A Visual Programming Language for Designing Interactions Embedded in Web-based Geographic Applications

The Nhan Luong, Patrick Etcheverry, Christophe Marquesuzaà, Thierry Nodenot

A Visual Programming Language for Designing Interactions Embedded in Web-based Geographic Applications

The Nhan Luong, Patrick Etcheverry, Christophe Marquesuzaà and Thierry Nodenot
T2i - LIUPPA - Université de Pau et des Pays de l'Adour
2 Allée du Parc Montaury, 64600 Anglet, France
{thenhan.luong, patrick.etcheverry, christophe.marquesuzaa, thierry.nodenot}@iutbayonne.univ-pau.fr

ABSTRACT
Visual programming languages (VPLs) provide notations for representing both the intermediate and the final results of a knowledge engineering process. Whereas some VPLs particularly focus on control flow and/or data flow of a software, very few VPLs stress on the interactive dimension of application (dialogue flow). This paper focuses on a VPL allowing designers to specify interactions between a user and a system, in the field of Web-based geographic applications. We first present the underlying interaction model that the VPL is based on, and then the detailed characteristics of the VPL. We show how this VPL has been integrated in a graphical design framework allowing designers to immediately assess their specification. Then we illustrate the way to use the framework from the design step to the final code generation step. Last, we detail an experimentation aiming at evaluating the strengths and the weaknesses of our VPL.

Author Keywords
Visual design language, interaction design, geographic application design, visual authoring tools.

ACM Classification Keywords
H.5.m. Information Interfaces and Presentation (e.g. HCI): Miscellaneous

General Terms
Design, Languages.

INTRODUCTION
Visual programming languages (VPLs) aim at facilitating software design and implementation by minimizing or avoiding complex coding steps. The visual nature of these languages is characterized by the use of graphic elements that can be combined and connected within a two (or more) dimensional design workspace. The expressive power of these languages explain their attractiveness for simplifying the description of complex things. This richness can also be a constraint when it comes to interpreting the meaning of diagrams in order to automatically generate executable code.

Our research consists in promoting VPLs to empower designers developing interactive applications. Depending on the domain, such VPL-based prototyping tools are more or less difficult to produce. Our challenge is to focus on an ill-structured domain that addresses (on a medium-term) the design of applications promoting "Engaged reading interactive scenarios that make use of geographic information". We chose such a domain for three main reasons. Firstly, geographic information promoting engaged reading can be quite informal, but its semantics, once captured by automatic or semi-automatic analysers, can be exploited to promote rich interactive scenarios. Secondly, there is a real need for some VPLs to design applications for touristic, cultural or educational purposes. Lastly, deploying such interactive applications from a VPL is a challenging task because the required VPL must stress on the interactive dimension of the application (dialogue flow) and not only on the control flow and/or the data flow.

In this paper we present this VPL that does not address computer-science companies with structured teams of developers but people (with some computer-science background) from firms or organisations who are interested in deploying interactive applications without having to manage all the complexity of the technology. This visual language only focuses on human computer interactions and proposes specific descriptor elements allowing designers to rapidly specify and assess what happens (from a system reaction viewpoint) when a user carries out an action.

The paper is structured as follows: Section 2 ("Related Work") defines more precisely the concept of VPL and positions our contribution according to similar and related work. Section 3 ("Geographic Application Modelling") describes the underlying model on which the VPL is based on. This three-part model allows designers to define the contents that will be handled, how they will be displayed on the screen and the possible interactions allowed within these contents. Section 4 ("Visual Specification of Interactions") presents the main contribution of the paper: we describe the characteristics of the VPL that will allow designers to specify interactions according to the model described in Section 3. Section 5 ("Application Design with the WINDMash Prototype") illustrates the operational nature of the proposed VPL through a graphical design environment called WINDMash. This design environment integrates our VPL and allows designers to visually specify Web-based geographic applications from an interactive viewpoint and then to generate the corresponding executable code. Section 6 ("Evaluation") presents a first experimentation aiming at evaluating the main strengths and weaknesses of our VPL. Finally, we conclude this paper by summing up the results and presenting our plans for future work.

RELATED WORK

This paper intends to offer a VPL allowing designers to graphically describe human computer interactions and to generate the corresponding Web-based application. The contribution focus on a VPL adapted to the description of interactions. Focusing on interaction is particularly interesting because it is generally the most complex thing to describe, to specify and to implement in a program. In this paper, the underlying code generation process is not considered as a scientific contribution but as an engineering work based on existing model transformation techniques (MDA).

In next sections, we use visual programming to mean the construction of a program starting from a graphical representation of its behaviour. We define a VPL in the same way as [15, 19, 20, 22]: any graphical language that lets programmers create programs by handling program elements represented graphically rather than textually. According to the classification described in [3] (which compiles the viewpoint of authors such as [6, 7, 22]), we offer a hybrid text and visual language: programs are visually created and then translated into an underlying high-level textual language, which is then translated into executable source code.

Integrating VPLs in Geographic Information Systems (GIS) is not a new idea: ArcGIS, Mapinfo, AutoCAD Map3D are sophisticated GIS restricted to specialists. VPLs try to make these systems available to standard users. As shown in [9], the two recurring problems addressed by VPLs in such systems relate to the creation/selection/aggregation of data and to the manner of representing this data on a map or textually. To our knowledge, there is no VPL allowing designers to define user interactions with displayed data, except the default interactions provided by the displayer (e.g. changing the background of the map that displays the data). Moreover, default interactions provided by displayers cannot (generally) be set up or disabled by designers.

Similar problems can be highlighted with Mashup systems such as Google Mashup or IBM mashup center which allow designers to graphically create/aggregate data (from feeds for example) and to display it in a specific way (on a map with Google Mashup, using widgets with IBM mashup center). However, these tools do not allow designers to create new interactions with displayed data: interactive possibilities are predefined by displayers (sorting data in a spreadsheet widget for example) and designers have little control over these predefined interactions.

The underlying programming approach is also very important. We can identify two main approaches [9, 25] that use VPLs in GIS: The traditional programming approach and the programming by demonstration approach. The first approach deals with using a visual language to describe what the program must do (*cf.* ModelBuilder in ArcGIS or the Workflow Designer in AutoCAD Map3D). The resulting description is then compiled or interpreted and run. The second approach consists in creating a program by showing an existing system how to carry out a specific task: The system memorizes each step of the process as a new program. Each step is graphically represented to allow designers checking the features of the program they are elaborating (see for example the C-SPRL system in [25]).

In these two approaches, VPLs are used to specify which data must be handled and what must be done with this data. There is also a continuous feedback which allows designers to control/correct the program they are elaborating. However, designers activity is quite different: In a traditional programming approach, designers carry out a specification/formalization activity of the program to elaborate whereas in a programming by demonstration approach, designers carry out a "tutor" activity by showing an existing system how to execute a specific task. The VPL we propose aims at elaborating geographic application from scratch and so, designers will use our language to design their program in a traditional way.

The interaction programming language we propose is based on UML which provides several models to describe interactions in an application. As shown in [1, 14, 26], UML can be used as a visual programming language to generate Java code starting from classes and statechart diagrams. Generally, the interactive aspect of the application is described using statechart diagrams that specify the reactions of a system according to the actions of a user. Statechart diagrams allow designers to describe the interactive dimension of an application globally but this global description can be complex if the application includes many interactive possibilities.

The sequence diagram-like VPL we propose allows designers to decompose the global interaction of a system into several separate interactions. The design is facilitated because each diagram describes only one interaction. The global view and the coherence of the interactive possibilities of the system can be reconstituted by merging the set of sequence diagrams to produce a state chart diagram as presented in [14] and in [26].

GEOGRAPHIC APPLICATION MODELLING

We offer to guide the design process considering the contents presented to users and the interactive possibilities given to them. With such an approach, designers' work involves defining the contents to be manipulated, how they will be displayed and the interactions handling these contents.

We offer design tools allowing designers to focus on the application's interactive dimension. To achieve this goal, we rely on operational models to be elaborated and integrated in design environments. These environments should assist the designer from the specification step to the deployment step.

This section defines the main key concepts that we offer to describe the interactions within an application. Figure 1 presents an overview of these concepts. The model may be divided into three parts (*Content*, *Interface* and *Interaction*) articulated around the concept of geographic contents (*geocontents*) which represents the central concept of our model because our design approach mainly focuses on emphasizing geographic data [16].

In the content part, geocontents are defined by a set of structured information; each one has a type (absolute or relative spatial entity), a value (e.g. "*Mauléon-Licharre*") and one or

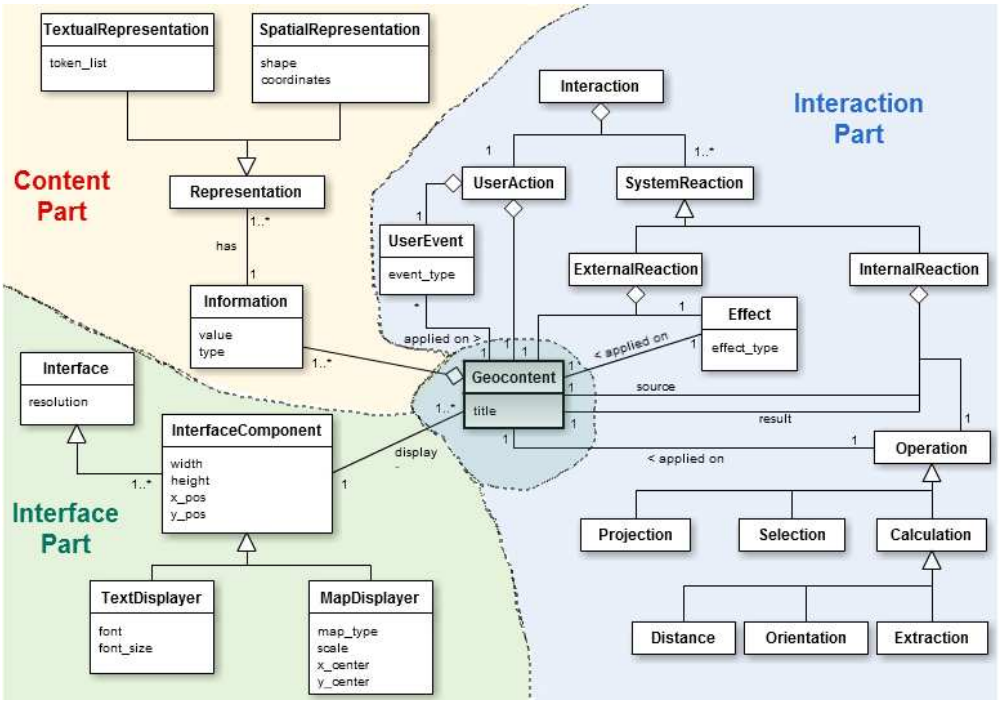

Figure 1. Global model for describing geographic applications

more possible representations of this value (e.g. 9th token of the first paragraph in the text or *POINT(2, 45)* in the map). Each considered representation allows a geocontent specific dimension (textual or spatial) to be emphasized.

The application interface is a visualization layer allowing geocontents to be displayed in various forms [8].

In our approach, design is guided by interactions to emphasize contents specified by the designer. As proposed by [10, 24], the vocabulary used for designing interactions is based on user action and system reaction. An interaction is defined as a communication between a user and the system. This communication is always initiated by the user and ends when the application has visually reacted to the user request. An interaction is implemented by a user action triggering a system (external or internal) reaction.

To automatically generate a Web-based application according to this model, we have implemented a design environment. Each model's part is instanciated into an RDF file [17] which lists the characteristics of geographic contents / interface / interactions. Merging these RDF descriptions constitutes a structured specification which is used to automatically generate the final application thanks to MDA techniques.

We will now define how to instantiate the interaction model part with a visual specification language allowing designers to specify interactions according to previously defined concepts.

VISUAL SPECIFICATION OF INTERACTIONS

To use the interaction model during the design activity, we offer a visual language [4, 23] allowing designers to describe the characteristics of the interactions to set up. For an interaction, it is necessary to specify: the user action triggering the interaction, the system reactions, and the geocontents which are visually modified by the interaction.

As mentioned in Related Work, we offer a visual language inspired from UML sequence diagrams. Sequence diagrams are often used as a graphic language to describe interactions between a user and a system and also between system components [12, 5]. [13] proposed specific uses of these diagrams to describe interactions according to graphic components displayed on the interface. We also agree with this approach since our objective is to describe interactions according to what the user sees on the screen (interface components) but also contents displayed by these components.

[11] proposed an approach dedicated to interaction design based on the nature of the contents to emphasize. This approach was already based on the traditional UML sequence diagrams. We currently offer a language derived from the sequence diagrams which is both simplified and partially specialized for the description of interactions on geographic contents. It is not an extension of traditional sequence diagrams but only a resumption of their graphic formalism in order to describe interactions according to our interaction model.

In the following subsections we present the interaction elements of the visual language that we offer to specify interactions. Each interaction is described by a diagram specifying the user action initiating the interaction as well as the system reactions. Geocontents involved in the interaction are represented on these diagrams and are defined according to the model presented in the previous section or calculated during the interaction via internal reactions.

Specification of a user action

A user action is initiated by a specific event (e.g. *click*, *mouse-over*, ...). This event is applied on contents which are displayed on an interface component. As an interaction is triggered by a user event, a user action is represented by an arrow going from the user toward the system and labelled with the name of this triggering event (Figure 2).

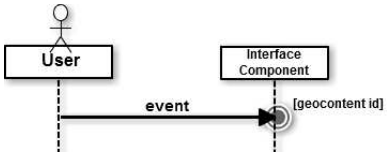

Figure 2. Specification of a user action

The arrow destination refers to the contents involved in the interaction and the interface component used to display them. The contents have an automatically generated identifier that can be renamed by the designer. This identifier allows the designer to handle the same contents into several interactions.

Specification of an external system reaction

This reaction is always visible by the user. It results in a visual modification of contents displayed on the interface. This modification is carried out by the system by applying one or some visual effects on the contents to emphasize.

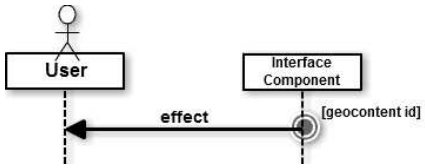

Figure 3. Specification of an external system reaction

Since an external reaction is applied on contents and can be seen by the user, it is represented by an arrow going from the contents to emphasize toward the user (Figure 3). The arrow label specifies the effect applied to emphasize these contents.

Specification of an internal system reaction

We consider three distinct internal reactions: *projection*, *selection* and *calculation*.

Projection

The projection operation involves transferring contents on a given interface component. Thereafter, the transferred contents can either be displayed on the target interface component or be used to calculate new contents on this interface component. The arrow origin (Figure 4) specifies the contents that must be projected and the destination defines the interface component where the contents is projected.

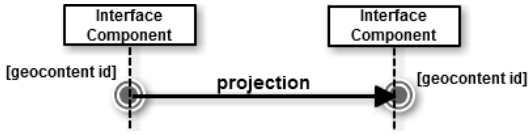

Figure 4. Specification of a projection operation

Selection

A selection is an operation allowing the designer to specify contents by selecting a subset of displayed contents. This subset becomes new contents that can be emphasized in the continuation of the interaction.

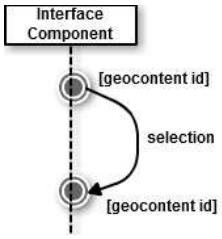

Figure 5. Specification of a selection operation

The selection operation is graphically represented by an arrow going from the initial contents toward the subset contents designated by the user (Figure 5). The selected contents belong to the interface component displaying the initial contents. Thenceforth, these new created contents can be displayed via another external reaction or can be transferred on another interface component with a projection operation.

Calculation

This operation allows designers to create new contents by applying a calculation operation on specific contents. As the contents are in a geographic nature, the authorized operations are also in a geographic nature: distance calculation, orientation, surface, etc. A calculation operation is represented by an arrow labelled with the calculation service to apply. The arrow connects the input and output contents (Figure 6).

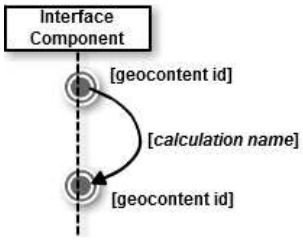

Figure 6. Specification of a calculation operation

The calculation parameters must be defined during the design activity. The calculation operations must also be pertinent with the input contents. These consistency checks can only be carried out if the design activity is supported by an adapted software environment.

APPLICATION DESIGN WITH WINDMASH

In order to illustrate our approach, we use the WINDMash[1] prototype allowing a designer to elaborate in a visual way Web-based geographic applications.

Starting from a text and a map presenting some towns located on the French Atlantic and Mediterranean coasts, the main goal of the application to design is to learn the concept of

[1] `http://erozate.iutbayonne.univ-pau.fr/Nhan/windmash4/`

"*département*"[2]. The behaviour of the application (Figure 7) is as follows: When the user clicks on a town written on the text (located on the left part of the screen) or displayed on the map zone (located on the right part of the screen), then the name of the corresponding "*département*" is automatically displayed (in a dedicated zone in the top centre of the screen) and the border of the "*département*" are highlighted (in a dedicated zone in the bottom centre of the screen). The name of the clicked town in the main text and the border of this town in the main map will also be highlighted.

Figure 7. Screenshot of the final application

A designer may use three complementary workspaces on the WINDMash environment. These workspaces respectively deal with defining the geocontents to be handled (*Data* workspace), organizing the presentation layout of the application (*Interface* workspace) and specifying how end-users will be able to interact with the application (*Interaction* workspace). Each workspace corresponds to the instanciation of its specific part of the global model presented in Figure 1. Each workspace allows designers to specify a particular viewpoint of the application, and each specification leads to the instanciation of a specific part of the global model.

Defining geocontents of the application

In order to build the geocontents that will be handled in the final application, designers use the *Data* workspace (Figure 8) which provides (on the left side) a set of services allowing to automatically extract geocontents from a text or a geographic database.

We have used the following text to illustrate the example described above: "*France has a long ocean coastline which is made up of a combination of cliff areas, rocky areas and sandy beaches. The south-west coast is washed by the Atlantic ocean and offers mile upon mile of unbroken sandy beaches from Arcachon southwards the area of Biarritz. It all changes as one reaches the Basque country, where the seaside is quite built up from Capbreton to the Spanish border. The south-east coast of France borders the warm Mediterranean sea. The seas are generally calm along the coast and the waters warm and very salty with many famous of French beaches from Perpignan to Nice.*"

Figure 8 presents a simple processing chain which corresponds to the presented example. The design of such processing chain is based on drag and drop. Starting from the

[2]A "*département*" is an administrative and geographic French concept corresponding to a region. Each place, town, city belongs to one (and only one) of 95 "*département*" (except overseas territories).

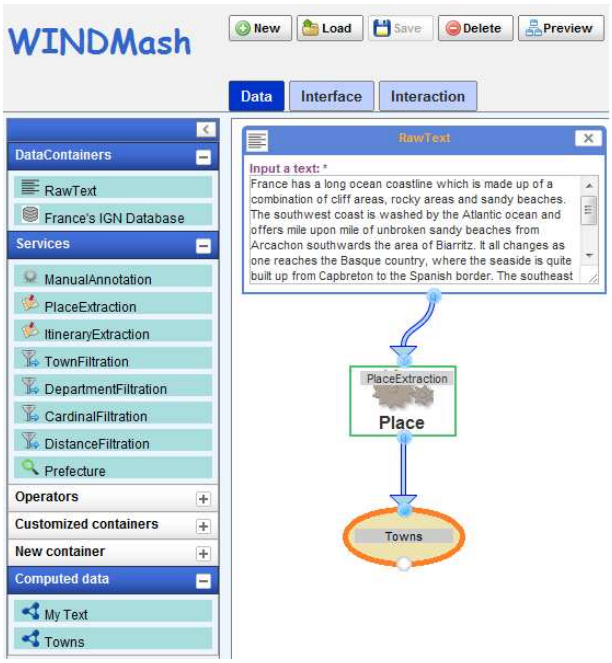

Figure 8. *Geocontents* designed with WINDMash

`RawText` presented above, the designer can use a service named `PlaceExtraction` which automatically extracts all places quoted in the text. These extracted places become a new set of geocontents (named here "`Towns`") which are added in the geocontent library of WINDMash.

Organizing presentation layout of the application

The *Interface* workspace of WINDMash enables designers to organize the presentation layout of their Web-based geographic application. WINDMash supports many ways to represent geographic contents such as `TextDisplayer`, `MapDisplayer`, `ListDisplayer`, `CalendarDisplayer`, `TimelineDisplayer` and `PhotoDisplayer`. Indeed, in this phase, designers have to decide how previously defined geocontents will be displayed on the screen. This choice is done by selecting specific displayers that will define the appearance of the geocontents.

In our example, designers have to create a graphic interface composed of a main left `TextDisplayer` showing the text and a main right `MapDisplayer` showing spatial information of the towns quoted in this text. Final application also needs a bottom centre secondary `MapDisplayer` showing the department border of the selected town and a top centre secondary `TextDisplayer` showing the name of the corresponding department. For more reader-friendliness, designers decide also to highlight all towns quoted in the primary text.

Figure 9 illustrates how designers can build the corresponding graphical interface of the final application. Designers have to drag and drop two `TextDisplayers` and two `MapDisplayers` onto the central workspace. Designers can easily

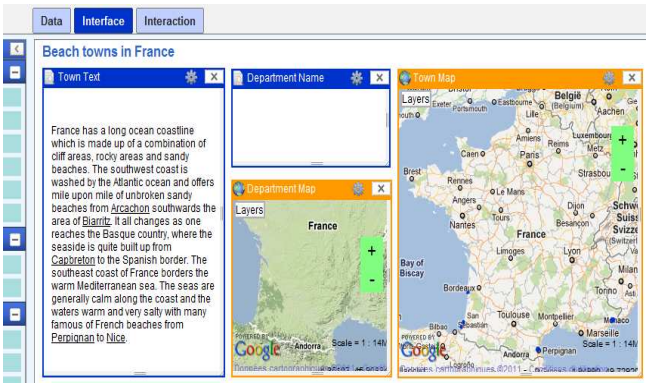

Figure 9. *Interface* designed with WINDMash

position and resize these displayers in order to build the final interface of the application. They can set up the name of the primary text displayer "`Town Text`", the name of the primary map displayer "`Town Map`", the name of the secondary text displayer "`Department Name`" and the name of the secondary map displayer "`Department Map`".

Then, designers select the set of geocontents that must be initially appeared in each one of these displayers. From the geocontents library in the left menu of WINDMash, they drag the set of annotations named "`Towns`" and drop it into the displayer "`Town Text`" as well as the displayer "`Town Map`". Two displayers named "`Department Name`" and "`Department Map`" do not hold geocontents yet, they will show computed geocontents when interactions are triggered.

Thus, the *Interface* phase involves positioning displayers into the interface and then dragging and dropping a set of geocontents into each displayer. Next subsection will show how to make geocontents interactive in the *Interaction* phase.

Specifying user interactions of the application

WINDMash allows designers to specify the interactions of their application using the VPL presented in the previous main section. In our example (Figure 10):

When the user selects a town in the text displayer named "`Town Text`" (A, B), this town (C) is highlighted in this displayer (D) as well as in the map displayer named "`Town Map`" (E, F). In addition, the text displayer named "`Department Name`" will show the name of the department of the selected town (G, H, I, J) and the map displayer named "`Department Map`" will zoom in on the department (G, H, K, L).

The second interaction triggered from towns selected in the map can be designed in the same way.

The diagram construction is carried out rather simply: when a designer drags and drops created displayers into the *Interaction* workspace, they become lifelines. Then, the designer has to drag the necessary interaction elements (*user action*, *selection*, *projection*, *calculation* or external *system reaction*) and drop them on the adequate lifeline.

The five interaction elements defined for our VPL allow designers to build complex interactions as we see in Figure 10.

A demonstration video of the complete design process is available at: **`http://youtu.be/3uxR8euHPwM?hd=1`**.

EVALUATION

We conducted a first test protocol in order to assess if our visual language allows designers to correctly design the interactive behaviour of their geographic Web-based application. To prepare and to carry out this evaluation, we focused on three research papers [2, 18, 21]. The goal was to assess the cognitive dimensions of our VPL notations. We wanted to concretely evaluate if designers understood the VPL components but also their underlying advantages and drawbacks.

[21] suggested that syntactic and semantic density were the main characteristics of visual languages. [2] proposed new dimensions with different levels of adoption and refinement such as consistency, visibility, viscosity, hidden dependencies, creative ambiguity or abstraction management. As more recently presented by [18], we used a set of nine principles for evaluating the cognitively effective visual notation of our VPL. This approach uses a combination of craft and scientific knowledge. Moreover, [18] proposed a modular structure which allows notation designer to easily add or remove principles where each principle is defined by a name, a semantic (theoretical) definition, an operational (empirical) definition, some design strategies, exemplars and counter exemplars.

Participants, Procedure, and Measurement

We invited thirty two volunteers in second year "DUT Informatique" (High National Diploma in Computer Science) to participate in this evaluation. None of the students had any significant academic experience of the UML sequence diagram. The evaluation procedure was organized into six steps.

Example presentation

We gave a 35 minutes presentation on a simple but complete example of how to design a Web-based geographic application. Within this example, we specified an application helping the user to know the "*préfecture*"[3] of a list of given cities.

The example application (Figure 11) displays both a list of towns and cities and a map initially displaying a visible point for each town/city. The application behaviour is presented as follows: When the user clicks on any place name in the top list, the application computes the "*préfecture*" of this place and the map zooms in on this "*préfecture*" (see **`http://bit.ly/uwAZne`**).

The sequence of interaction elements can be done in some equivalent ways for defining a visual interaction. For example, a sequence of a projection and a calculation or the inverse sequence produces two different diagrams, but the resulting behaviour is identical.

During this presentation, we focused on the *Interaction* phase in order for the participants to understand the role of our VPL components.

[3] A "*préfecture*" is a French administrative city corresponding to the main city of a "*département*".

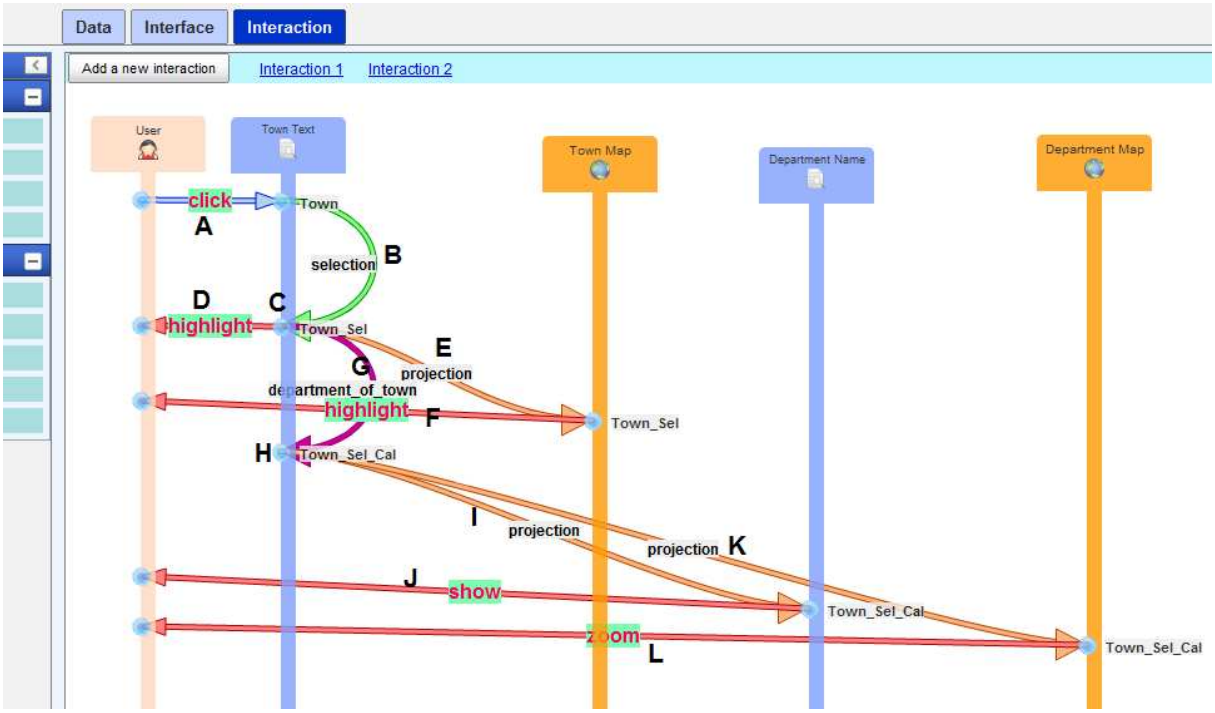

Figure 10. `Interaction` designed with WINDMash

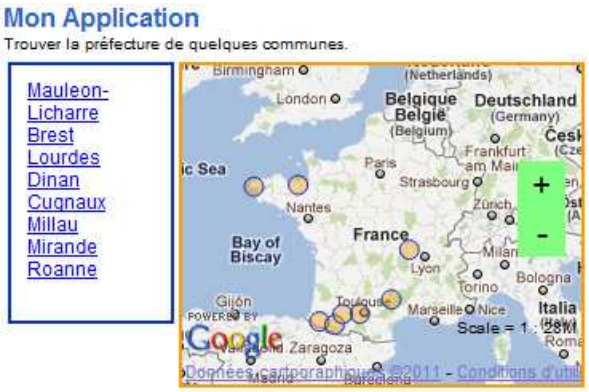

Figure 11. Screenshot of the example application

Oral presentation of the test application
Then, during ten minutes, we presented the work for participants to design the application presented in Figure 7.

Guided creation of the Content and Interface phases
Then, we have co-designed during ten minutes the two phases corresponding to the design of embedded geocontents (*Content* phase) and the design of graphical user interface (*Interface* phase).

Hand-made production of the Interaction phase
Then, we made a break with the WINDMash environment in order for the participants to focus on the design of the interactive abilities of the application. During fifteen minutes, they produced, without any help nor answer to their questions, some paper interaction diagrams using our VPL.

This preparatory work allowed them to focus on the VPL independently from the WINDMash environment.

Implementation and assessment of the Interaction phase
During fifteen minutes, participants had to translate their interaction diagrams and then to generate the executable code in order to assess the final application.

Evaluation
Last we asked participants to give us their hand-made paper interaction specifications while justifying if they had to make some changes within the WINDMash environment.

To conclude, we spent ten minutes with the participants to fill in an evaluation form composed of eleven questions (presented in a colloquial wording in French) relative to the nine cognitive dimensions [18] used to evaluate our VPL:

- **Semiotic clarity** (**SC1**: Is there redundancy between semantic constructs and graphical symbols? **SC2**: Is there deficit between semantic constructs and graphical symbols?)
- **Perceptual Discriminability (PD)**: Is any symbol clearly distinguishable to each other?
- **Semantic Transparency (ST)**: Does visual representation appearance suggest its meaning?
- **Complexity Management (CM)**: Does visual language include explicit mechanisms for dealing with complexity?
- **Cognitive Integration (CI)**: Does visual language include explicit mechanisms to support integration of information from different diagrams?

- **Visual Expressiveness (VE)**: Does visual language use the full range and capacities of visual variables?
- **Dual Coding (DC)**: Does visual language use text to complement graphics?
- **Graphic Economy (GE)**: Is the number of different graphical symbols cognitively manageable?
- **Cognitive fit** (**CF1**: Is visual language handled similarly on paper and within the WINDMash environment? **CF2**: Is visual language handled well by designers unfamiliar with UML sequence diagrams?)

For any question, four possible answers were available (Strongly Agree, Agree, Disagree, Strongly Disagree). Moreover, participants had the options to justify their answer with a short open comment.

Results and Discussion

Figure 12 resumes the participants' answers:

	Strongly Agree	Agree	Disagree	Strongly Disagree	
SC1	0%	0%	13%	88%	⁚‿
SC2	0%	0%	63%	38%	⁚‿
PD	38%	50%	0%	13% ⁚⁔	⁚‿
ST	50%	38%	13%	0%	⁚‿
CM	38%	63%	0%	0%	⁚‿
CI	13% ⁚⁔	13% ⁚⁔	25% ⁚‿	50% ⁚‿	: –
VE	13%	50%	38%	0%	: –
DC	0%	13%	38%	50%	⁚‿
GE	25%	75%	0%	0%	⁚‿
CF1	75%	25%	0%	0%	⁚‿
CF2	0%	63%	13%	25%	: –

Figure 12. Result table of the experimentation

From a global viewpoint, the evaluation results are encouraging as illustrated by smileys added in the above result table.

However, some specific points still need to be improved. Relating to Cognitive Integration, it should be interesting to define mechanisms allowing designers to reuse interaction parts of a diagram in other diagrams. This traditional software engineering approach (black box vs. glass box) would increase reuse abilities of the VPL. Another interesting perspective relates to the Visual Expressiveness principle. We should improve the usability of each interaction element, and specially its visual representation, to better suggest its role. The underlying design environment may also better check connection possibilities between graphic elements in order to reduce some possible designers errors.

Moreover, for the test application, participants have produced two different artifacts/deliverables. We may raise some patterns to their strengths and weaknesses. They first produced a hand-made production of the Interaction phase. During this step, 81% of the participants have produced a "fully-correct" proposal. Other 19% have syntactic problems which may be hided/overcome by an electronic assistance-tool (our graphical Interaction description WINDMash tool). 6% of the participants have produced a proposal with 4 diagrams whereas 94% used 2 diagrams.

In addition, the study of hand-made interaction diagrams highlights the simplicity, the flexibility and the power of our VPL that allows designers to specify complex interactions: it is possible to decompose the description of a complex interaction into some simple interactions whose behaviour is equivalent to the one of the complex interaction. For example, a participant proposed the four following diagrams (Figure 13) that are equivalent to the diagram in Figure 10:

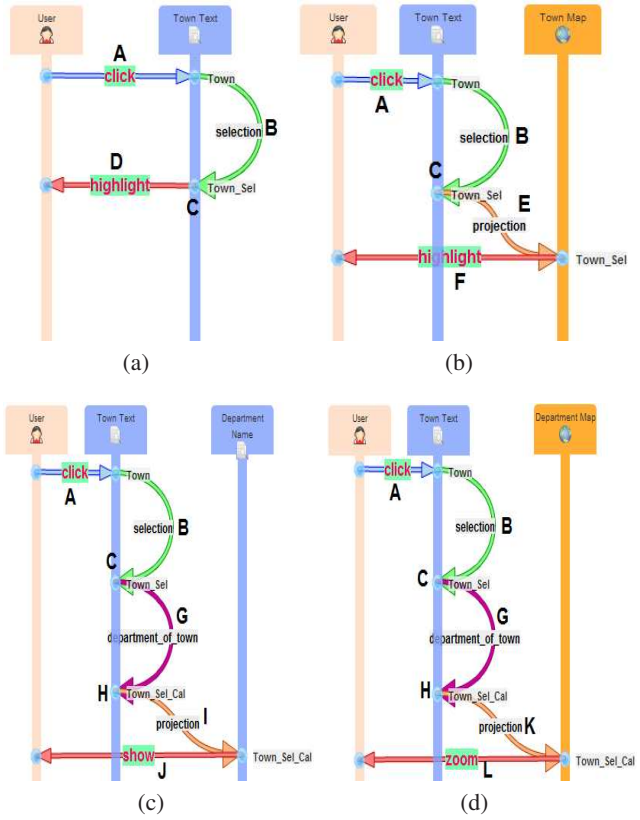

Figure 13. Interaction diagram in Figure 10 divided into four diagrams

- the first for describing the highlight of the "`Town`" name clicked in the displayer named "`Town Text`" (Figure 13(a))
- the second for describing the highlight in the displayer named "`Town Map`" of the "`Town`" clicked in the displayer named "`Town Text`" (Figure 13(b))
- the third for calculating the department of the selected "`Town`", then sending and showing the result of this calculation in the displayer "`Department Name`" (Figure 13(c))
- the fourth for calculating the department of the selected "`Town`", then sending and zooming in on the result of this calculation in the displayer "`Department Map`" (Figure 13(d))

The interaction elements used in Figure 10 (A, B, C, ..., L) can also be found in Figure 13 but they are dispatched into four diagrams.

Lastly, the participants had to transfer their hand-made proposal into the WINDMash environment and to generate their application to immediately check their design. As WINDMash is still a prototype, most of the participants encountered problems due to our environment and especially when they wanted to correct or to modify a diagram. Our system is still unstable and they had to start again the whole three-steps process. Hence, during the 15 minutes given time, some participants (a quarter) could not achieve. However, among those who had syntactic problems with their hand-made proposal, all could correct some or all of their "mistakes". This was quite frustrating for the participants but we had noticed that it was "only" a debugging problem for our WINDMash environment (that we have to shorten drastically before next evaluation).

CONCLUSION AND FUTURE WORK

In this paper we have offered a VPL allowing designers to specify and to implement interactions into Web-based geographic applications. Our VPL has been integrated in a graphic design framework called WINDMash for designing applications according to three main views:

- Which data (geocontents) must be emphasized?
- How this data must be visually displayed?
- What kind of interactions are available to handle this data?

The considered interactions focus on users and are always described in terms of user action triggering system reactions. We gave priority to this "simple" vision of interactions for two main reasons:

- to facilitate designers' work by describing "simple" processes rather than a global and complex process;
- to favour the elaboration of operational models that can be exploited to generate executable code from visual specifications thanks to model transformation techniques.

Interactions diagrams resulting from our VPL highlight the exchanged flow between the user and the system but also between the interface elements composing the system. We have chosen to describe interactions according to data presented on the interface. We think that it is a natural way to describe:

- the interface elements that may trigger an interaction;
- how the system will react from a visual viewpoint.

The resulting interaction diagrams may be simple but these diagrams are always described with high level abstraction elements that increase the expressive power and the meaning of the diagram: The designer may define what happens when a user selects a town, without defining what is a town. Designating something on the interface is easy to describe / understand from a graphical viewpoint, even if the designated element is something complex (any town).

Our VPL is mostly independent from the geographic field we work on because interactions are described according to:

- user actions which deal with designating data presented on a displayer;
- internal system reactions which deal with computing what has been selected by the user, transferring data toward specific displayers or computing new data (in this work calculation operations are the only elements that are specific to the geographic area);
- external system reactions which deal with emphasizing (*highlight*, *show*, *hide*, etc.) specific or computed data.

As shown on the experimentation presented in this paper, our VPL offers sufficient flexibility to manage the complexity of an interaction. According to their expertise level, designers may choose to describe a complex interaction (composed of multiple system reactions) with a single diagram or with several diagrams, each one specifying a system reaction part of the global interaction.

First evaluation results about our VPL are very encouraging. Of course, the evaluation task is still a work in progress and we must elaborate new test protocols to evaluate our interaction language according to other criteria such as those presented in [2]. However, this first experimentation has shown that our visual language is rather easy to master for designers specialists knowing UML sequence diagrams. We have noted that sequence diagrams seem particularly adapted when it deals with describing interactions thought in terms of flow exchanges between a user and the system but also between system components.

Future work deals with elaborating mechanisms allowing designers to check the consistency of the designed interactions. As seen in this paper, describing interactions in a separate way is of course an advantage when it deals with designing a complex interactive system. However, this advantage can also become a problem when designers need to evaluate the consistency of all the system's interactive possibilities. Currently, interactions are specified separately by a set of diagrams but there is no mechanism control that checks the global coherence of the described interactions. The interactions described below introduce an example of two ambiguous specifications that can lead to an incoherent system reaction:

- When the user clicks on a town in the text, the map zooms in on this town;
- When the user clicks on a town in the text, the map displays the corresponding administrative province of this town.

Currently, each specified interaction is encoded using RDF. We plan to exploit the underlying merging and querying mechanisms of RDF to try to detect possible inconsistencies between all interactions specified by the designer. This hard task deals with specifying rules that must been verified to ensure the consistency between two diagrams. The idea is to integrate these checking rules into WINDMash in order to provide a first assistance level allowing designers to keep control over the global behaviour of the elaborated application.

ACKNOWLEDGEMENTS

This work has been supported by the ANR MOANO (`http://moano.liuppa.univ-pau.fr`) project.

REFERENCES

1. Barbier, F. Supporting the uml state machine diagrams at runtime. In *Model Driven Architecture Foundations and Applications*, I. Schieferdecker and A. Hartman, Eds., vol. 5095 of *Lecture Notes in Computer Science*. Springer Berlin / Heidelberg, 2008, 338–348.
2. Blackwell, A., and Green, T. R. A Cognitive Dimensions Questionnaire Optimised for Users. In *Proceedings of 12th Workshop of the Psychology of Programming Interest Group* (Corigliano Calabro, Cosenza, Italy, 2000), 137–154.
3. Boshernitsan, M., and Downes, M. S. Visual programming languages: a survey. Tech. Rep. UCB/CSD-04-1368, EECS Department, University of California, Berkeley, Dec 2004.
4. Botturi, L., and Stubbs, T., Eds. *Handbook of Visual Languages for Instructional Design - Theories and Practices*. Information Science Reference, 2007.
5. Bowles, J. Decomposing interactions. In *Algebraic Methodology and Software Technology*, M. Johnson and V. Vene, Eds., vol. 4019 of *LNCS*, Springer Berlin / Heidelberg (2006), 189–203.
6. Burnett, M. M., and Baker, M. J. A classification system for visual programming languages. *J. Vis. Lang. Comput. 5*, 3 (1994), 287–300.
7. Chang, S.-K., Ed. *Principles of visual programming systems*. Prentice-Hall, Inc., 1990.
8. Cooper, A. *About Face: The Essentials of User Interface Design*, 1st ed. John Wiley & Sons, Inc., 1995.
9. Dobesova, Z. Visual programming language in geographic information systems. In *Proceedings of the 2nd international conference on Applied informatics and computing theory*, AICT'11, World Scientific and Engineering Academy and Society (WSEAS) (Stevens Point, Wisconsin, USA, 2011), 276–280.
10. Engels, G., Hausmann, J. H., Heckel, R., and Sauer, S. Dynamic meta modeling: a graphical approach to the operational semantics of behavioral diagrams in UML. In *Proceedings of the 3rd International Conference on the Unified Modeling Language: advancing the standard*, UML'00, Springer-Verlag (2000), 323–337.
11. Etcheverry, P., Marquesuzaà, C., and Corbineau, S. Designing suited interactions for a document management system handling localized documents. In *Proceedings of the 24th annual ACM international conference on Design of communication*, SIGDOC '06, ACM (2006), 188–195.
12. Harel, D., and Marelly, R. *Come, Let's Play: Scenario-Based Programming Using LSC's and the Play-Engine*. Springer-Verlag New York, Inc., 2003.
13. Hennicker, R., and Koch, N. Modeling the user interface of web applications with UML. In *Workshop of the pUML-Group held together with the "UML"2001 on Practical UML-Based Rigorous Development Methods - Countering or Integrating the eXtremists* (2001), 158–172.
14. Khler, H.-J., Nickel, U., Niereand, J., and Zndorf, A. Using UML as visual programming language. *Technical Report tr-ri-99-205* (1999).
15. Koegel, J. F., and Heines, J. M. Improving visual programming languages for multimedia authoring. In *ED-MEDIA '93, World Conference on Educational Multimedia and Hypermedia* (1993), 286–293.
16. Luong, T. N., Laborie, S., and Nodenot, T. A framework with tools for designing web-based geographic applications. In *ACM Symposium on Document Engineering* (2011), 33–42.
17. Manola, F., and Miller, E. RDF Primer. Recommendation, W3C, February 2004. `http://www.w3.org/TR/rdf-syntax/`.
18. Moody, D. The physics of notations: Toward a scientific basis for constructing visual notations in software engineering. *IEEE Trans. Softw. Eng. 35* (November 2009), 756–779.
19. Myers, B. A. Taxonomies of visual programming and program visualization. *Journal of Visual Languages and Computing 1* (March 1990), 97–123.
20. Narayanan, N. H., and Hübscher, R. *Visual language theory: towards a human computer interaction perspective*. Springer-Verlag New York, Inc., New York, NY, USA, 1998, 87–128.
21. Raymond, D. R. Characterizing visual languages. In *Proc. 1991 IEEE Workshop on Visual Languages. (Kobe*, Society Press (1991), 176–182.
22. Shu, N. C. Visual programming: Perspectives and approaches. *IBM Systems Journal 38*, 2/3 (1999), 199–221.
23. Stubbs, T., and Gibbons, A. The power of design drawings in other design fields. In *Handbook of Visual Languages in Instructional Design; Theories and Practice* (2007).
24. Stühmer, R., Anicic, D., Sen, S., Ma, J., Schmidt, K.-U., and Stojanovic, N. Lifting events in RDF from interactions with annotated web pages. In *Proceedings of the 8th International Semantic Web Conference*, Springer-Verlag (2009), 893–908.
25. Traynor, C., and Williams, M. G. *End users and GIS: a demonstration is worth a thousand words*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 2001, 115–134.
26. Ziadi, T., Blanc, X., and Raji, A. From requirements to code revisited. In *Proceedings of the 2009 IEEE International Symposium on Object/Component/Service-Oriented Real-Time Distributed Computing*, ISORC '09, IEEE Computer Society (Washington, DC, USA, 2009), 228–235.