicados como RSS feed são acessados e apresentados por componentes especializados, aptos a serem manipulados e compostos pelo sistema de autoria Componere.

Para tanto, fez parte da implementação deste protótipo a criação de componentes em JavaScript que acessam os dados dos sensores, como também um componente para a apresentação dos dados. O componente especializado no acesso aos dados utiliza o XML-HTTPRequest que possui o objetivo de enviar requisições HTTP ou HTTPS para um servidor web e carregar os dados de resposta diretamente no script [41], neste caso dados RSS.

Seguindo a arquitetura do Componere que é independente de plataforma e navegador – conforme detalhado em [36] – os componentes são escritos 100 % em JavaScript. Um dos componentes atua como Proxy, acessando os dados remotos do servidor e os entregando localmente para os demais componentes, que realizam operações locais de apresentação dos dados e se integram com outros componentes do ambiente de autoria.

3.3 Implementação e Experimento

Para se testar a arquitetura proposta, um experimento foi implementado. O experimento selecionado foi o estudo do movimento acelerado de um cilindro que desce por um plano inclinado. O objetivo do experimento foi capturar dados de um cilindro real rolando em um plano inclinado e o reproduzir em uma animação em navegador Web, usando o Componere como suporte para a sua construção.

Do ponto de vista do software, a implementação para este experimento envolveu três etapas: a programação da placa microprocessadora, o desenvolvimento de um software intermediário entre a placa e o arquivo RSS e a integração dos componentes da simulação da queda do cilindro no Componere.

3.3.1 Experimento

O experimento envolveu um computador, uma placa microprocessadora seeeduino, um sensor de posição ultra-sônico, um protoboard, cabos, uma mesa inclinada e uma garrafa pet cheia. A Figura 3.2 ilustra a montagem do experimento criado.

O sensor ultra-sônico captura os diferentes valores referentes a posição do cilindro (garrafa pet) na medida em que ele desliza sobre um plano inclinado. A garrafa foi posicionada próxima ao sensor em uma das extremidades da mesa inclinada e, ao ser liberada, rolou até a outra extremidade.

A captura dos dados foi realizada através de um circuito montado com a placa seeeduino ligada ao sensor ultra-sônico. O software implementado envolveu o desenvolvimento de um módulo em C, que roda dentro da placa. Ele identifica os valores que foram capturados pelo sensor ultra-sônico e os transmite para a interface de saída da placa (USB).

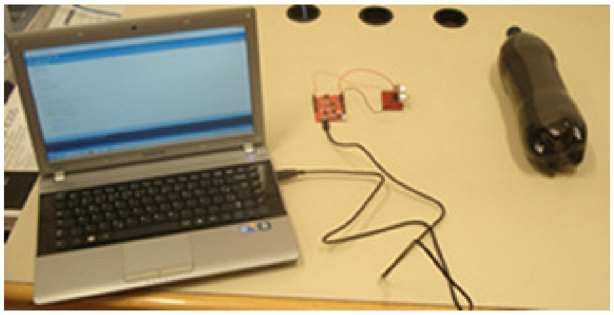


Figure 3.2: Kit utilizado para a execução do experimento

Optou-se pela utilização da placa seeeduino, pois esta é baseada no projeto Arduino, uma plataforma de hardware livre de baixo custo com entradas e saídas analógicas e digitais embutidas [32].

3.3.2 Software Intermediário

O software intermediário tem por objetivo estabelecer a ponte entre a placa seeeduino e a publicação de feeds. Ele foi implementado em Java e faz uso de duas APIs de comunicação: uma para permitir o acesso aos dados do buffer de entrada via USB e uma outra para enviar os dados ao arquivo RSS.

3.3.3 Integração no Componere

A integração com o ambiente de autoria Componere envolveu a construção de componentes especializados, para acesso aos dados de sensores e a reprodução do movimento do cilindro na forma de uma animação. O software está disponível online em: http://fluidweb.sourceforge.net/hundred/html-examples/

Este endereço dá acesso a diversos exemplos de composição feitas no Componere. Para visualizar especificamente o software implementado neste trabalho clique em "Moving Object". O cenário criado foi formado por uma rampa - simbolizando o plano inclinado - e um círculo - representando a base do cilindro que em queda.

A arquitetura do Componere se baseia em componentes como blocos de construção básicos das composições. A ferramenta de autoria subsidia a customização e conexão de componentes, no processo de construção de composição. A Figura 3.3 ilustra a composição de componentes para o experimento realizado. Dois deles, o Proxy RSS e o Cilindro Móvel foram implementados como parte deste projeto.

Conforme ilustra a figura, a Chave de Ativação está ligada ao Proxy RSS. Ela é responsável por dar início ao processo de recuperação de dados dos sensores e sua re-

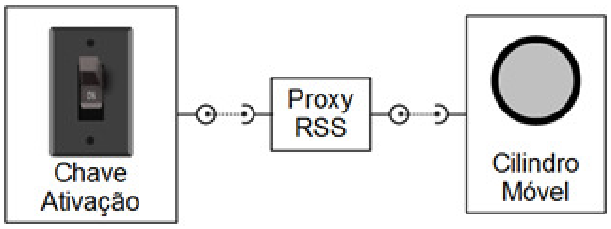


Figure 3.3: Composição do Experimento

produção. O componente Proxy RSS recupera dados dos sensores em formato RSS gerados e os entrega localmente para o componente Cilindro Móvel.

Na medida que o componente Cilindro Móvel recebe os eventos RSS com as leituras do sensor ele posiciona o cilindro na tela, gerando uma animação que reproduz o movimento do cilindro original.

Dado que o Componere é um sistema ainda em processo de desenvolvimento, a ferramenta que permite a edição interativa da composição por manipulação direta ainda não está concluída. Deste modo, a descrição da composição é feita através de tags HTML e pode ser inserida diretamente dentro de páginas, sem a necessidade de se escrever código em JavaScript (vide detalhes em [36]). Para ter acesso a especificação da composição basta clicar no link “Source HTML” no rodapé do sistema online.

3.4 Conclusões

Este trabalho apresentou nosso protótipo da ferramenta Componere Mundi que integra o conceito de sistemas supervisórios com a ferramenta de autoria Web Componere.

Foram implementados e integrados como partes desta ferramenta: (i) um módulo de software em C executado na placa microprocessadora; (ii) o módulo de capturar dados do mundo real e publicação em RSS; (iii) componentes para o Componere, para acesso aos dados em RSS e sua reprodução em uma animação. Foi realizado um experimento prático capturando e reproduzindo o rolamento de um cilindro em um plano inclinado.

Comparado com outras abordagens de captura e apresentação de dados de sensores – tais como a de supervisórios (e.g., LabVIEW - http://www.ni.com/labview/), ou sistemas especializados (e.g., Lego Mindstorms - http://mindstorms.lego.com) – nossa abordagem tem a vantagem de permitir a integração dos dados de sensores com o restante dos recursos de autoria na Web providas pelo Componere. Por exemplo, os dados de movimento capturados podem ser sincronizados com o movimento simulado de um cilindro baseado em uma equação. Isto permite comparar dados sintetizados com dados capturados.

Os trabalhos futuros incluem: a finalização no desenvolvimento do módulo de autoria por manipulação direta; a construção de uma biblioteca de componentes especializados;

a integração de componentes de arquiteturas de supervisório na atual arquitetura.

3.5 Acknowledgments

This work was partially funded by CAPES, CNPq, FAPESP, CAPES-COFECUB (AMIB project) and INCT in Web Science (CNPq 557.128/2009-9).

Chapter 4

Web-Based Virtual Lab for Taxonomic Description

4.1 Introduction

Virtual laboratories simulate physical equipments and the infrastructure of a physical laboratory by using computational techniques. They can represent experiments by graphical interfaces and offer interactive simulations.

Virtual labs can be tooled to afford learning experiences comprising exercices, theorical explanations, and interactive assistants that explain experiments step by step. In many cases, they can be used any time and from anywhere. This kind of laboratories are also known as simulated laboratories or e-laboratories [25].

By handling and combining visual software components, users can describe specimens in a virtual laboratory. This paper presents our work of such a tool involving the description of living beings. In this work we investigate a specific kind of Biology virtual laboratory to support taxonomic description of specimens, in which the basic lab elements are virtualized as visual software components.

This is an ongoing work and, in order to validate our proposal, we have implemented a preliminary prototype with components to describe monitor lizards of the genus *Varanus*. The lab is based on a system called Varan-ID.

Varan-ID is an online determination system for monitor lizards. It is based on a morphological knowledge base of a group on carnivorous lizards, the genus *Varanus*. The system is based on the idea that not only experts are involved with monitor lizards. Students, curious, breeders, keepers are either interested in this subject, but may not have the necessary knowledge to work with the specimen. The process of identification in the Varan-ID system is based on descriptors.

In this paper we present a prototype of our virtual web laboratory to describe and to identify living beings. It is based on visual components handled by direct manipulation, which play roles of building blocks for descriptions and lab tools. Therefore, when a user inserts a component that represents a tail in the composition, he/she will add a related tail descriptor in the lizard description. The entire lab runs over the web on top of the *Componere* authoring environment[36], which explores the Rich Internet Application (RIA) approach to provide an interactive interface.

The remaining of the paper is organized as follows: Section 2 presents a taxonomic description model of the context of the developed tool. Section 3 presents implementation details of the tool. Section 4 presents future works and conclusions.

4.2 Taxonomic Description Model

The starting point for designing our lab was the software Xper2 (http://lis-upmc.snv.jussieu.fr/lis/?q=en/resources/ software/xper2/). This tool supports the identification and description of specimens. It follows the character/character state (C,CS) [26] approach, organized in three phases:

(i) to define descriptors and possible states;
(ii) to relate descriptors/states to species;
(iii) to identify a given specimen by recognizing values for each descriptor.

The Varan-ID system was developed over the Xper2. Its descriptors are organized in two distinct groups, easy-to-see descriptors and expert descriptors. Table 1 shows an example of some easy-to-see descriptors and their respective states.

Table 4.1: Easy-to-see Descriptors

Lizard Part	Descriptor	States
Tail	transversal section of the tail	roundish or laterally compressed
Head	position of nostrils between eyes and tip of snout	same distance from eyes than from tip of snout or nearer the tip of snout than the eyes or nearer the eyes than the tip of snout
Tongue	tongue coloration	red, light pink or whitish or blue, purple or black

Our tool is able to interact with Xper2 by accessing SDD files it can export. The Structure Descriptive Data format (SDD) is an open standard endorsed by the TDWG

(Taxonomic Database Working Group) and DELTA (Descriptive Language for Taxonomy) for representing taxonomic descriptions in a XML format (http://wiki.tdwg.org/SDD/).

Figure 4.1 shows a diagram representing a fragment of a SDD file containing data to describe Varanus lizards. The hexagons represent elements, the rectangles represent texts and the ovals represent attributes. The CategoricalCharacter element defines a descriptor and possible states. In this example it defines the tongue coloration element and three possible states: red, light pink or whitish and blue, purple or black.

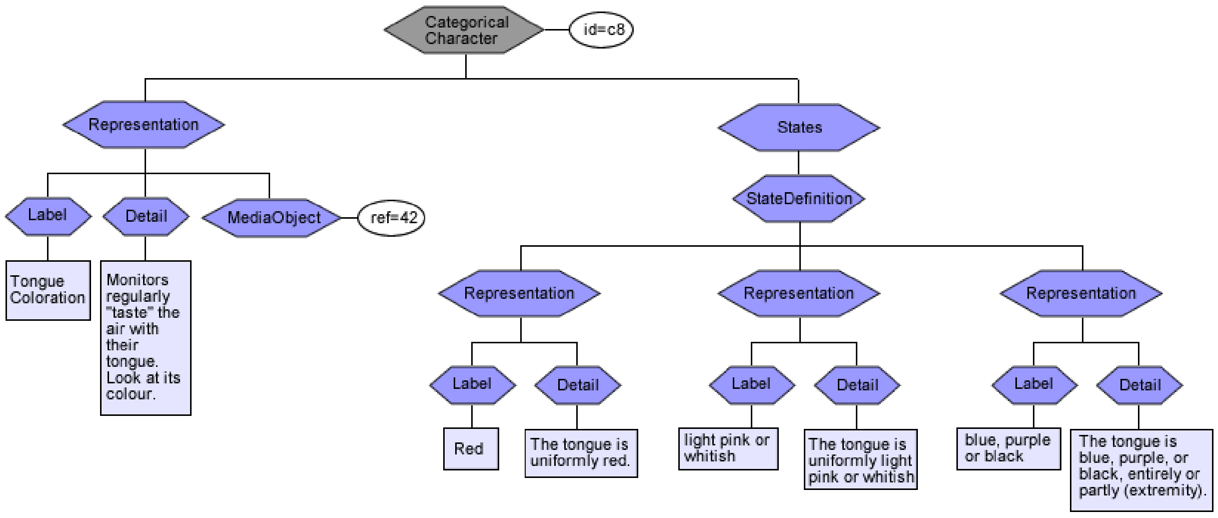


Figure 4.1: A Fragment of a SDD file of Varanus Lizards

The States element aggregates all possible states: StateDefinition elements. As can be seen in the diagram, the Representation element can be applied in many levels of the schema, containing textual and multimedia descriptions. This element is formed by a label, a detailed description (Detail) and references to multimedia resources (MediaObject)

In our tool we map these SDD description blocks in the following way:

(i) each CategoricalCharacter becomes a description component;
(ii) the set of states that the CategoricalCharacter can assume is transformed in a set of possible states that the component can assume;
(iii) every time the description component assumes a state it provides a visual feedback.

These description components transform the task of describing specimens in selecting and customizing components, which are combined in compositions.

Our resulting composition reflects another part of the SDD representation, illustrated in Figure 4.2. Besides the Representation element, described before, the diagram shows the summary data of the lizard. This element contains the element Categorical that represents the description of a given specimen (a lizard in the example), specifying Categorical

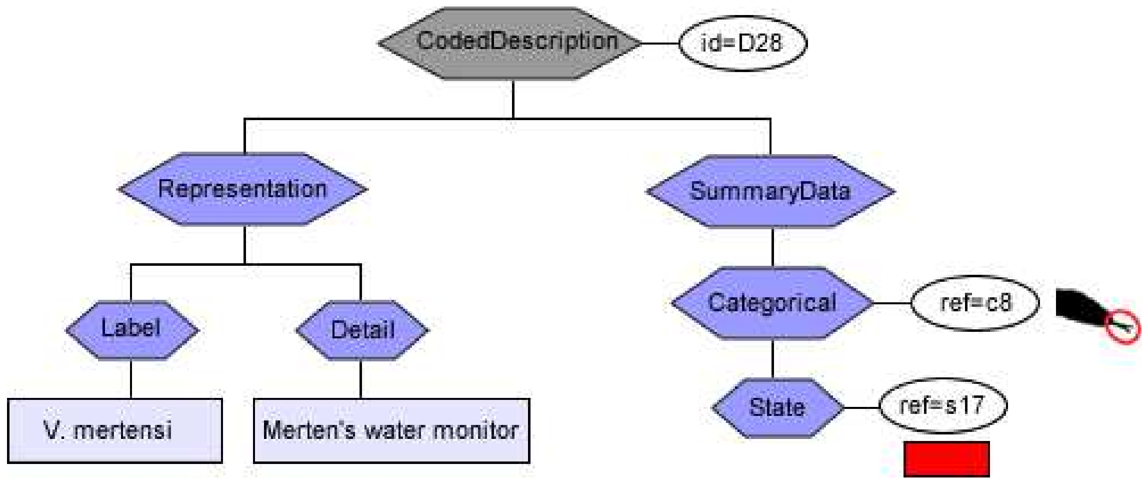


Figure 4.2: The Structure of Lizard SDD Base

Characters – referenced by the Categorical element – and setting values to it, through the State element. In this example, the categorical character ”tongue coloration” assumes the state ”red”. Since each composition specifies the description of a given specimen, the resulting composition can be mapped to a SDD structure presented in Figure 4.2 and vice-versa.

In order to automatically derive SDD Categorical Characters to description components, a process was created to get all information on the SDD file and use it to fetch the respective description components, customizing them with the respective values. Figure 4.3 illustrates the creation process of lizard components and their use during a composition. A proxy reader component accesses a SDD file (step 1). This information is delivered to a component generator that creates all the description components (step 2 and 3) – lizard description components in the example. Therefore, the categorical element will be used to identify a specimen characterization – a lizard in the example – to generate its respective composition. The State element will generate the value that the parameter assume.

The author customizes and combines components, building a composition, whose parameters are recorded by the Settings component (step 4). When the author clicks in the Candidate button it triggers the Result component (step 5), which in turn retrieves the parameters from the Setting component (step 6), and uses the proxy component to fetch data from XML files in the base (step 7); which uses the HTTPXMLRequest API (step 8).

In our proposal, a description component is one type of the available components. Another type comprises tool components, a group that will be responsible to support the authoring process. The classification based on this two types of components is illustrated

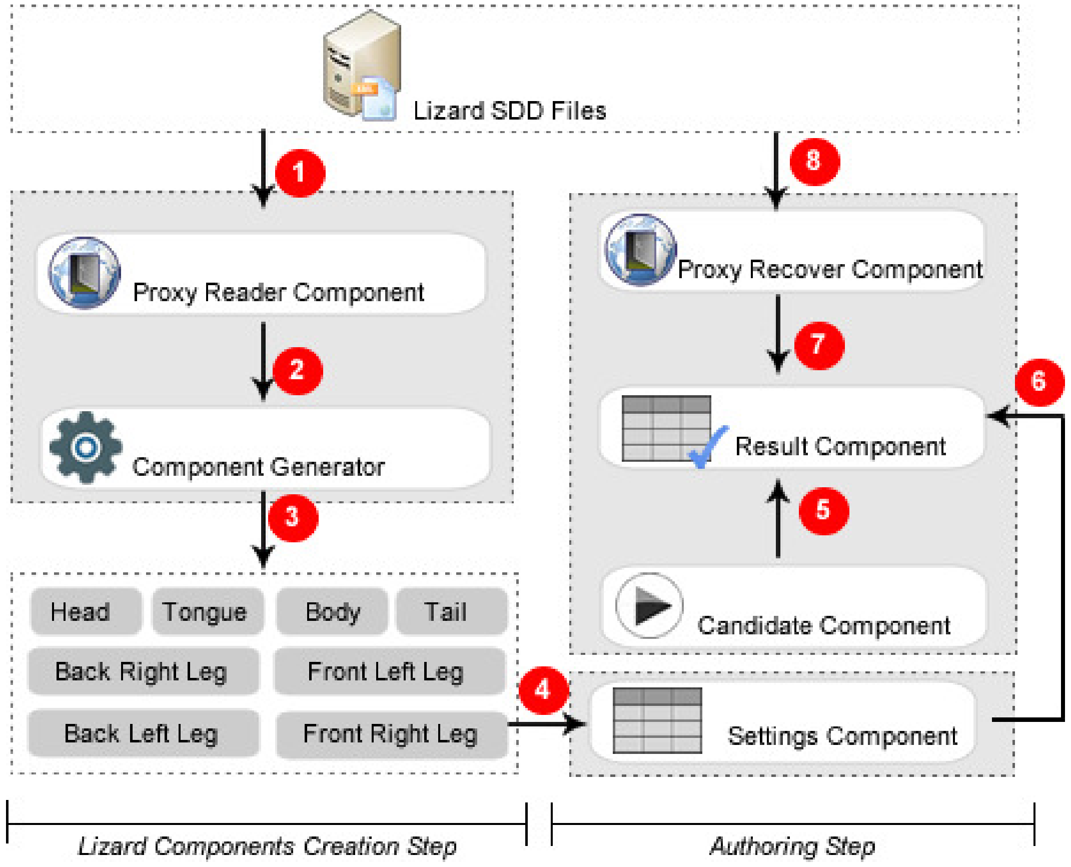


Figure 4.3: The Creation of Lizard Components and Composition Process

in Figure 4.4.

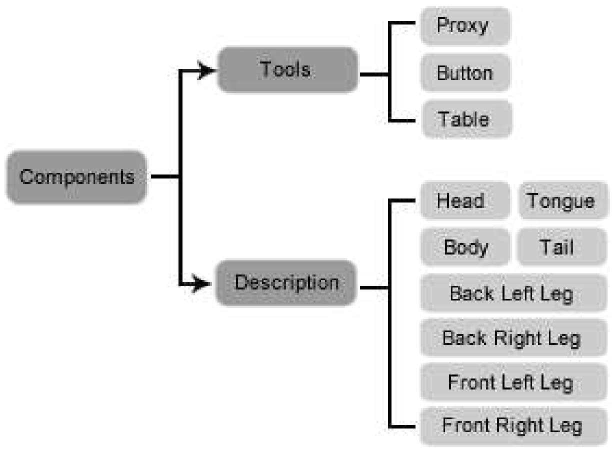


Figure 4.4: The Component Classification for the Lizard Lab

Our lab is designed to afford any kind of description component for living beings.

However, in our prototype we have produced only description components representing each part of the Varanus lizard. They are visual components derived from Varan-ID easy-to-see descriptors. The tool components can be visual or not. For example, the button is a visual component that starts the execution process. The table is a visual component that contains data organized as rows and cols. The proxy is a non visual component that brings data stored in a database to the composition.

As mentioned before, our laboratory is built over the *Componere* environment. In the original *Componere* authoring environment all components play the role of building blocks. Our lab, on the other hand, introduces this new kind of component - the tool component - to assist the authoring task itself.

4.3 Implementation

The implementation of the proposed laboratory involved two steps: the construction of description components and the construction of the laboratory on *Componere*. As mentioned before, this first prototype is focused on a specific practical scenario involving the identification of Varanus lizards, based on the Varan-ID database.

4.3.1 Construction of Lizard Components

The Varan-ID base is composed by 7 knowledge bases: the Main base, V. indicus-group base, V. prasinus-group base, V. timorensis-group base, V. gouldii-group base, V. salvator-group base, and the Australian spiny-tailed base. Each base can be exported as an SDD file. To build the components, we analysed the available descriptors in these bases.

A component builder engine was developed to extract information from the exported database and to use them to build each one of the lizard components.

There are two ways to build *Componere* compositions. The first is by using a javascript code to instantiate and to connect components. The second is by embeding compositions in HTML pages through microformats based specifications [36]. During the authoring process the laboratory uses the first dynamic approach. Resulting compositions can be further materialized as HTML embedded compositions.

4.3.2 The Laboratory on Componere

As mentioned before, *Componere* is a framework based on javascript components that works over web browsers.

Thus, the Lizard Lab is an environment totally based on javascript, mainly directed to beginners in the monitor lizard identification process. An overview of the system is illustrated in Figure 4.5.

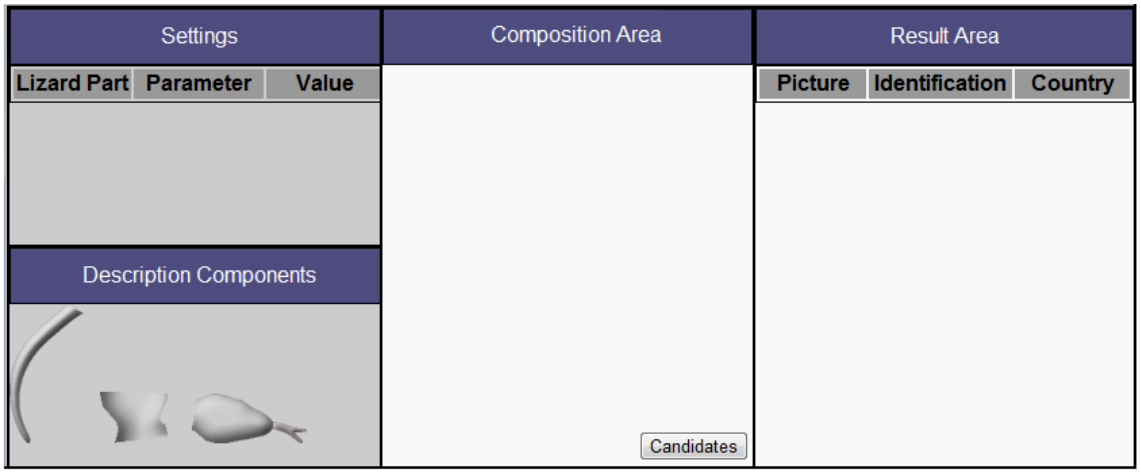


Figure 4.5: The Model of the Lizard Lab

The environment is organized in four areas: description components, settings, composition and result area. The description components area is where the components representing parts of the lizard are placed. Each one is independent and has its own set of parameters to be configured.

In order to produce a description the author drags description components to the Composition Area where they are customized and connected. Each description component has two basic main actions: close and configure. The first will remove the component of the composition. The second will open a dialog box with the parameters to be configured. Whenever a component is changed, the Settings area (displayed in Figure 4.5) stores and shows a log of the values assigned to descriptors. Figure 4.3 illustrates this relation in the process.

During the description/identification process the author can access the base containing available descriptions of existing species – lizard in this case – whose descriptor/states match with those already assigned in the lab. For example, if the author assigns a specific tongue color and tail shape, the system will record these settings in the Settings area; whenever the author clicks in the "Candidates" button, available in the environment (see Figure 4.5), the system fetches and presents all lizards in the base which have the informed tongue color and tail shape. The steps to execute this process is illustrated in Figure 4.3. This technique to present progressive candidates is based in Xper2 approach to describe specimens.

The prototype is available at http://fluidweb.sourceforge.net. The page has many experiments using different kind of software components. To access the work proposed in this paper, click on the link "Lizard Prototype".

4.4 Related Work

Our work combines two approaches: virtual laboratories and tools to describe and identify specimens.

According to [25, 30] the laboratories can be classified in three categories: real, remote and virtual. Real labs are physical rooms, having concrete equipments and infrastructure. Remote labs enable access to physical resources of real labs by networks – as the Internet – through a simulation software, which replicates the remote environment. Virtual labs have the goal of offering a simulation environment to support virtual experiments. Our proposal can be considered a mixture of the three contexts, since it is a virtual lab that grabs data from the real world and is built over the web.

Laboratories usually offer specialized resources according to the context they are inserted. [13] proposes an educational environment for electronics and electrical engineering. [14] presents an environment for genetics learning. [15] presents a virtual lab of Chemical Vapor Deposition aimed to complement a physical laboratory in the undergraduate course curriculum. [31] proposes a virtual laboratory for medical digital analysis based on grids.

The tool to describe specimens – as Xper2, detailed before, and Lucid (http://www.lucidcentral.org) – are designed for specialists and does not follow a laboratory approach. As far as we know, there is no such a tool combining the characteristics of virtual laboratories and biology description/identification tools.

4.5 Conclusions

In this paper we present our virtual laboratory based on virtualized visual components. In other to validate it, we have implemented a prototype of the virtual lab to describe monitor lizards of the genus Varanus based on a system called Varan-ID. The environment allows direct manipulation of visual components that work as basic elements of an authoring process, to support the identification and description of living beings. These basic elements derives from descriptions present on the Varan-ID database. The proposed tool runs over the web on top of the *Componere* framework.

The main contribution of this paper is our unified approach to produce a virtual lab for taxonomic description, combining the perspective of tools to describe specimens with the virtual laboratory model. It involved the design of a new component based description approach, in which components work as basic descriptive building blocks.

Future works include to expand the laboratory features, enabling it to better integrate with real world resources, i.e., fetching images and other kinds of media of real world specimens, including them in the description process. We are also working to generalize the process of building description components, enabling smooth expansion to new descriptors and other domains.

4.6 Acknowledgments

This work was partially funded by CAPES, CNPq, FAPESP, CAPES-COFECUB (AMIB project) and INCT in Web Science (CNPq 557.128/2009-9).

Chapter 5

Metalaboratory Technical Aspects

5.1 Introduction and Motivation

Sciences teaching – for example, chemistry, physics and biology – should not be dissociated from Science foundations. Therefore, it involves going beyond the presentation of results achieved by Science and systematized in disciplines, giving to the learner the opportunity of knowing the scientific method adopted to achieve such results. The cycle including observation and experimentation in the real world (analysis) to the proposition and validation of models that describe them (synthesis) can be considered the most relevant aspect to learn this method.

Taking a laboratory perspective, the analysis can include experiments in laboratory or field observations. The synthesis usually involves models that describe the world, anchored in examples based on everyday observations. It is possible to go from the analysis to the synthesis, or vice-versa. There are several scenarios in which researchers wish to compare collected data (analysis) with results from models (synthesis). This is interesting, for example, to verify if a model is in accordance with a respective description of an observed phenomena in the real world.

This notion of hybrid laboratory was idealized by [10]: an environment where it is possible to confront theory and experimental knowledge by combining virtual laboratories (theoretical models) with remote laboratories (real world experiments).

In the Education context, there are specialized tools for analysis and synthesis tasks. On the analysis side, a combination of hardware and software takes the form of kits that are able to collect, interpret and validate data from real world – e.g., Arduino and its Processing language [32]. In the synthesis side, there are specialized software to implement and execute scientific models, for example, based on: mathematical modeling – e.g., Modellus [29] –, simulation programs – e.g., Interage Simulations [37] – and virtual labs. However, tools are segmented and it is difficult to integrate and compare results.

In parallel, we are facing the emergence of a new data-intensive paradigm, in which "In almost every laboratory, 'born digital' data proliferate" [5]. A deluge of this data is available on the Web, becoming possible to produce remote asynchronous[1] laboratories, settled on existing Web data. This notion of asynchronous labs and the respective technique to implement them arose as consequence of this work.

Not only data is available on the Web, but also tools and services, which can be in many cases "mashed-up". This work exploits this potential of the Web as a basis to build laboratories. We propose a metalaboratory approach that treats Web resources as combinable building blocks to build Web laboratories.

Departing from an existing authoring environment, the Componere [36], we contribute in this work with three techniques envisaged to support lab authoring, but which can expanded to other contexts: a metalaboratory family of components, laboratory composition patterns and Active Web Templates.

The remaining of the document is organized as follows: Section 5.2 presents the related work. Section 5.3 describes our metalaboratory proposal. Section 5.4 presents our conclusions and future work.

5.2 Related Work

Related work comprises laboratories and metalaboratories (section 5.2.1) and a framework called Componere (section 5.2.2).

5.2.1 Laboratories and Metalaboratories

Laboratories can be classified into three categories: real or physical, virtual and remote [25][30]. Real laboratories are the classic one, with physical rooms with real equipment and infrastructure. Remote laboratories use physical resources of real laboratories through sensors, actuators and a communication infrastructure, transferring to computers the task of controlling experiments, capturing, transforming and storing data. The term remote here refers to computers connected to lab equipment, but apart from them. Therefore, "remote" can refer to a computer in a lab which interacts with equipments inside the same physical lab. Remote labs can be applied, for example, in manufacturing cells [12] or perform experiments in Digital Signal Processors (DSPs) through the Web [4]. Virtual laboratories can simulate the physical equipments behaviour and all the infrastructure of a real laboratory using computational techniques. They can be applied to Chemistry [15] and Medicine [31].

[1]The term asynchronous here refers to the fact that data from remote labs are not retrieved as soon as they were produced, but afterwards from archives – in our case, shared through the Web.

The term metalaboratory received different interpretations in related work, making difficult to achieve an unified definition. The term is used in [23] to describe a tool for the composition of technologies and services aimed at teaching chemistry. In [11] the term is used in a virtual laboratory perspective. A group of related work adopt the interpretation of metalaboratory as an hybrid laboratory [34] [2] [3].

Cramer and De Meyer [10] introduced the idea of interacting virtual laboratories with real laboratories to bridge theoretical models and experimental data. Authors present an interesting perspective of a symbiotic connection between real world and virtual models, comparing it with the way our brain works producing models over the observable world.

We will use the term metalaboratory in this work to refer to a laboratory to build laboratories. It is aligned with Cramer and De Meyer [10] perspective, as it can be applied to build hybrid laboratories.

5.2.2 Componere

Our metalaboratory is built over a pre-existing authoring environment called Componere, which is a Web authoring system based on components fully running on browsers [36]. It is based on the concept of multimedia authoring systems, which allow authors to create multimedia products by using a combination of multimedia primitives [7].

Figure 5.1 shows the authoring process in the Componere environment. It is organized in three stages: primitives building, authoring and execution. In the first stage, primitives are built as components and stored in a library. New components are built by software developers or end-users according to the type. Software developers are in charge of those components which require programming. They are based on the Digital Content Component (DCC) model [36].

There are two types of DCCs: Passive DCCs and Process DCCs. The first one encapsulate contents, like images, videos and texts in a Complex Digital Object (CDO) format. The second one encapsulates executable software that handles the Passive DCCs. This structure provides independence of the content from a given implementation, fostering the management, sharing and reuse of components.

In the authoring stage, the user combines the available primitives to produce compositions. Components are the raw material for this stage, which involves customization and composition of components. The final product of this stage can also be stored in a library. Finally, in the last stage, the compositions are executed over a browser.

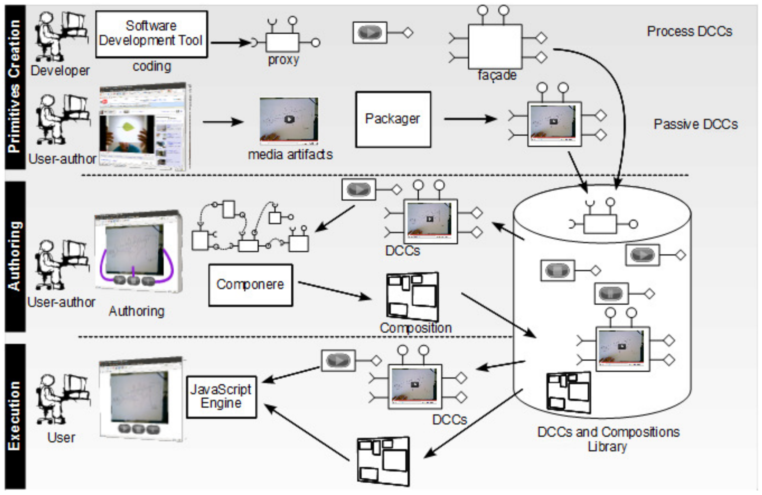


Figure 5.1: Graphical representation of Componere [36]

5.3 The Metalaboratory Proposal

Our metalaboratory is an environment to produce tailored laboratories, treating software components as combinable building blocks. It offers the possibility of building three types of laboratories: virtual, remote(synchronous and asynchronous) and hybrid. Virtual labs are defined as environments where the experiments will be driven by simulations based on computational models – e.g., math models. It uses synthetic (virtual) data.

In remote labs, experiments are based on data from an external online (synchronous) or off-line (asynchronous) source – e.g., data captured from a sensor which can be stored in a file. It offers the possibility of studying a phenomena by making a bridge between the real and virtual worlds.

Finally, a hybrid lab is an environment where virtual and remote experiments are put together. It offers the possibility of studying how results of the real world differ from results of synthetic models and how to abstract synthetic models from real world data.

Our approach to conceive a metalaboratory departing from an authoring tool is based in three elements: a metalaboratory / laboratory Web components family, laboratory composition patterns and an active Web template based technique. In the following sections we detail these elements.

5.3.1 Metalaboratory / Laboratory Web Components Family

In the following sections we will present the concepts of components and the component family of the metalaboratory.

Metalaboratory Components

Our approach aims to support the composition of laboratories through composition of components. Szyperski [39] presents a definition of software component: “A software component is a unit of composition with specified contractual interfaces and explicit context dependencies. A software component can be deployed and is subject to composition by third parties”. To [43] component is “an independently deliverable unit that encapsulates services behind a published interface and that can be composed with other components”.

A software component integrates data and functions and implements interfaces, which can be interpreted as an implementation of specifications of services that are provided to or required from other components [6]. Interfaces make explicit the protocol to exchange data among components. A component can have multiple interfaces [24].

Our environment is based on Componere and its DCC model [36]. DCCs comply with the principles of software components presented before, but expand them to afford content components. They are internally organized as complex objects and externally they act like software components.

This work contributed in reviewing the DCC framework running in the Javascript platform, originally proposed and implemented in [36]. The javascript language has some design limitations, which hampered the proper application of the DCC component model. Two highlights are:

- Javascript lacks a namespace mechanism. It makes difficult to limit the scope of variables. This is a central issue to reuse third party code in components.
- The language does not provide primitives to selectively load code modules.

For these reasons, the new framework adopted the YAHOO YUI as its bases. The YUI provides a set of improvements in the Javascript fundamental operations, plus a library of modules and widgets. Among the improvements adopted in this work, we highlight:

- A library to support namespace management.
- A mechanism to manage dynamic module loading.
- Dependency management among modules.

Besides the language improvements, our framework encapsulated as components (DCCs) several widgets provided by the library.

DCCs declare through interfaces externally configurable properties, operations and events. Our laboratory components family departed from a basic DCC configuration, presented in Figures 5.2 and 5.3, which was extended for all components. Figure 5.2

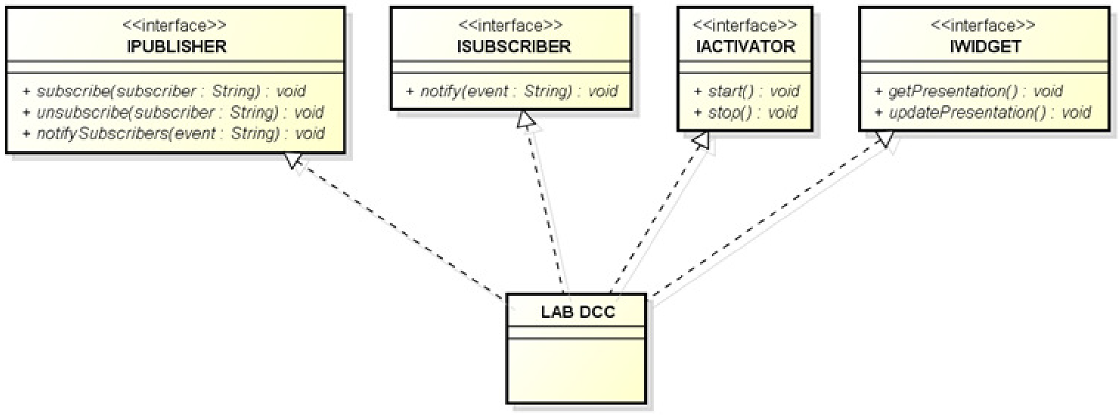

<<interface>>
IPUBLISHER
+ subscribe(subscriber : String) : void
+ unsubscribe(subscriber : String) : void
+ notifySubscribers(event : String) : void
<<interface>>
ISUBSCRIBER
+ notify(event : String) : void
<<interface>>
IACTIVATOR
+ start() : void
+ stop() : void
<<interface>>
IWIDGET
+ getPresentation() : void
+ updatePresentation() : void
LAB DCC

Figure 5.2: Basic Structure of a Laboratory Component

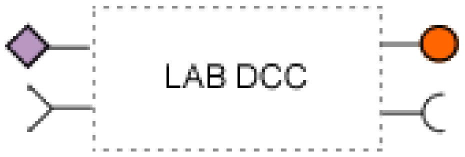


Figure 5.3: Visual Structure of a Laboratory Component

presents some basic interfaces implemented by components. Figure 5.3 presents a component diagram and the symbols to represent interfaces.

In Figure 5.3, the dashed box represents a DCC, the headed pin and semi-circle headed pins represents the provided / required interfaces, respectively. This pair of interfaces will be plugged by a connector to work together. A component that implements a required interface requests services to the component that is connected to it. A component that implements a provided interface executes the requested services and delivers the results to the requester.

The diamond headed pin and semi-diamond headed pin represent the publish / subscribe interfaces, respectively. It is detailed in the IPUBLISHER and the ISUBSCRIBER interfaces (upper). Components that implements the IPUBLISH interface are the publishers and sends to subscribers messages when a determined event occurs, interested subscribes request subscription by the subscribe method. Components that implements the ISUBSCRIBE interface are subscribers. These components register their interest in an event and will only receive messages of this particular event through the notify method. Table 5.1 presents a description of the methods.

Over the defined basic structure, new operations can be defined according to the role a component will play. For instance, a component that will contain a video can declare operations like play, stop and pause.

Table 5.1: Metalaboratory Component Interfaces

Operations	Interface	Definition
`subscribe()`	IPUBLISHER	Register all the components that are interested in the messages posted by the publisher
`notifySubscribers()`	IPUBLISHER	Send a message to the registered subscribers
`notify()`	ISUBSCRIBER	Receive the delivered messages
`start()`	IACTIVATOR	Starts the action of a component
`stop()`	IACTIVATOR	Stops the action of a component
`getPresentation()`	IWIDGET	Returns an object with the visual appearance of the component
`updatePresentation()`	IWIDGET	Implements a refresh operation of the appearance

Metalaboratory / Laboratory Components Family

In order to systematize the laboratory production, we define a family of metalaboratory and laboratory components, aimed to capture atomic functionalities of a lab.

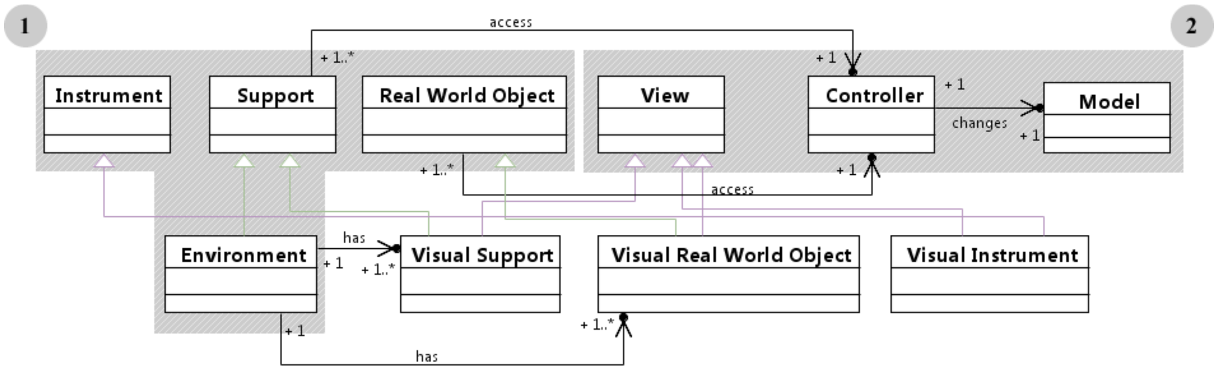


Figure 5.4: The Metalaboratory Class Diagram

DCCs are organized in OWL classes according to their functionality. Each DCC class defines the semantics of a component in a taxonomy and the provided/required interfaces. Our family of metalaboratory components is systematized around a set of classes illustrated in Figure 5.4. We adopted an UML visual approach to represent OWL classes. There are two groups of classes (groups inside gray boxes in the figure):

- Group 1: related to the role of a component in a laboratory – **environment, support, instrument and real world object**

- Group 2: related to the component profile according to the Model-View-Controller (MVC) pattern.

The classes were designed as complementary classifications. A component typically specializes one class of each group. Therefore, we have: visual instrument, real world model etc. A DCC belonging to a class means it complies to a set of expected interfaces, related to the role of the component and the way it interacts with the environment and other components.

In order to detail each group of the family, Figure 5.5 expands Figure 5.4. The numbers in the figure link to subsequent figures with more details.

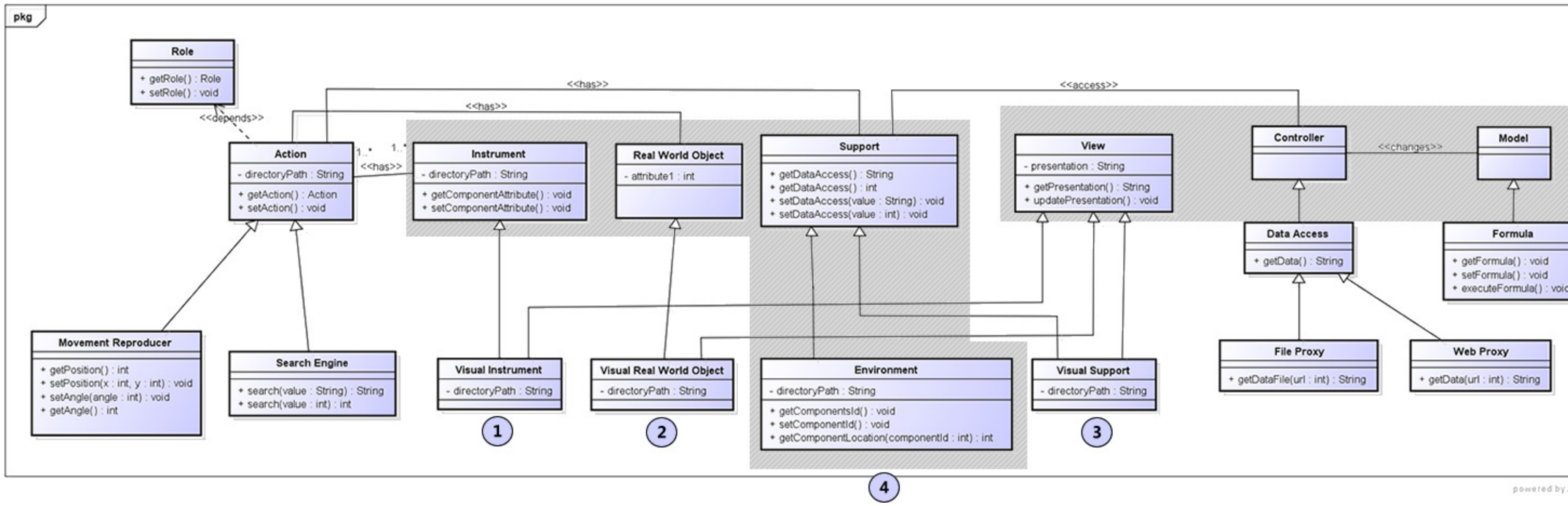


Figure 5.5: Metalaboratory Class Diagram - Main Part

Table 5.2: Group 1 – Laboratory Role

Component	Description
Instrument	Virtual equivalents of laboratory instruments. Authors can make them available in a laboratory to perform, for example, measurement tasks.
Support	Perform auxiliary control and management tasks.
Real world objects	Components that simulate or reproduce the behaviour of objects in the real world.
Environment	Virtually represent the space in which the laboratory runs. It performs tasks and provide information of the environment.

Table 5.3: Group 2 – MVC

Component	Description
Model	Represents the components that deals with different application data, such as files, formulas, or structured / raw data.
View	Manages outputs and user inputs through the application interface. Responsible for visual presentation of data.
Controller	Mediates the relation between model and view, and acts in tasks for controlling the whole process.

The MVC pattern has the goal of separating user input from data access and business logic in three roles: model, view and controller [8].

Visual support represents components that have visual characteristics and plays support roles, such as buttons or tables (see Figure 5.6).

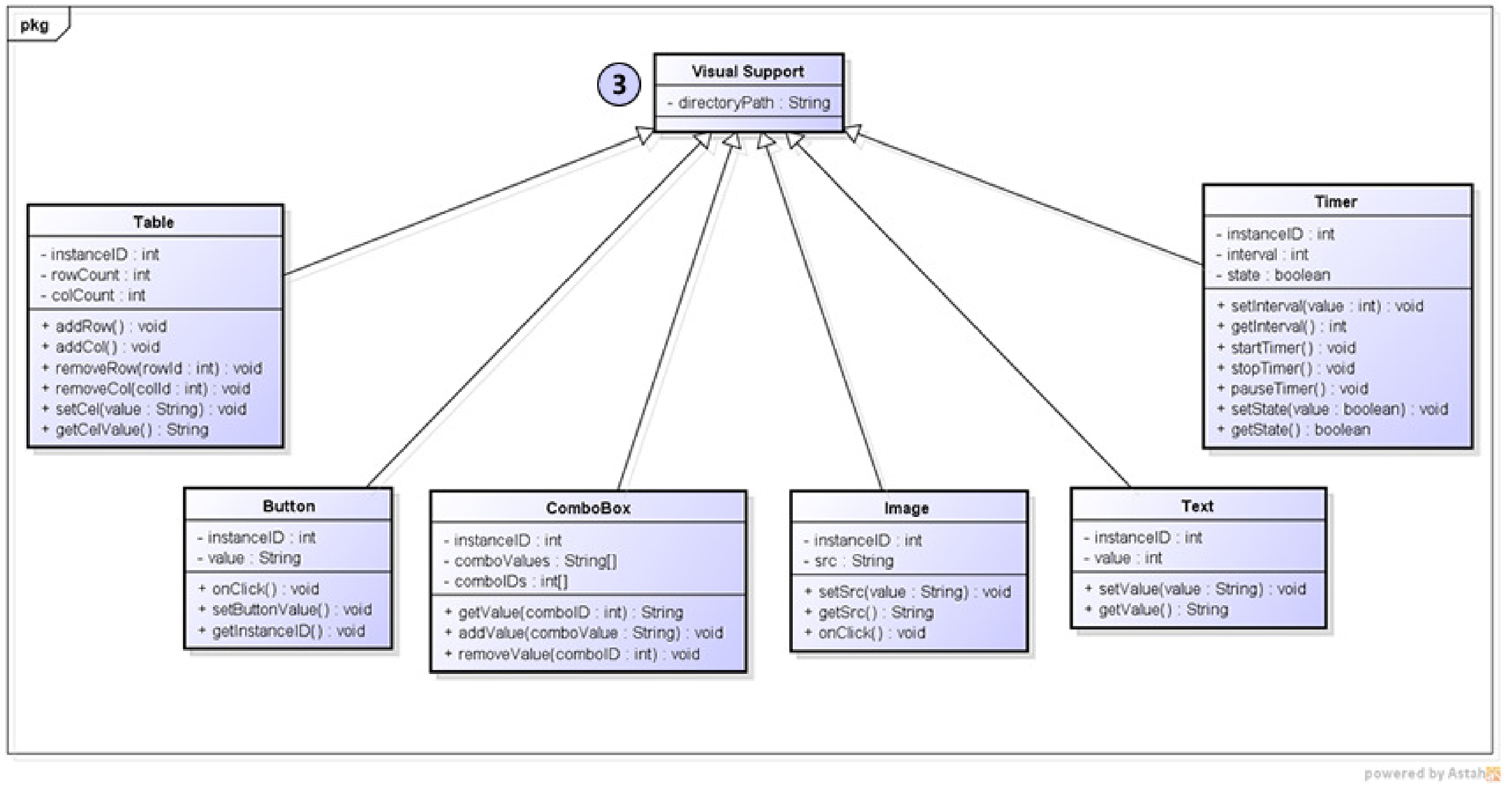


Figure 5.6: Metalaboratory Class Diagram - Visual Support

Visual real world objects (see Figure 5.7) are components that have visual characteristics and plays real world objects roles. For instance, an animated ball or a pendulum.

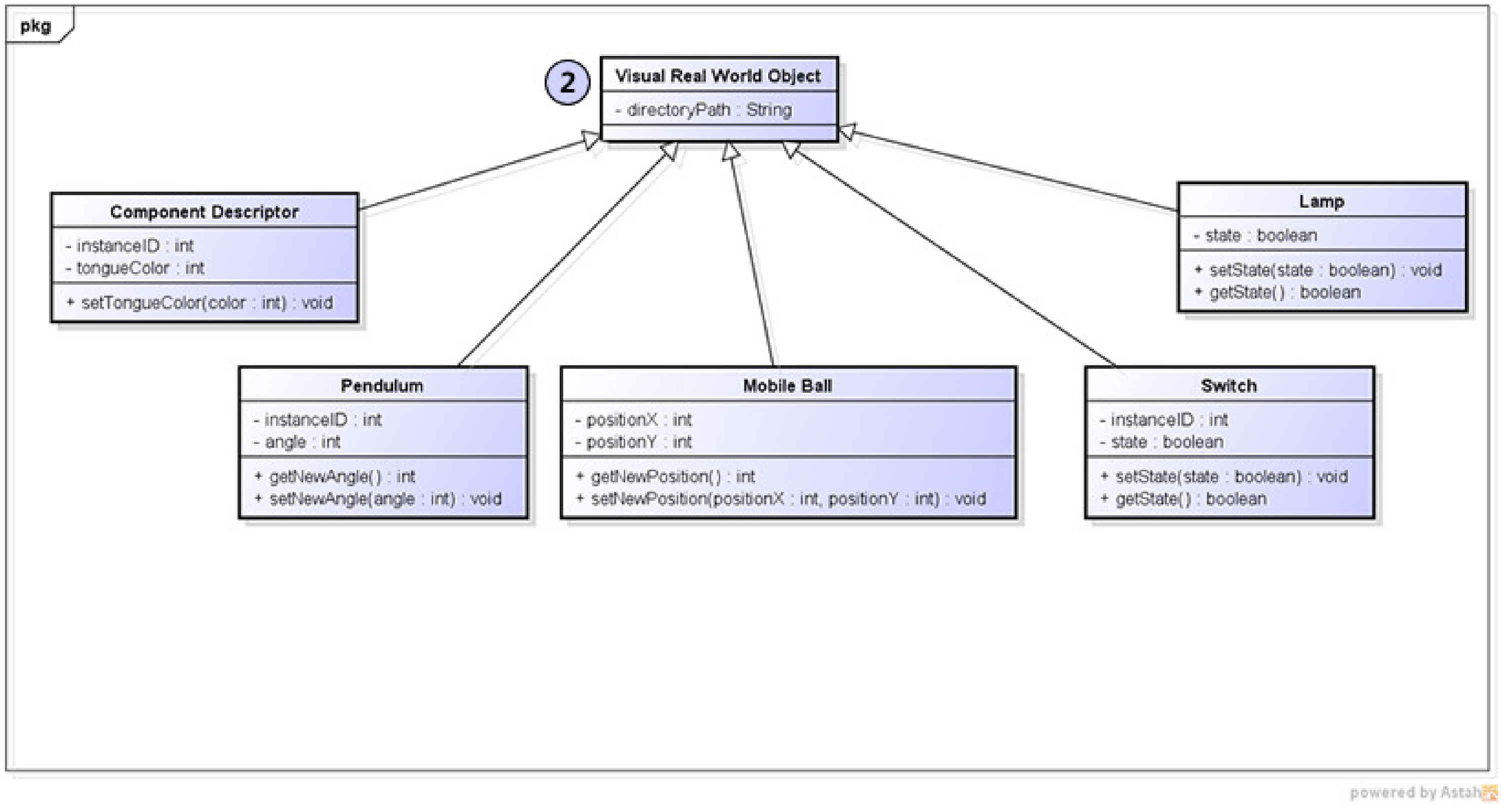


Figure 5.7: Metalaboratory Class Diagram - Visual Real World Object

Visual instruments (see Figure 5.8) can be managed by the user inside a lab, as a virtual instrument. For example, a magnifier will amplify a given image and the ruler can be used to measure objects.

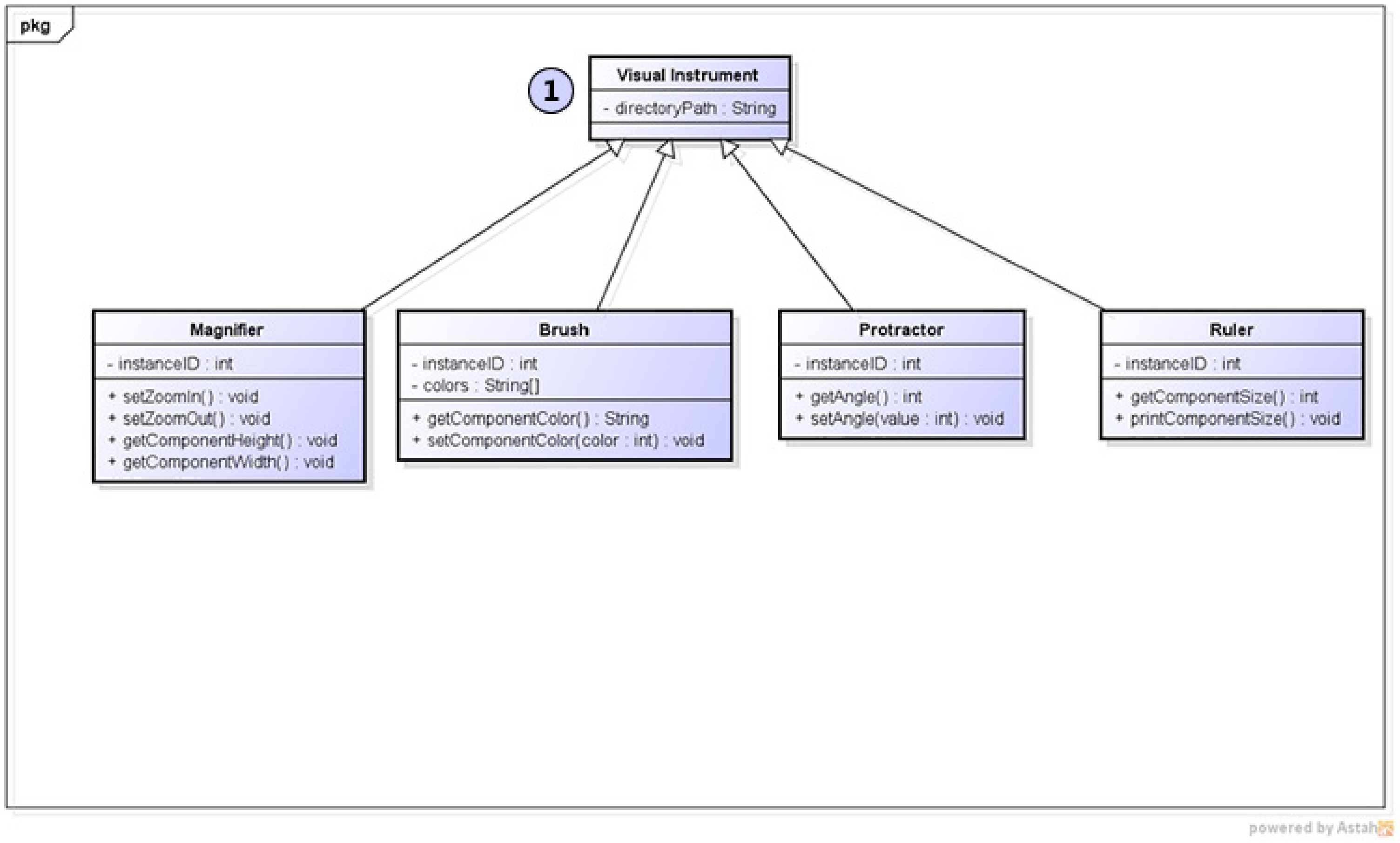


Figure 5.8: Metalaboratory Class Diagram - Visual Instrument

Environment (see Figure 5.9) is an independent set of DCCs that represent and manage "virtual spaces", in which other DCCs will run. A complete list with metalaboratory and laboratory components family is presented in Appendix A.

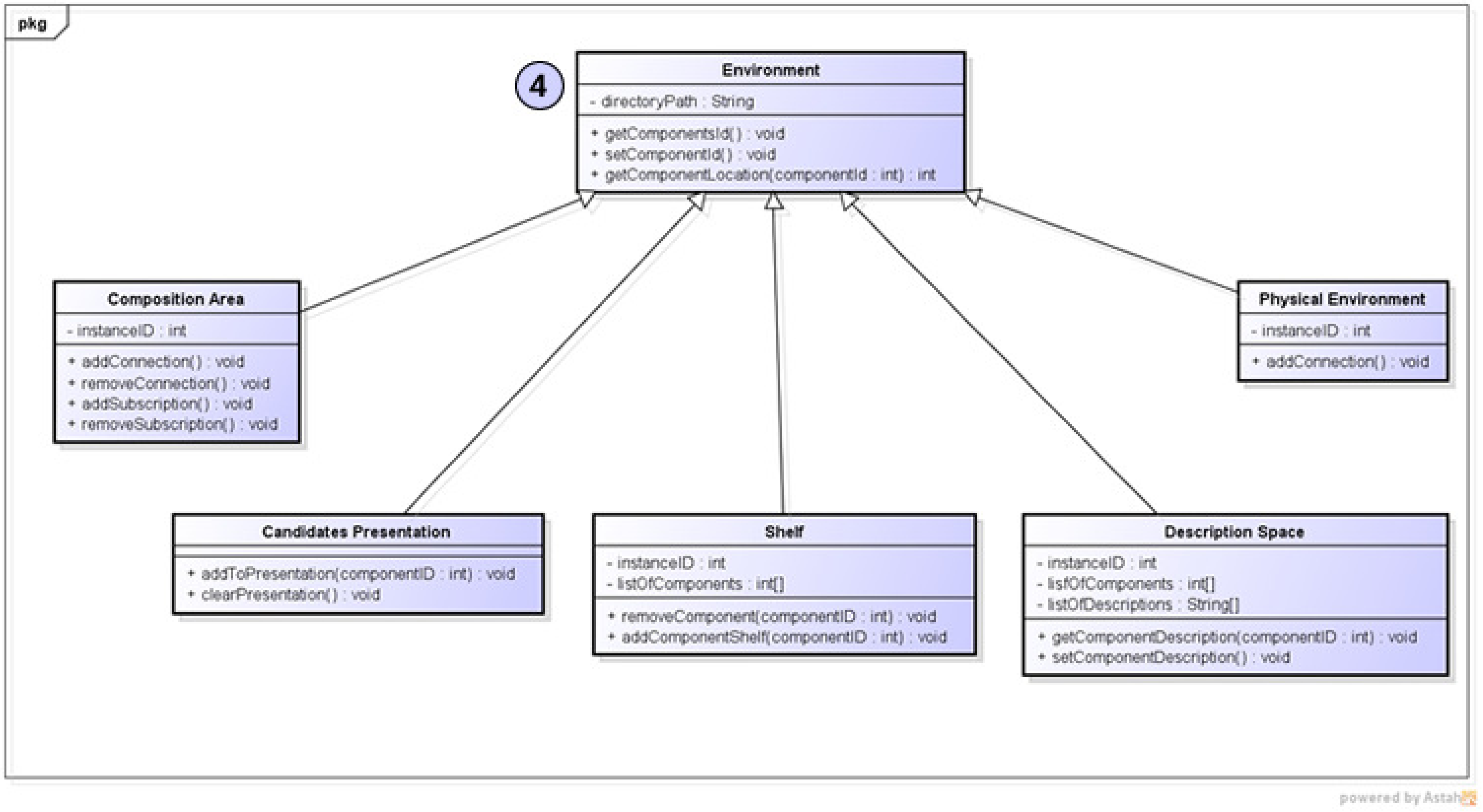

pkg
4
Environment
- directoryPath : String
+ getComponentsId() : void
+ setComponentId() : void
+ getComponentLocation(componentId : int) : int
Composition Area
- instanceID : int
+ addConnection() : void
+ removeConnection() : void
+ addSubscription() : void
+ removeSubscription() : void
Physical Environment
- instanceID : int
+ addConnection() : void
Candidates Presentation
+ addToPresentation(componentID : int) : void
+ clearPresentation() : void
Shelf
- instanceID : int
- listOfComponents : int[]
+ removeComponent(componentID : int) : void
+ addComponentShelf(componentID : int) : void
Description Space
- instanceID : int
- lisfOfComponents : int[]
- listOfDescriptions : String[]
+ getComponentDescription(componentID : int) : void
+ setComponentDescription() : void
powered by Astah

Figure 5.9: Metalaboratory Class Diagram - Environment

5.3.2 Laboratory Composition Patterns

Inside a composition, it is possible to identify patterns to compose laboratories. Consider the composition to simulate the motion of a pendulum, illustrated in Figure 5.10:

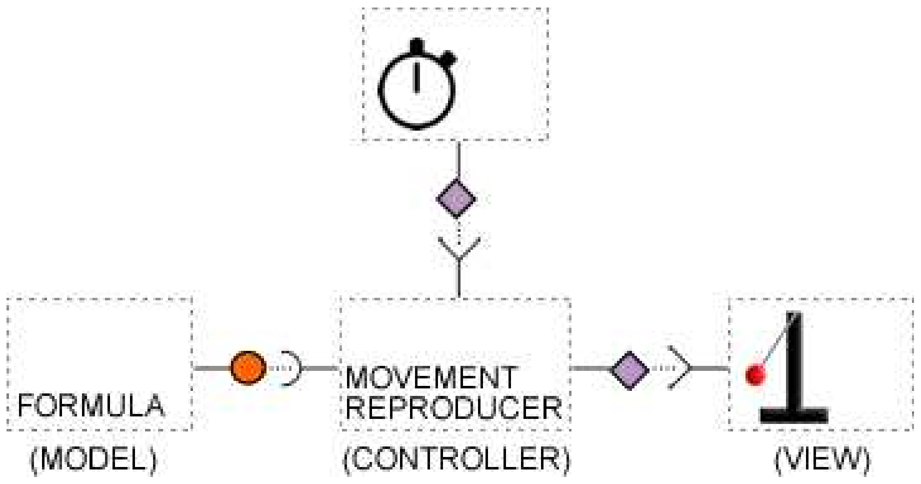


Figure 5.10: Composition to simulate a pendulum

The first component (**MODEL**) represents a formula aimed to calculate positions of a pendulum. The second component (**CONTROLLER**) retrieves each position calculated by the formula and trigger a position update in the Pendulum (**VIEW**). Its rhythm is defined by a **Timer** component. By replacing the model (**FORMULA**) by a data source, captured from a real world pendulum (Figure 5.11) the composition reproduces asynchronously a real pendulum, instead of a synthetic one. The data is retrieved by a pair of components: **XML Proxy** retrieves data from a XML remote source; **XML-to-Movement Converter** converts the data to pendulum positions. These examples allow to recognize our reproducer / simulator lab pattern.

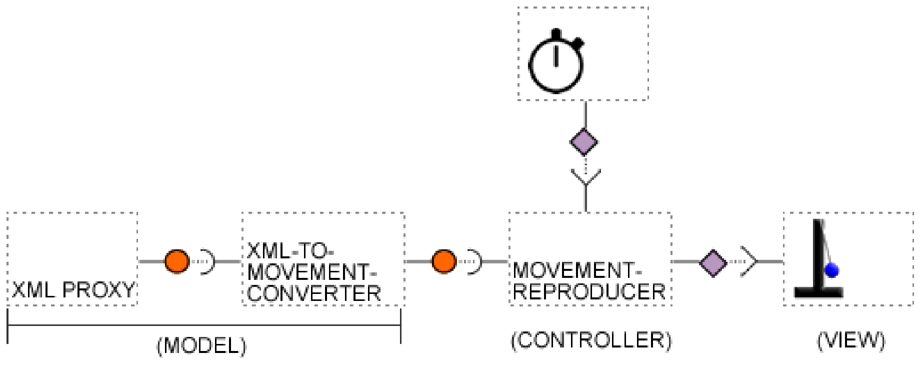


Figure 5.11: Composition to reproduce a pendulum movement

Figure 5.12 illustrates the behaviour reproducer / simulator lab pattern, which is proposed in this work. It is based in three tiers, which can be executed by one or more components. The model tier provides data concerning the movement (behaviour) – it can be a data source or producer; the controller tier consumes data from the model and manages the reproduction of the movement in the view tier, which in turn provides a visual animated real world object, apt to display the movement.

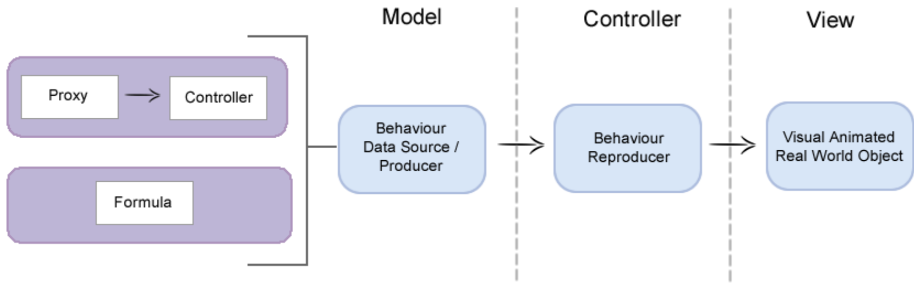

Model
Controller
View
Proxy
Controller
Formula
Behaviour Data Source / Producer
Behaviour Reproducer
Visual Animated Real World Object

Figure 5.12: Behaviour Reproducer / Simulator Lab Pattern

In the model tier we define two pattern specializations. The **Asynchronous Lab Pattern** combines a Proxy with a DCC to convert the input format to the expected by the reproducer. The **Virtual Lab Pattern** is a DCC which produces simulated values – in our example a pendulum movement defined by a formula. These patterns are fundamental to our metalaboratory approach, as they generalize common practices.

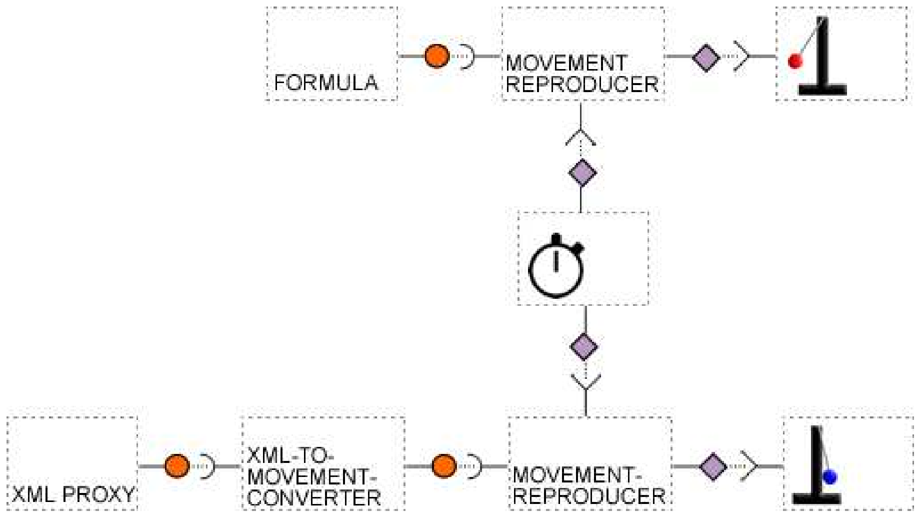


Figure 5.13: Synchronization Pattern

Consider a synchronization of the pendulums of Figure 5.10 and Figure 5.11; as illustrates Figure 5.13. The figure shows an example of our synchronization pattern. The basic idea is connecting all synchronization movements to the same timer. Other examples of lab patterns we produced is a control panel pattern. In order to capture and reuse such patterns, we developed a Web template based methodology.

5.3.3 Active Web Templates

Even tough it is possible to start a new laboratory from scratch, our observations show that laboratories tend to follow patterns according to their specificity, and that it is important to reuse not only components but also the composition design. Thus, we developed a technique based on Active Web templates, which are generalizations of Web documents blended with DCC compositions. They capture both, composition patterns and the lab layouts, in a document.

Active Web templates are an evolution of the existing approach, defined by Componere, to blend compositions in HTML. Therefore, we start by summarizing the main aspects of this approach.

Blending Compositions with Web Documents

In Componere, components become active elements, which can be inserted, customized and connected by tags in a HTML document. Microformats[22] are adopted to provide a smooth integration with the HTML structure. The pre-existing HTML `div` element and the attributes `class` and `name` are used to describe a DCC instance inside the Web document. Table 5.4 summarizes the main elements of the Componere / Microformats specification used to represent DCC instances and connections.

Consider the simple composition of Figure 5.14 in which a **Switch** component is connected to a **Timer** component. When the user clicks in the **Switch** it triggers the start of the **Timer**.

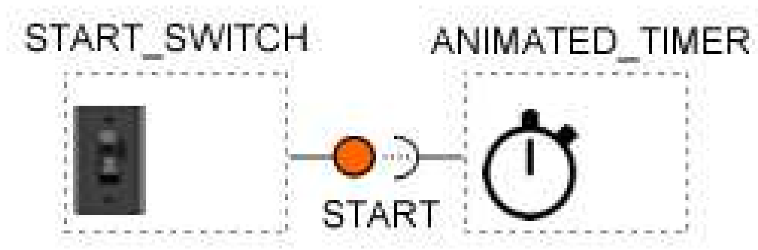


Figure 5.14: Switch / Timer Composition

As illustrated in the top of the components, each DCC receives an instance identifier in the composition: `START_SWITCH` for the **Switch** and `ANIMATED_TIMER` for the **Timer**.

Table 5.4: Componere/Microformats specification to represent DCC compositions.

HTML element	purpose	attr	purpose
`<div class="dcc">`	instantiates a DCC in the composition		
`<div class="property">`	defines the value of a DCC property	`name`	name of the property
div's content	value of the property		
`<div class="connector">`	connects two DCCs	`name`	name of the connection
`<div class="component">`	addresses a component instance involved in the connection – the connection is directed from the first declared DCC to the second one	`name`	role of the DCC in the connection
div's content	identity of the DCC		

The following HTML fragment represents an instantiation of the **Timer** DCC in the composition of Figure 5.14:

```
<div class="dcc">
  <div class="property" name="type">
     http://purl.org/dcc/dccdb/timer</div>
  <div class="property" name="id">
     Animation_timer</div>
  <div class="property" name="frequency">
     1000</div>
</div>
```

The property `type` refers to the type of the DCC – it is specified as a URI-based unique identifier. The property `id` indicates the unique id of the instance in the composition. The property frequency sets the timer frequency.

A connector is an independent element that refers to the two DCC instances involved in the connection. The following HTML fragment connects an instance of a **Switch** DCC and the **Timer** DCC, as illustrated in Figure 5.14.

```
<div class="connector" name="start">
```

```
 <div class="component" name="action">
    Start_switch</div>
 <div class="component" name="animation">
    Animation_timer</div>
</div>
```

Whenever the composition is presented to the user in a browser, a Javascript engine (100% running in the browser) converts the Microformats marks in Javascript code, which instantiates the DCCs and connects them. This approach enables mixing elements of a Web page and a composition and are the basis to build active Web templates. We define a three phases technique to produce laboratories from templates, illustrated in Figure 5.15.

From Active Web Documents to Active Web Templates

Active Web templates will be generalizations of common compositions + pages structure. A composition inside a template is like a regular composition, replacing configurable spots by a special kind of component we call meta-DCC. Templates can also have constant DCCs, which represent its pre-defined not variable spots, which are not designed to be replaced, even though, they can be further deleted and customized. The active Web template is meant to be executed during the second phase – authoring phase – illustrated in Figure 5.1. When a template is executed, the meta-DCCs provide personalized spaces to select and customize DCCs. Different kinds of meta-DCCs are programmed to support tailored authoring.

A meta-DCC is at the same time: (i) a surrogate of a future DCC and (ii) a software unit that will select and tailor this future DCC. As a surrogate (i), it ocupies in the Web document structure the specific spot where a future DCC will replace it. The software unit (ii) is triggered during the lab authoring, when a template is instantiated. Each meta-DCC becomes an active spot, containing an "editor component" specialized in selecting and tailoring DCCs to that specific part of the composition. Therefore, it is possible to have meta-DCCs specialized in: selecting data-sources, playing and customizing real world objects etc.

The presentation of our active Web template proposal will be based on a practical example, illustrated in Figure 5.15. It puts side-by-side two balls moving on an inclined plane. The left one captures and reproduces the movement of a real world movement and the right one simulates the movement based on a mathematical equation.

Figure 5.15 illustrates our three steps approach to produce labs based on active Web Templates. Step (a) shows the initial structure of the active Web Template. A template designer builds this template, usually by taking a pre-existing lab composition and generalizing it by replacing components by meta-DCCs. In step (b) an author starts by

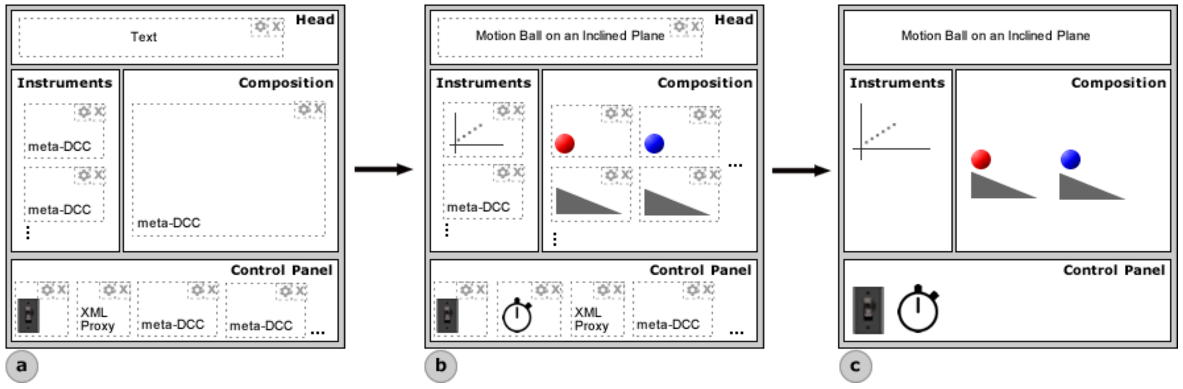


Figure 5.15: Metalaboratory Template

instantiating an active Web template. Whenever the template is executed, each meta-DCC is activated inside its position of the document. We call the template active since it is a runnable auto-configurable template. It contrasts with passive templates adopted by other approaches. The author customize the template through these meta-DCCs. According to their speciality, they allow authors to: select proper data sources, drag and position components in a canvas space etc. In the last step (c), an engine converts the template in a final Web document + composition. The template meta-DCCs give directions to the engine of the DCCs to be inserted and tailored.

The active Web template of Figure 5.15 is organized in four areas – head, instruments, control panel and composition – further detailed:

HEAD – Contains a regular DCC (not a meta-DCC) of a label to be customized by the author. In this case, a meta-DCC is not necessary since the type of DCC is pre-defined and the author will only customize its properties.

COMPOSITION – Has an unique wide meta-DCC of the class Canvas. It delimitates an area and allows freely drag and drop new DCCs inside this area during the authoring process. Components can be positioned in any part of the Canvas. When the template is converted in a composition, the wide Canvas meta-DCC is replaced by the DCCs dragged into it.

CONTROL PANEL – Defines a set of controlling DCCs organized in a panel. It presents two pre-defined DCCs – the **Button** and the **XML Proxy** – plus a set of meta-DCCs, which will select and customize future DCCs playing a control role.

INSTRUMENTS – Has a set of meta-DCCs, where the author can select and customize DCCs that will work as instruments in the laboratory, during the execution.

In the HTML point of view, an active Web template is represented by using the same Microformats approach, detailed in the previous subsection. Meta-DCCs are represented by a special `<div class = "metadcc">`. The rest of the configuration is equivalent to a DCC. The example of Figure 5.15 is organized in four HTML areas, delimited by divs and layouted by CSS.

This example show how meta-DCCs and DCCs are adopted to define pre-defined and configurable areas in the template, as well as their interaction with the HTML and CSS constraints. A template defines a skeleton of work in the authoring process, however, authors are free to insert, delete and reconfigure any available component and connection. When the author finishes the authoring process, meta-DCCs will be replaced by customized DCCs. Unused meta-DCCs will be deleted. The composition becomes the actual laboratory.

In the next section we will present step-by-step how metalaboratory is executed with our proposal of laboratory pattern and active Web templates.

5.3.4 Authoring Laboratories

A laboratory authoring cycle by using a metalaboratory is illustrated in Figure 5.16, as a state machine diagram. It is formed by three states: loading workspace, authoring and presentation.

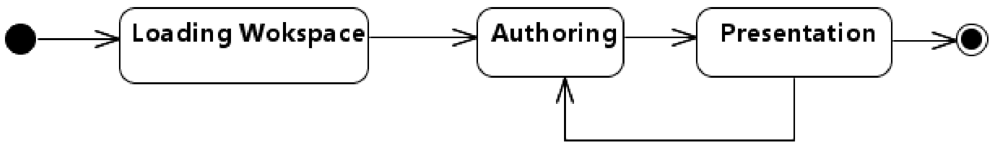


Figure 5.16: Metalaboratory State Machine Diagram

The first state occurs in the metalaboratory environment start-up. The second stage involves composing a virtual, asynchronous or hybrid laboratory. Finally, the last state can be triggered when the artifact is ready to be executed by the end user. In the following sections we present details about each stage.

Loading Workspace

If the laboratory to be built is a composition of components, the authoring tool (metalaboratory) is itself also a composition of components. Therefore, the first stage involves loading components, connecting them in a composition and running the result.

Figure 8 illustrates the metalaboratory workspace. It is formed by two main areas, a shelf of components and connections (left) and a composition area (right), which contains a laboratory template and a execution button. On the shelf (top), the user can select the components that will be inserted on the composition. It presents two comboboxes. In the first one, the user selects a DCC or meta-DCC to be inserted in the template. The second

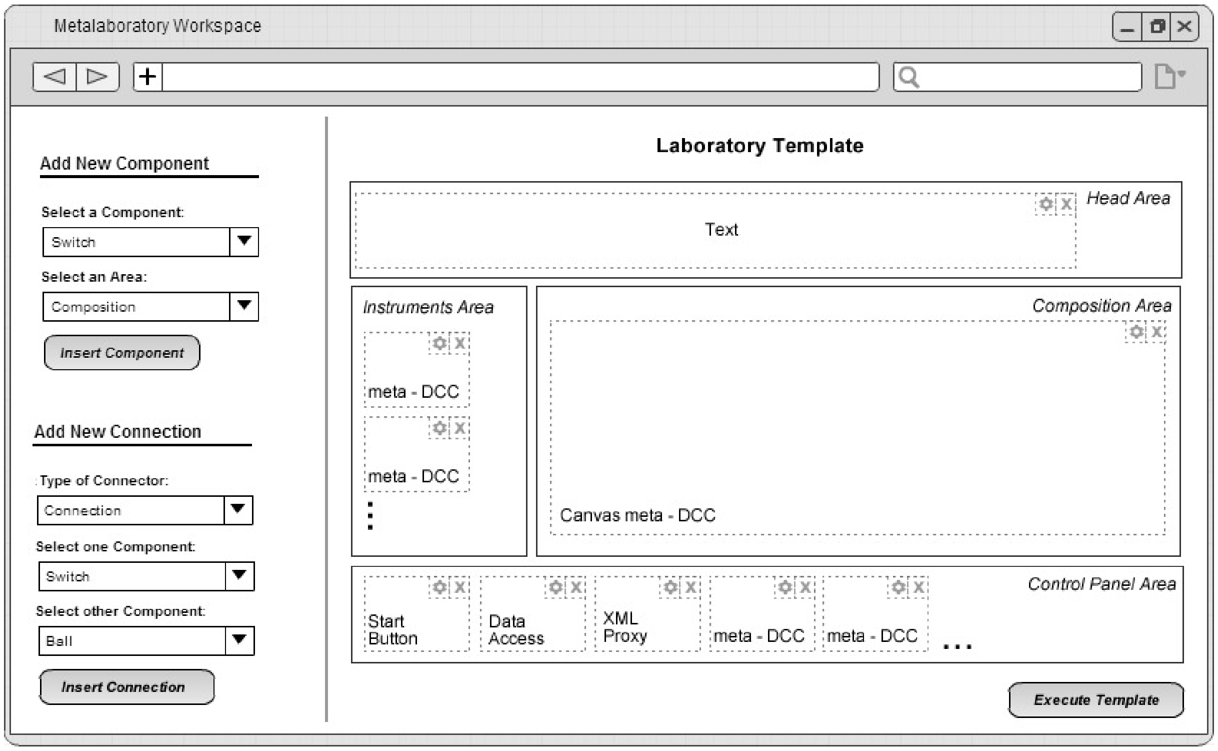


Figure 5.17: Metalaboratory Workspace Schema

combobox defines the template area to insert the DCC. The first combobox is filled by the components available in the component repository. The second is based on the areas of the template.

In the area below the shelf, the user can select the type of connection that will bind two components. It has three comboboxes. In the first combobox, the user selects the type of the connector. In the other two comboboxes, the user selects which components will be binded. The two types of connectors available for selection represent the approaches to connect DCCs: publish / subscribe and provided / required interfaces. The other two comboboxes show the components that were already inserted on the template. On the composition area, the user builds the active Web Template.

The workspace is formed by DCC compositions illustrated in Figures 5.18, 5.19, 5.20. They are specializations of environment – shelf, composition template –, visual support – comboboxes, buttons, meta-DCCs –, controller – XML proxy –, and model – component access.

Figure 5.18 shows the main components of the metalaboratory workspace, structured in MVC layers. The **shelf component** is responsible for managing the components and connections available in the **new components area** and the **new connections area**. The **composition area** is responsible for managing the composition process in

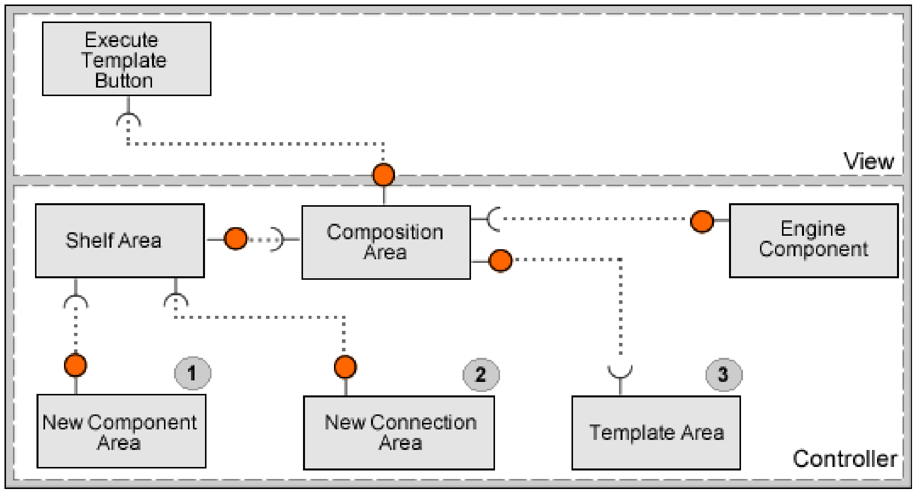


Figure 5.18: Metalaboratory Workspace Basic Components

conjuction with the **template area**, which handles templates over compositions. The **execute template button** triggers the process in which the composition is dispatched to the **engine component**, which generates the laboratory.

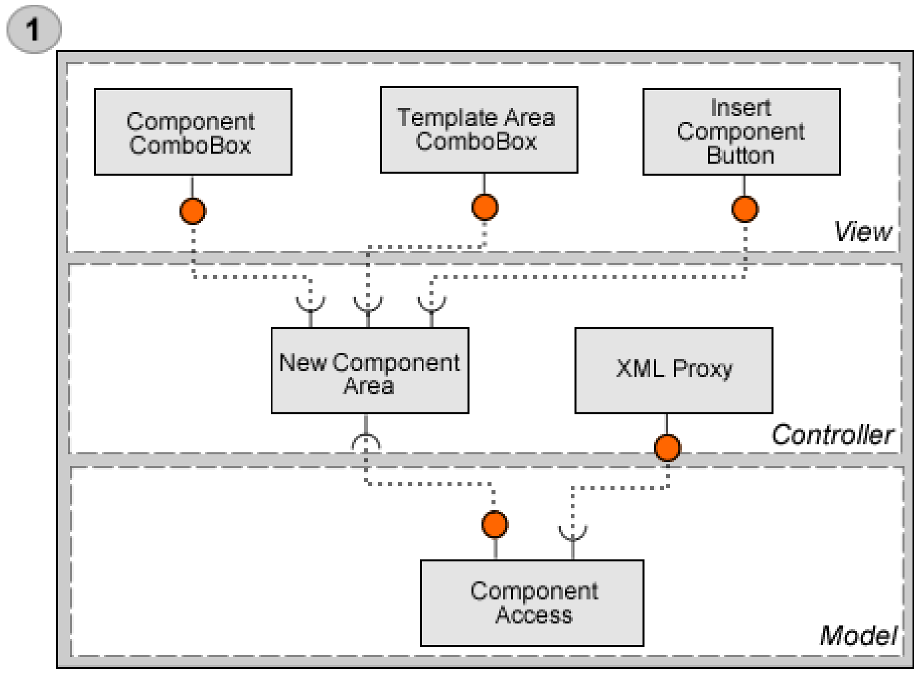


Figure 5.19: Metalaboratory Workspace Basic Components - Group 1

Figure 5.19 show the components that are related to the **new components area** component. The **new components area** is responsible for requesting to the **component**

access the available components and to the **template area** the areas available in the template. It delivers the result to the **component combobox** and **template area combobox**, respectively. The **component access** requests to the **XML proxy** data about the components available for composition, which are stored in a XML repository. The **component access** maps the available components to an internal representation and delivers them to the **new components area**.

The **insert component button** is responsible for requesting to the **new components area** the selected component and the target template area and for delivering this information to the **composition area**.

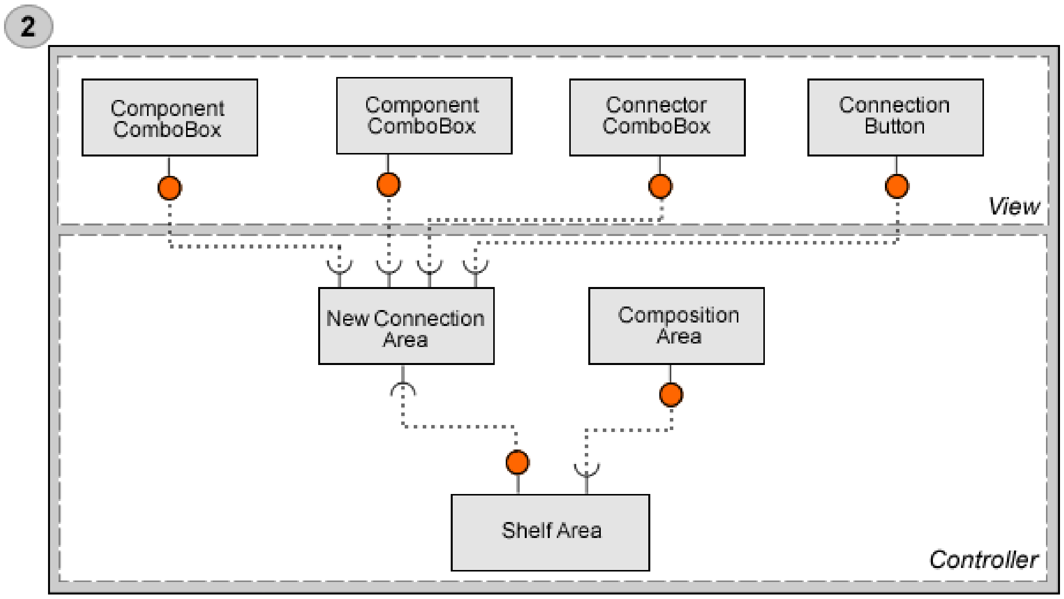


Figure 5.20: Metalaboratory Workspace Basic Components - Group 2

Figure 5.20 shows the components that are related to the **new connection area** component. It requests to the **composition area**, through the **shelf area**, the components instantiated and places them in both component comboboxes. The available connections are directly placed on the **connection combobox**.

The **connection button** is responsible for requesting to the **new connection area** the selected connection and components for delivering this information to the **composition area**.

The Authoring State.

The second state is the authoring. Here the author will transform personal knowledge into multimedia compositions. A process that allows the development of narrative structures based on different types of media [7].

The authoring state offers the infrastructure to build a laboratory based on the template methodology. This task involves composing laboratories by using components that are already created, stored on the component repository, and available on the shelf. Through meta-DCCs, authors are free to insert, delete and configure the available components and connections.

Figure 5.17 shows the workspace environment. In order to build a laboratory, the user selects, for instance, a mobile ball component. By clicking on the insert button component, the selected component can be inserted in an available meta-DCC of the template area. To do this, the **shelf area** component delivers an instance of the selected component of the **composition area**. The **composition area** delivers the instance to the defined **template area**.

Many authoring systems offers two possibilities of artifact creation: by direct manipulation or using an authoring languages. In the first case, the software offers a graphical interface where the media elements can be composed on a interactive way to produce a presentation. In the second case, the software offers a particular authoring scripting language to construct or enrich the composition. Adobe Flash with the Action Script language and Adobe Director with the Lingo are some examples. Our approach can be associated with the first case, but has elements of the second one, as it is possible to author compositions in HTML.

At this moment, the authoring stage do not allow the creation of new components through the graphical interface. As the metalaboratory runs over Componere, it is possible to construct and store components directly on the framework. When the user finishes the authoring process, the composition becomes the actual laboratory and the artifact will be ready to run as a presentation in the next state.

The Presentation State.

The presentation represents the final state of the metalaboratory execution. It occurs when the user ends the laboratory composition and starts its execution.

When the user clicks on the execute template button, an engine that converts the composed template into the laboratory is started. The flow of this execution is presented in the activity diagram of Figure 5.21. The engine is responsible for saving the template structure with the composition into a new HTML file.

As detailed in Figure 5.21, the engine starts by creating the HTML file, that will be the laboratory. After opening and enabling the file to be written, the engine will write in the file the template structure. Each area of the template example – head, instruments, composition, control panel – will be a `div` element.

The next step is to identify all the components that were parts of the authoring and their respective location in the template. The engine requests this information to the

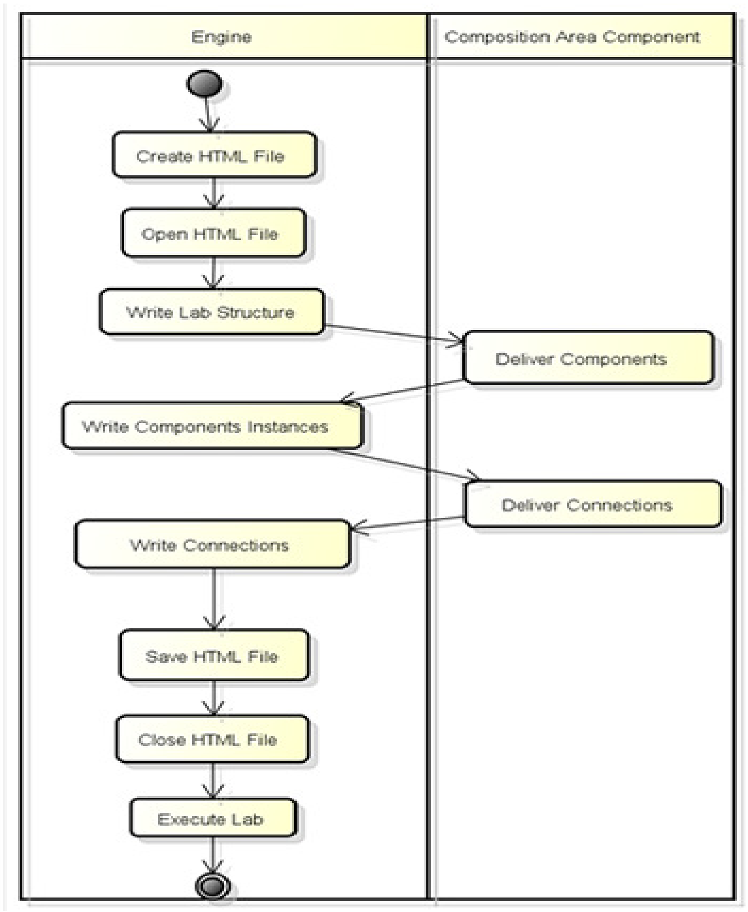


Figure 5.21: Presentation State Activity Diagram

meta-DCCs in the template, the components instances.

The engine will replace meta-DCCs by DCCs in the template, connecting them and removing the meta-DCCs. The engine saves the file in the server side and open it on a new tab of the browser as a laboratory, ready to be used. In this process, a Javascript engine converts the Microformats marks in Javascript code, which instantiates the DCCs and connects them. During the execution, only visual components will be visible and available for interaction.

During the execution, the components interacts in the composition by basically two strategies: service-oriented or actor-oriented. A service-oriented occur when a component has a call-response behaviour. Thus, it waits for requests, executes the respective tasks and sends back responses with the results. An actor-oriented component runs as an independent process, interacting with the environment by exchanging messages and when an event occur, the component send a message notifying the destination component.

The user can at any moment return to the metalaboratory workspace to change the template and to save a new version of the laboratory. This characteristic represents the experimental aspect of the construction of a laboratory, which makes the metalaboratory itself an environment of experimentation.

5.4 Conclusion and Future Work

We presented here our metalaboratory, an environment to build laboratories through authoring, with an experimental approach. In this environment, virtual, asynchronous and hybrid laboratories can be built by combining components through direct manipulation.

We have developed a metalaboratory environment based on a composition of software components. Metalaboratory components can encapsulate different types of content. The model is based on our proposal of metalaboratory family of components, where the developed components are grouped according to the role they play. That proposal aims to offer a classification and organization of components, facilitating the construction of laboratories.

The metalaboratory environment was developed over the Componere, an authoring environment. It adds to Componere three new features: the family of components, the lab composition patterns and the meta-DCCs for the construction of artifacts based on an active template method. The metalaboratory itself is based on components.

As far as we know, this is the first initiative to produce this kind of metalaboratory – a laboratory to experimentally build laboratories – enabling to combine asynchronous and virtual labs in the same environment.

To validate our proposal, we built laboratories in different domains. In [20] we present a virtual biology laboratory for taxonomic descriptions. In [17] we present an asynchronous physics lab to work on the motion of an object on an inclined plane. In [19] we present a hybrid physics lab to study the motion of damped pendulums. In this work, we expand the laboratory of [17] to a hybrid lab, to present how a laboratory is constructed in the metalaboratory environment.

Future work include the expansion of the authoring tool to provide enhanced support for template creation; the definition of new template operators to enable more flexible customization; the expansion of our laboratory component family; the possibility of creating new components inside the metalaboratory environment.

Chapter 6

Conclusion

There are many solutions to support the construction of scientific knowledge. We introduced here the notion of metalaboratory as a laboratory environment to produce laboratories, in an experimental way, using building blocks. As far as we know, none of related work offers the support that our metalaboratory proposes, plus the possibility of placing analysis and synthesis results in a same environment.

Our approach adopts the Web as the basis of the metalaboratory, encapsulating the rich Web content and services.

6.1 Contributions

The challenge of this work was to propose a model to create virtual, asynchronous, and hybrid experiments using a single environment that follows a laboratory approach. Our metalaboratory specializes the authoring process to support laboratory creation.

To validate the metalaboratory proposal, we introduced the asynchronous laboratory approach in a practical experiment. This was our first laboratory, used to test the architecture of accessing asynchronous data. It involved the execution of an experiment based on data captured by sensors during a real world experience, and stored for asynchronous access.

The second laboratory applied our model in a virtual lab for taxonomic description. It validated our model in a different context, the Biology. The entire laboratory was developed with components. In this case, we introduced the model of description components.

From the development of the asynchronous and virtual laboratories, we proposed the concepts: family of components, composition patterns and active Web templates. Those ideas were validated in a third laboratory, a hybrid lab on the Physics context, which placed side-by-side two equivalent setups that are based in different sources of asynchronous and virtual data.

6.2 Extensions

The metalaboratory model was based on our experience with the construction of virtual and asynchronous laboratories, inserted in the Biology and Physics domains, respectively. The construction of laboratories in other domains may indicate new classes of components as members of the components family.

Based on the developed laboratories, we identified the existence of a pattern for the construction of virtual laboratories and another for asynchronous laboratories. The construction of laboratories in other domains can also lead to the identification of other types of composition patterns, improving the proposal of this work.

We implemented a metalaboratory authoring tool prototype. This prototype can be enhanced to provide better graphical interface to create active Web templates.

Bibliography

[1]

[2] M. Abdulwahed and Z.K. Nagy. Developing the trilab, a triple access mode (hands-on, virtual, remote) laboratory, of a process control rig using labview and joomla. *Computer Applications in Engineering Education*, 2010.

[3] J.P. Agrawalkand and Y.E. Cherner. A classroom/distance learning engineering course on optical networking with virtual lab. In *Computational Technologies in Electrical and Electronics Engineering, 2008. SIBIRCON 2008. IEEE Region 8 International Conference on*, pages 73–77. IEEE, 2008.

[4] F. Barrero, S. Toral S., and Gallardo. edsplab: remote laboratory for experiments on dsp applications. *Internet Research*, 18(1):79–92, 2008.

[5] Gordon Bell, Tony Hey, and Alex Szalay. Beyond the data deluge. *Science*, 323(5919):1297–1298, 2009.

[6] F. Bronsard, D. Bryan, W. Kozaczynski, E.S. Liongosari, J.Q. Ning, Á. Ólafsson, and J.W. Wetterstrand. Toward software plug-and-play. In *ACM SIGSOFT Software Engineering Notes*, volume 22, pages 19–29. ACM, 1997.

[7] D.C.A. Bulterman and L. Hardman. Structured multimedia authoring. *ACM Transactions on Multimedia Computing, Communications, and Applications (TOMCCAP)*, 1(1):89–109, 2005.

[8] S. Burbeck. How to use model-view-controller (mvc). *ParcPlace Systems Inc., see http://stww.cs.uiuc.edu/users/smarch/st-docs/mvc.html*, 1992.

[9] Eduardo Cerqueira, Ronei Poppi, and Lauro Kubota. Utilização de filtro de transformada de fourier para a minimização de ruídos em sinais analíticos. *Workshop de Teses e Dissertações - CBComp*, 23(5), 1999.

[10] P. G. Cramer and G. De Meyer. The philosophy of the virtual laboratory, 1997.

[11] D. Morrison D. and Brian. Metalab: supporting social grounding and group task management in cscl environments through social translucence. In *Diversity in Computing Conference, 2005 Richard Tapia Celebration of*, pages 20–22. IEEE, 2005.

[12] O.G. Bellmunt D.M., Miracle S.G., Arellano A., Sumper A.S., and Andreu. A distance plc programming course employing a remote laboratory based on a flexible manufacturing cell. *IEEE Transactions on Education*, 49:278 – 284, May 2006.

[13] M. Duarte and B. P. Butz. An intelligent universal virtual laboratory (uvl). *IEEE Transactions on Education*, 51(1):2 – 9, 2008.

[14] K. M. Breakey et al. Genetics education. *Genetics*, 179:1151 – 1155, 2008.

[15] M. Koretsky et al. Enhancement of student learning in experimental design using a virtual laboratory. *IEEE Transactions on Education*, 51:76–85, 2008.

[16] Athula Ginige, David B. Lowe, and John Robertson. Hypermedia authoring. *IEEE Multimedia*, 2(4):24–35, 1995.

[17] Alessandra Gomes and André Santanchè. Autoria virtual baseada em dados do mundo real. *X Workshop of Tools and Applications (WebMedia)*, 2011.

[18] Alessandra Gomes and André Santanchè. Metalaboratory technical aspects. *To be submitted to Institute of Computing – UNICAMP (Technical Report)*, 2013.

[19] Alessandra Gomes and André Santanchè. Web metalaboratory: Composition of laboratories on the web. *Submitted to the International Workshop on Lightweight Integration on the Web (ComposableWeb)*, 2013.

[20] Alessandra Gomes, André Santanchè, and Fabiani Souza. Web-based virtual lab for taxonomic description. *XI Workshop of Tools and Applications (WebMedia)*, 2012.

[21] V. Hoyer, K. Stanoesvka-Slabeva, T. Janner, and C. Schroth. Enterprise mashups: Design principles towards the long tail of user needs. In *IEEE International Conference on Services Computing*, volume 2, pages 601–602. IEEE, 2008.

[22] R. Khare. Microformats: The Next (Small) Thing on the Semantic Web? *IEEE Internet Computing*, 10(1):68–75, 2006.

[23] A. Laganà, A. Riganelli O., Gervasi P., Yates K., Wahala R., Salzer E., Varella, and J. Froehlich. Elchem: a metalaboratory to develop grid e-learning technologies and services for chemistry. *Computational Science and Its Applications–ICCSA 2005*, pages 69–102, 2005.

[24] K.K. Lau and Z. Wang. Software component models. *Software Engineering, IEEE Transactions on*, 33(10):709–724, 2007.

[25] J. Ma and J. V. Nickerson. Hands-on, simulated, and remote laboratories: A comparative literature review. *ACM Comput. Survey*, 38(3):1–24, 2006.

[26] Paula Mabee, Michael Ashburner, Quentin Cronk, Georgios V. Gkoutos, Melissa Haendel, Erik Segerdell, Chris Mungall, and Monte Westerfield. Phenotype ontologies: the bridge between genomics and evolution. *Trends in Ecology and Evolution*, 22(7):345 – 350, 2007.

[27] Duane Merrill. Mashups: The new breed of web app. http://www.ibm.com/developerworks/web/library/x-mashups/index.html, 2009.

[28] Jan Mikáč, Cécile Roisin, and Bao Le Duc. An export architecture for a multimedia authoring environment. In *ACM Symposium on Document Engineering*, pages 16–19. Citeseer, 2008.

[29] Modellus. Modellus – interactive modelling with mathematics. http://modellus.fct.unl.pt/, May 2012.

[30] Zorica Nedic, Jan Machotkd, and Andrew Najhlsk. Remote laboratories versus virtual and real laboratories. *Frontiers in Education, 2003. FIE 2003. 33rd Annual*, 1:1 – 6, 2003.

[31] S. Olabarriaga, T. Glatard, and P.T de Boer. A virtual laboratory for medical image analysis. *IEEE Transactions on Information Technology in Biomedicine (TITB)*, 14(4):979–985, 2012.

[32] Arduino Project. What is arduino? http://arduino.cc/en/Guide/Introduction, 2011.

[33] Fernando Ranieri. Sistema supervisório de parâmetros de máquinas elétricas via tcp/ip e painel eletrônico de mensagens. Dissertacao de Mestrado. Universidade Federal de Sao Carlos., 2007.

[34] X. Cao S. and Zhu. ieelab practice: A hybrid remote laboratory for distance education in electrical engineering. In *Computer Science and Education (ICCSE), 2010 5th International Conference on*, pages 592–596. IEEE, 2010.

[35] André Santanchè and Peter Baumann. Component-based web clients for scientific data exploration using the dcc framework. In *Procs. of Int'l Conf. on Geographic Information Science*, pages 1–5.

[36] Andre Santanche, Matheus Mota, Diego Costa, Nicolas Oliveira, and Christianne O. Dalforno. Componere autoria na web baseada em componentes. *WebMedia*, pages 91–98, 2009.

[37] Interage Simulation. Interage simulation. http://www.cienciamao.if.usp.br/, May 2012.

[38] Lizet Suaréz and Ricardo Gudwin. Análise do conhecimento sensorial segundo a perspectiva da semiótica computacional. *Workshop de Teses e Dissertações - CBComp*, pages 91–98, 2002.

[39] C. Szyperski, D. Gruntz, and S. Murer. *Component software: beyond object-oriented programming*. Addison-Wesley, 2002.

[40] C. Szyperski, D. Gruntz, and S. Murer. *SCADA: Supervisory Control and Data Acquisition, ISA-The Instrumentation, Systems, and Automation Society*. Addison-Wesley, 2002.

[41] Anne van Kesteren. W3c candidate recommendation. http://www.w3.org/TR/XMLHttpRequest/, 2011.

[42] F. Wild, F. Moedritscher, and S. Sigurdarson. Designing for change: mash-up personal learning environments. *e-Learning Papers*, 9, 2008.

[43] A.C. et al. Wills. Objects, components and frameworks with uml: The catalysis approach. 1999.

[44] J. Yu, B. Benatallah, F. Casati, and F. Daniel. Understanding mashup development. *Internet Computing, IEEE*, 12(5):44–52, 2008.

Appendix A

Description of Metalaboratory Components Classes

This appendix presents a description of each class of the metalaboratory model and in which practical case it was implemented.

Table A.1: Description of Metalaboratory Component Classes

Class	Definition	Implementation
`Action`	Represents an action that a component can have	All the laboratories
`Brush`	`Visual Instrument` that changes the color of a selected component.	Not implemented
`Button`	`Visual Support` that triggers an event it is clicked	Taxonomic Laboratory
`Candidates Presentation`	Specialization of `Environment` that manages the presentation of the identification process – lizards in our example.	Taxonomic Laboratory
`ComboBox`	`Visual Support` widget that stores a list and trigger an event when an item is selected	Metalaboratory Prototype
`Component Descriptor`	`Visual Real World Object` that stores the descriptions of each living being – lizards in the example.	Taxonomic Laboratory

`Composition Area`	`Environment` that manages the description of the living beings component parts – lizards in our example.	Taxonomic Laboratory and Metalaboratory Prototype
`Control Area`	`Environment` that manages components, which controls the execution of a laboratory.	Metalaboratory Prototype
`Controller`	Responsible for controlling tasks and bridging `View` and `Model` classes	All laboratories
`Data Access`	`Controller` that accesses and delivers data from `Model`.	All laboratories
`Description Space`	`Environment` that manages the description of the living beings parts – lizards in our example.	Taxonomic Laboratory
`Environment`	`Support` that is responsible for managing a virtual space including all the components that are inserted in its area	All laboratories
`File Proxy`	`Data access` that accesses a file content.	All laboratories
`Formula`	`Model` that calculates a predefined formula	Damped Pendulum Laboratory and Mobile Ball Confront Laboratoty
`Image`	`Visual Support` that stores an image and can trigger an event when it is clicked	All laboratories
`Instrument`	Visual component that enables measurements of components during the execution	Metalaboratory Prototype
`Lamp`	`Visual Real World Object` that simulates a lamp changing from ON to OFF when it is clicked	Metalaboratory Prototype
`Magnifier`	`Visual Instrument` that applies zoom in and out in a visual component when it is clicked	Not implemented
`Mobile Ball`	`Visual Real World Object` that illustrates a ball moving according to some role.	Mobile Ball Laboratory and Mobile Ball Hybrid Laboratory.
`Model`	Represents data according to a model	All laboratories

`Movement Reproducer`	`Action` that reproduces the motion of an object.	Mobile Ball Laboratory, Mobile Ball Hybrid Laboratory and Damped Pendulum Laboratory
`Pendulum`	`Visual Real World Object` that simulates a pendulum object.	Damped Pendulum Laboratory
`Physical Environment`	`Action` that manages the motion of an object.	Mobile Ball Laboratory, Mobile Ball Hybrid Laboratory and Damped Pendulum Laboratory
`Protractor`	`Visual Instrument` that shows the angle of an object.	Not implemented
`Real World Object`	Responsible for simulating or reproducing the behaviour of an object of the real world.	All laboratories
`Role`	Represents the role that will be applied to an `Action`.	All laboratories
`Ruler`	`Visual Instrument` that measures the size of an object	Not implemented
`Search Engine`	`Action` that searches for living beings candidates during the identification process – lizards in our example	Taxonomic Laboratory
`Shelf`	`Environment` responsible for storing, selecting and delivering components to another environment component.	Taxonomic Laboratory and Metalaboratory Prototype
`Support`	Responsible for giving assistance by controlling or managing components.	All laboratories
`Switch`	`Visual Real World Object` that changes its state from ON to OFF when it is clicked, producing an event.	Mobile Ball Laboratory, Mobile Ball Hybrid Laboratory, Damped Pendulum Laboratory
`Table`	`Visual Support` that stores and shows data in rows and cols.	Taxonomic Laboratory
`Text`	`Visual Support` that stores and shows text information	All laboratories

`Timer`	`Visual Support` that generate events in a fixed frequency rate.	Mobile Ball Laboratory, Mobile Ball Confront Laboratory, Damped Pendulum Laboratory
`View`	Responsible for managing the user input and the output through a visual component	All laboratories
`Visual Instrument`	Represents components that have visual characteristics and plays instrument roles.	Not implemented
`Visual Real World Object`	Represents components that have visual characteristics and plays real world objects roles.	All laboratories
`Visual Support`	Represents components that have visual characteristics and play support roles.	All laboratories
`Web Proxy`	`Data Access` that accesses a Web file content.	Taxonomic Laboratory, Mobile Ball Laboratory