Towards a taxonomy for interactive graphics systems[1]

G. Pierra

LISI/ ENSMA
B.P. 109 - 86960 FUTUROSCOPE Cédex
e-mail : pierra@ensma.fr

Abstract. It has often been pointed out that the different architecture models proposed for interactive computer graphics were too much imprecise. Global architecture models intended to define the macro-modules that constitute such systems do not precise the functional description nor the interface of these modules. Multi-agent models, intended to define the micro-structure of the building blocks of such systems, do not precise the criteria to be used for agent identification and structurization, nor the complete set of relationships that exists between these agents and the domain-specific component that represents the semantic part of the system.
In this report we propose a taxonomy for interactive graphics systems through seven orthogonal criteria. These criteria enable to classify every systems during the analysis phase. The possible uses of this taxonomy include the following:

- facilitating the selection process of a suitable architecture model for a system under design,
- enabling the architecture model designers to precise the classes of systems that constitute the target application domain of their models,
- promoting the emergence of precise architecture models that address the requirements of the different categories of interactive graphics systems.

[1] *paru dans les* Proc. of Design, Specification Validation of Interactive Systems, Palanque, P., Bastide R., Eds Springer Computer Science Series, Spronger, 1995, pp. 362-370.

Towards a taxonomy for interactive graphics systems

G. Pierra

LISI/ ENSMA
B.P. 109 - 86960 FUTUROSCOPE Cédex
email : pierra@ensma.fr

Abstract. It has often been pointed out that the different architecture models proposed for interactive computer graphics were too much imprecise. Global architecture models intended to define the macro-modules that constitute such systems do not precise the functional description nor the interface of these modules. Multi-agent models, intended to define the micro-structure of the building blocks of such systems, do not precise the criteria to be used for agent identification and structurization, nor the complete set of relationships that exists between these agents and the domain-specific component that represents the semantic part of the system.
In this report we propose a taxonomy for interactive graphics systems through seven orthogonal criteria. These criteria enable to classify every systems during the analysis phase. The possible uses of this taxonomy include the following:

- facilitating the selection process of a suitable architecture model for a system under design,
- enabling the architecture model designers to precise the classes of systems that constitute the target application domain of their models,
- promoting the emergence of precise architecture models that address the requirements of the different categories of interactive graphics systems.

1 Introduction

The goal of architecture models for interactive graphics systems is, or should be, to provide guidelines to system designers, both for modular decomposition of their system, and for selection of the suitable software tools. In fact, a lot of models have been proposed. Therefore, the first designer's choice is to select the suitable architecture model for the system under design.

The UIMS Tool developers Workshop [22] already pointed out that the suitable architecture of the application may depend on the designer's goals (e.g., maximising runtime performance vs. buffering the remainder of the system from the effects of

evolving Interaction Toolkits). In this report we suggest that such an architecture is heavily dependent on the specific requirements of the target application and we propose seven orthogonal criteria that enable to classify systems during their analysis phases.

This paper results from the discussions that took place during the Eurographics Workshop on Design, Validation and Specification of Interactive Systems (DVS-IS'95) in the Working Group on Taxonomy. The members of this working group were the following: B. David (ECL, Fr), P. Girard (LISI/ENSMA, Fr), F. Jensayer (Roskilde Univ., Dk), J. Munoz (LIS/Toulouse 1 Univ., Fr), G. Pierra (LISI/ENSMA, Fr), M. Rautenberg (ETHZ, Ch), J. Vanderdonckt (FUNDP/Namur, Be). The starting point of these discussion was a position paper [19].

The content of this report is as follows. In the first section, we recall the content of the Arch model [22] that provides a useful framework for describing the runtime architecture of interactive systems. In the second section, we discuss the goal of the taxonomy and the meta-criteria used to select the relevant criteria. In section three we present the criteria that define the proposed taxonomy.

2 Architecture models for interactive graphics system

A lot of models exist for the design of interactive systems. Each model focuses on a different point and is intended to solve a different problem. Static architecture models mainly address the modular structure of the system. Some of them, such that the Seeheim model [12], or the Computer Graphics Reference Model [2, 7] suggest a top-down approach. They define the macro-modules that should constitute the system. Some others, like MVC [11] or PAC [4], suggest a bottom-up approach. They define the structure of the building blocks that should constitute the complete system. Dynamic architecture models are intended to capture the behaviour of interactive systems. Once again some of them are more top-down oriented, such that the models based on the linguistic approach [9, 23], some others, often called multi-agent models, define the fine-grain reactive units that should be used to model this behaviour [6, 8, 13, 17, 21]. It is not clear how the top-down-oriented and the bottom-up-oriented models fit together. And it does not exist any consensus about when, and even how, the different models should be used [5, 20].

In order to get a common understanding and terminology about the runtime architecture of these systems, a series of workshops took place in 1990 and 1991. The result of these workshops, known as the Arch model, was published in 1992 [22]. The Arch model defines five components (sub-systems) for the architecture of interactive systems. The two ends of this architecture are the *Domain-Specific Component* and the *Interaction Toolkit Component* that may be considered, for some applications, as pre-existent (see fig 1)

The *Interaction Toolkit Component* implements the physical interaction with the end-user (via hardware and software). The *Domain-Specific Component* controls, manipulates and retrieves domain data, and performs other domain-related functions. Between these two ends, the *Dialogue Component* has responsibility for task level

sequencing and for mapping back and forth between domain-specific formalisms and user-interface-specific formalisms. Two more components are defined for buffering an operational system from changes in technology. The *Presentation Component* buffers from changes in user interface toolkit by providing an abstract view of user Interaction Objects. The *Domain-Adaptor Component* is a mediation between the Dialogue and the Domain-Specific Components. In this report, the Interaction Toolkit Component and the Presentation Component will sometimes be referred to as the *user-side components*, the Domain-Specific Component and the Domain-Adaptor Component as the *domain-side components.*

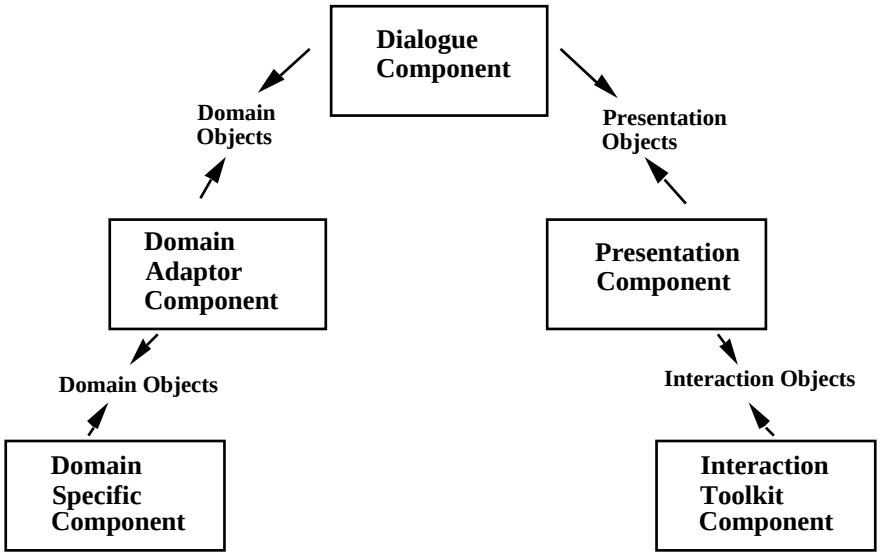

Fig. 1. The Arch model

To acknowledge the diversity of real-world applications, the Arch report points out that the same functionality may be shifted from one component to another one of the architecture. It uses the term "Slinky" to emphasise that the Arch model is in fact a meta-model that may be differently tailored both regarding the component to which is assigned some particular functionality (functional specification), and regarding the data flow between the various components (interface specification). Besides, the Arch model does not consider, but implicitly for the Interaction Toolkit Component, the internal structure of each component.

It shall be noticed that, unlike most of the other proposed models, the Arch model is not intended to be a prescriptive model. In particular, the model itself does not specify neither how to separate an interactive system into the different identified subsystems, nor the internal structure of each subsystem. The main interest of this model is to substantially define a framework within which any interactive system or prescriptive model for such systems may be described and/or compared. This framework may thus be used to discuss the criteria that significantly affect the software architecture of interactive systems.

3 Goals of the taxonomy

The goal of the proposed taxonomy is to identify the characteristics of interactive graphics systems that have a major impact on the design of these systems. The intended uses of such a taxonomy include the following

- facilitating the selection process of a suitable architecture model for a system under design,
- enabling architecture model designers to precise the classes of systems that constitute the target application domain of their models,
- promoting the emergence of precise architecture models that address the requirements of the different categories of interactive graphic systems.

In order to support the first usage, the values of the characteristics used as taxonomy criteria shall be known before the beginning of the design process, i.e., during the analysis phase.

Regarding the criteria that should be used in the taxonomy, the Arch (meta-) model enables to assess the impact of given characteristics on the system architecture. Every characteristics may be considered as relevant criteria for the taxonomy if their values:

- affects the functionalities which are assigned to some Arch component (functional specification) and/or,
- affects the data flow between the different Arch components (interface specification), and/or
- affects the internal structure of some Arch component (detailed design).

4 The proposed taxonomy

We assume that the system analysis is done using some object oriented analysis (OOA) method, (e. g., the COAD&YOURDON OOA method [3]).

The result of such an analysis is twofold. First, the Domain Objects are modelled. In our proposed taxonomy, two criteria address the Domain Objects structure. Second, the system's responsibilities are defined. In interactive systems the fundamental system responsibility is to support the user tasks [1]. Two criteria address the task structure. In order to cover a large spectrum of applications, two other criteria address the actor model and characterise the control source (the user, the Domain-Specific Component, both) and the number of simultaneous users. The last criteria address an operational constraint on the system that results from multi-modal interactions.

All the criteria, but the control source criteria, identify orthogonal degrees of complexity for the system under design. For each criterion, only two values are discussed: the simplest case and the most complex case. Actual systems are often in between, and ad-hoc practices are sometimes used to remove some degrees of complexity.

The proposed criteria are the following.

4.1. Tasks arity

An application supports mono-object tasks if each user task involves only one (simple or structured) Domain Object. It supports multi-object task if a task involves several Domain Objects. Mono-object tasks may be encapsulated into Domain Object representation. They may be supported by direct manipulation techniques. In the simplest case, the dialogue component may not exist on its own. Multi-object tasks shall be represented independently from the Domain Objects. Direct manipulation techniques are not possible. Some dialogue component, of which the structure is independent from the object structure, shall exist. A typical ad-hoc solution to avoid this degree of complexity consists in providing multi-object selection (e.g., rubber rectangle) and enabling the user to do the same task on the set of objects (set traversal [14]). As an example, Mac Draw™ only supports mono-object tasks. Example of multi-object tasks are provided by the drafting systems that enable to create lines as , e. g., tangential to two circles.

4.2. Tasks structuring

An application supports atomic task if the user must specify each of its tasks independently, the result of this task being recorded in the state of the Domain-Specific Component. An application supports structured tasks if the user may input in pre-order its task/sub-task hierarchy [16], the result of the overall tasks being only recorded in the application domain component when the whole task hierarchy has been input. The support of structured tasks needs to record the state of the dialogue independently from the state of the Domain-Specific and of the Interaction Toolkit components. Atomic tasks may be encapsulated either in Domain Object or in Interaction Objects. Typical ad-hoc solutions to avoid this degree of complexity consists in providing modal dialogue variables that may be recorded in the Interaction Toolkit Component and in designing some predefined structured task patterns associated with a specific set of interaction objects (e.g., structured sets of dialogue boxes).

An example of structured task is provided by graphic expressions such expressions enable to specify an operand of some high level task by means of a function which the domain are other objects (e.g., to define the centre of a circle as the middle-point of some line).

4.3. Domain Objects autonomy

The Domain Objects are autonomous if the presentation of each Domain Object is only (or mainly) dependent on the state of the corresponding Domain Object. The Domain Object are relational if the presentation of each Domain Object is dependent on the state of other Domain Objects. If the Domain Objects are autonomous, each Domain Object may be mapped onto one Interaction Object that supports its rendering function [6]. If the Domain Objects are relational, the Interaction Toolkit Component shall provide rendering spaces that are used by a global rendering feature of the domain-side components. Graphics Standards, such that GKS [10], PHIGS [18] or CGRM [7] provide high-level mechanisms for rendering (geometric) related objects. Models like MVC [11] or PAC [4] are straightforward for rendering autonomous objects. Moreover, when Domain Objects are autonomous, high level

semantic feedback may be shifted from the domain-side components to the user-side components. It is not possible when objects are relational.

4.4. Domain Objects structuring

Domain Objects are structured when several levels of objects, structured by aggregation, may be accessed by the user. They are simple when one Domain Object is not part of another Domain Object. When Domain Objects are (highly) structured, the designation of one Domain Object may only be interpreted by the domain-side components. It requires a complete traversal of the system by the user-defined events. When Domain Objects are simple enough, the pick identifier may be interpreted, and echoed, by the user-side components. A typical ad-hoc solution to avoid this degree of complexity consists in providing some "association" function that enables gathering several objects in an aggregate object, the internal object being no longer pickable. This is done, for instance, in Mac Draw™.

4.5. Control source

The fifth criterion characterises the source of the events of which the type changes the dialogue component state. It may be either the user (interactive application) or the Domain (conversational application) or both (dialogue application). The dialogue component should offer asymmetric functionalities in the two first cases. It offers symmetric functionalities in the last one.

4.6. Mono/Multi user interactive system

A mono-user interactive system correspond to a system where only one user dialogues with a Domain Specific Component. This category of systems belongs to the scope of the Arch model. Multi user systems enable several users to interact with the same Domain-Specific Component. Multi-user interactive systems require an "Y" runtime architecture where several instances (with possible links) exist for some Arch components.

4.7. Sequential Vs Real Time interactive system

An interactive system is sequential if the behaviour of the system is only dependant on the order of the sequences of events and values input by the user-side and the domain-side components. It is a Real Time interactive system when the behaviour is also dependent on the time where an event or a value occurs. A Real Time system needs to have a timer within its Dialogue Component. It has been proved that multi-modal Interactive systems were Real Time Systems [15]. The following table summarises the proposed taxonomy.

criteria			
tasks arity	**mono-object tasks** • Task representation may be encapsulated in object representation. *Example:* Mac Draw™		**multi-object tasks** • Tasks shall be represented independently from objects. *Example: Database graphical interface*
tasks structuring	**atomic tasks** • no (or few) dialogue context. *Example:* Mac Draw™		**structural tasks** • A structured dialogue context shall be explicitly modelled. *Example: use of a display calculator to input some real value to a current task*
objects autonomy	**autonomous objects** (the presentation of Domain Object depends only on the state of the corresponding Domain Object) • One object in the user-side components reflects each Domain Object *Example:* Mac Draw™		**relational objects** (the presentation of a Domain Object depends on the state of the other Domain Objects). • No one-to-one relationship between Domain Objects and presentation objects *Example: process control interface*
objects structuring	**simple objects** • object designation may be done in the presentation component *Example:* Mac Draw™, *GKS*		**(highly) structured objects** • only Domain-side components are able to identify the Domain Object selected by a pointing device *Example: solid modelers*
control source	**user: interactive application** **• asymmetric dialogue** - control events come from the user - Domain reports to the Dialogue component (e.g., semantic error) - Domain reports by rendering to the user *Example: graphic editor*	**Domain: conversational application** **• asymmetric dialogue** - control events come from the Domain - data flow from and to the user *Example: login process*	**user + Domain: dialogue application** **• symmetric dialogue** - control events come from both the user and Domain - data flow from and to both the user and Domain the user *Example: process control interface*
mono/multi user	**mono user** • The run time architecture corresponds to the Arch model		**multi user: Y model** • The designer shall decide which components are shared and which ones are not *Example: CSCW systems*
sequential/Real Time	**sequential** • no timer *Example: usual Business applications*		**Real Time** • A timer shall exist in the Dialogue Component *Example: multi-modal systems*

Table 1: Taxonomy of interactive graphic system

Conclusion

In this report we have proposed a taxonomy of interactive graphics systems that is based on seven orthogonal criteria. We have shown that each of these criteria has a significant impact on the suitable architecture of the system to be designed. The values of these criteria are known during the analysis phase of a system. Therefore, they may be used to evaluate the suitability of a particular prescriptive architecture model for a system under design.

In this report, we have not discussed the suitability of any existing prescriptive architecture model for the various categories defined by the taxonomy. Nevertheless, it is hoped that reference to this taxonomy will contribute to the emergence of new and more precise prescriptive architecture models for the different categories of interactive systems we have identified.

References

1. Bass, L., Coutaz, J.: Developing software for the user interface, Addison-Wesley (1991).

2. Carson, G.: Introduction to the Computer Graphics Reference Model, Computer Graphics, 27, 2, 108-118, (1993).

3. Coad, P., Yourdon, E.: Object Oriented Analysis, Prentice Hall (1991)

4. Coutaz, J.: Interface homme-machine : un regard critique. Journées d'étude AFCET: Interface Homme-Machine, Paris, 21 oct. 1992,1-24 (1992)

5. Coutaz, J.: PAC: an implementation Model for Dialog Design. Proc. Interact'87, North Holland Publ., 431-436 (1987).

6. Duke, D., Harrison, M.: Abstract Interaction Objects. Computer Graphics Forum, 12, 3, 25-36 (1993).

7. Faconti, G.: The Reference Model of Computer Graphics. in: D.A. Duce *et al.* (eds): User Interface Management Design. New York, Berlin Heidelberg New York Tokyo: Springer-Verlag 1990, 7-14 (1990).

8. Faconti, G., Paterno, F.: An Approach to the Formal Specification of the Components of an Interaction, EUROGRAPHICS'90, 481-494 (1990).

9. Foley, J., Wallace, V.L.: The Art of Natural Graphic Man-Machine Conversation. Proc. of IEEE G2, (1974).

10. ISO/IS 7942, Information Processing Systems, Computer Graphics, Graphical Kernel System - Functional Description (1985).

11. Goldberg, A.: Smalltalk-80: The Interactive Programming Environment, Addison-Wesley (1984).

12. Green, M.: Report on on Dialogue-Specification Tools. In: G.E Pfaff (eds.): User Interface Management Systems. New York, Berlin Heidelberg New York Tokyo: Springer-Verlag 1985, 9-20 (1985).

13. Green, M.: A survey of three dialogue models, ACM Trans Graph. 5, 3, 244-275 (1986).

14. Halbert, D.: Programming by example, PhD. Thesis, Berkeley, California (1984).

15. Nigay, L.: Conception et modélisation logicielle des systèmes interactifs : application aux interfaces multimodales, PhD. Thesis, Université Grenoble 1 (1994).

16. Norman, D.: User Centered System Design, Lawrence Erlbaum Associates (1986).

17 Paterno', F.: A Theory of User-Interaction Objects, Journal of visual languages and computing, 5, 3, 227-249, (1994).

18 ISO/IS 9592:1989, Information Processing Systems, Programmers Hierarchical Interactive Graphics System - Functional Descritpion (1989).

19. Pierra, G., Girard, P., Guittet, L.: Towards precise architecture models for computer graphics: The H^4 architecture, position paper, in: Pre-Proceeding of Eurographics Workshop on Design, Validation and Specification of Interactive System, Bonas, France, June (1995).

20. Ten Hagen, P.: Critique of the Seeheim Model. in: D.A. Duce *et al.* (eds): User Interface Management Design. New York, Berlin Heidelberg New York Tokyo: Springer-Verlag 1990, 3-6 (1990).

21. Ten Hagen, P., Derksen, J.: Parallel input and feedback in dialogue cells. In: G.E Pfaff (eds.): User Interface Management Systems. New York, Berlin Heidelberg New York Tokyo: Springer-Verlag 1985, 109-124 (1985).

22. The UIMS Developers Workshop - A Metamodel for the run time Architecture of An Interactive System; SIGCHI Bulletin, 24, 1, 32-37 (1992).

23. Woods, W.: Transition network grammars for natural langage analysis. Comm. ACM,13, 10, 591-606 (1970).