A Distributed Multimedia Architecture for Intent-Based Video Authoring and Presentation

Steven Gribble
gribble@cs.ubc.ca

Andrew Csinger
csinger@cs.ubc.ca

Kellogg S. Booth
ksbooth@cs.ubc.ca

Department of Computer Science
University of British Columbia, Vancouver, BC

Abstract

The traditional authoring paradigm is fraught with difficulties and limitations, foremost of which is the compile-time binding of form and content of a document. By contrast, the intent-based authoring paradigm allows the form and content of a document to be determined at run-time, which is achieved through the decoupling of content and intent.

A prototypical distributed multimedia platform (Valhalla) is introduced. Agents inhabiting the Valhalla framework are members of one of three classes: client applications, media servers, and other service providers (such as an artificial intelligence based reasoning engine to support intent-based authoring). Implementations of members of these classes are introduced through the description of a departmental hyperbrochure application.

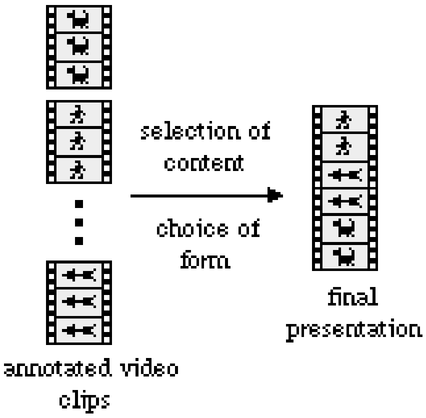

Figure 1: The Traditional Video Authoring Paradigm

1 Introduction

A traditional video author is faced with a number of well-identified but seemingly unavoidable obstacles. The author must first acquire, log, and annotate a database of raw footage, from which must be selected a list of clips to be cut into a final presentation. In most cases, the fruit of the author's labour is a single presentation, bound both in content and form.

Each stage in the authoring process requires a large investment of time, due to the linear temporal nature of the video medium and the iterative refinement that a typical author performs. The experience and knowledge gained during this lengthy process is almost largely wasted; since a single presentation is the end product of the process, the author is constrained to attempt to satisfy the needs of all potential viewers rather than customizing the presentation to accommodate the requirements of individuals.

Improvements on the traditional authoring paradigm can be introduced. The logging and annotation of footage do not necessarily have to be performed by the author, and can to some extent be automated. Once logged and annotated, a database of raw footage can be reused by subsequent authors to create different presentations. The form of the final presentation can also be cleverly chosen to provide viewers some choice in content; for example, hypermedia allows authors to pass some of the responsibility of content selection to the viewer.

However, introduced with each of these potential improvements is a new set of problems. Annotators and authors reusing annotated footage are prone to fall prey to the syntactic ambiguity and semantic unpredictability problems, identified by Csinger *et al.* [1]. Hypermedia authors, relieved of some of the burden of deciding final content, must instead provide many navigational links and must ensure that a viewer can easily navigate through those links to all potentially desirable content.

An alternative authoring paradigm called intent-based authoring attempts to solve the problems encountered by a traditional video author through the decoupling of form and content, and through the separation of the notions of content and intent. A prototype distributed multimedia platform (named Valhalla) is being developed at the University of British Columbia that includes a facility for intent-based authoring.

2 An Overview of the Intent-based Authoring Paradigm

The "compile-time" commitment to both form and content is the greatest deficiency of the traditional authoring paradigm. Short of producing a revised version of the original, a traditional author cannot tailor presentations to suit the characteristics of individual viewers.

Form, Content, and Intent

The decoupling of the form and content of a presentation (or document) is not a new idea. Structured document systems such as LaTeX and markup languages including SGML and HyTime attempt to separate the specification of the content of a document from the selection of its form, which permits a viewer to recompile the document to suit his or her particular preferences. However, such systems still require an author to select the content of the presentation, and in fact the author also provides a partial specification of the document's structure in the form of a hierarchical decomposition of its content.

Unlike intent-based presentation (see, for example, Seligmann and Feiner [6] or Karp and Feiner [3]), the intent-based *authoring* paradigm identifies a third attribute of a presentation in addition to form and content - the author's *intent.* Intent is usually implied in traditionally authored presentations; for example, the intent of a textbook is to educate, the intent of an advertisement is to sell, and the intent of many feature films is to entertain. If intent is wholly decoupled from form and content, it then becomes possible to tailor presentations to individual users at run-time rather than compile-time. Once knowledge about the characteristics and goals of the viewer is made available to the presentation system, then the content as well as the form of the final presentation can be selected in order to satisfy the intent of the author while catering to the needs of each individual viewer.

Knowledge-bases

As hinted to above, the automatic generation of form and content by an intent-based authoring system depends on the existence of a number of knowledge-bases. A minimal set of these bases may include:

1. **Domain independent knowledge:** an intent-based author must explicitly provide the system with his or her intent in the form of a presentation schema. A schema is an arbitrarily complex blueprint of a class of presentations. This class is populated by the set of all (uninstantiated) presentations having the same intent but which are customized to suit the needs of individual viewers.

2. **Domain dependent knowledge:** if a particular schema is to be used to create a presentation, then the domain-independent elements of that schema need to be instantiated using domain-specific information. For example, if a client of a company is to be given a presentation with the intent "inform", and the "inform" schema calls for a video clip illustrating the entity about which the viewer is to be informed, a set of axioms must be available to map the domain-independent requirement for an illustrative clip into the specific clip that satisfies this requirement. A domain expert is required to provide the set of axioms corresponding to his or her particular domain of expertise.

   In addition to these axioms which provide a mapping from the abstract intent of the author into domain specific information, a set of suitably annotated video clips must be available. The authoring system will select some number of clips in an attempt to satisfy the intention of the author.

3. **Media specific knowledge:** the particular characteristics of the media being used should be taken into account. For example, human perceptual abilities place a lower bound on the length of video clips chosen for a presentation - ignoring the use of subliminal suggestions, it makes little sense to show a clip of video that is less than 1/30th of a second in length. Aesthetic considerations can also be embodied in a media-specific knowledge base; perhaps a catalog of effective cinematographic techniques would be useful when

adding transitional cuts between segments in a video presentation. This type of information (and more) might be found in a media-specific knowledge base.

4. **User models:**

   Models of the users that will be viewing the presentations embody the final repository of knowledge needed by an intent-based authoring system. User models (Kobsa [4] or Wahlster and Kobsa [7]) may be acquired in a number of ways: they might be static models programmed by a knowledgeable agent, or they may dynamically change through direct, explicity feedback from the users themselves or through inferences drawn from the actions of the users. The information contained within a user model can be exploited to various degrees by the authoring system.

The Reasoning Engine

The final piece missing from a minimal intent-based authoring system is some sort of function or algorithm that will actually construct presentations based on the information contained within the various knowledge-bases. This algorithm can be thought of as a black box, which has a number of inputs and which produces as a single output a final presentation (Figure 2). In the framework discussed in this paper, the black box is implemented as a Horn-Clause reasoning engine, written in the prolog programming language.

A typical reasoning engine may use AI techniques to build presentations (Csinger, Booth and Poole [2]). Reasoning engines may actively gather other information, if it is available. For example, some useful information on a user may be gleaned from using the "finger" protocol if an Internet connection is available.

3 Valhalla

A prototype multimedia system (Valhalla) is currently being developed at the University of British Columbia. The design of Valhalla was guided by two principles: media and platform independence. A distributed architecture was chosen for the prototype implementation. Populating Valhalla's framework are a number of autonomous agents that fall into the following classes: client applications, network-based multimedia servers, and other service providers such as reasoning engines and annotated video databases. Figure 3 illustrates agents within the Valhalla framework.

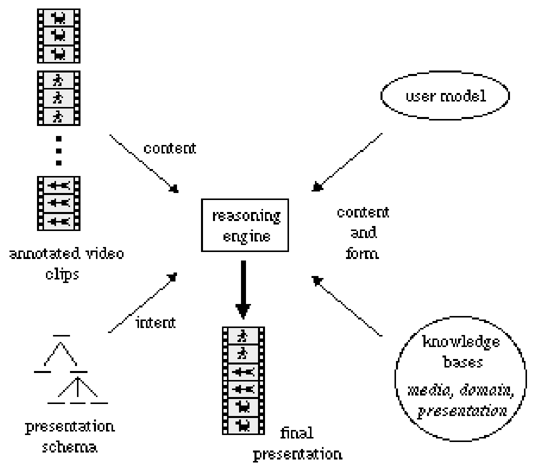

Figure 2: The Intent-based Video Authoring Paradigm

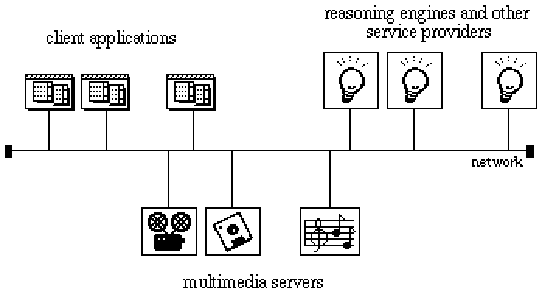

Figure 3: Valhalla's Distributed Architecture

Each class of agent has associated with it a single communications protocol, which allows client applications to transparently communicate with all instantiations of that class. It is this "plug and play" compatibility between members of a class that makes the Valhalla framework flexible and powerful.

One member of each class of application has been currently implemented in the prototype framework. A client application, also named Valhalla, provides a user with intent-based authoring services. This client uses the services of a reasoning engine to generate descriptions of video presentations, and displays the video sequences within the presentations using the services of a distributed multimedia server.

3.1 The Multimedia Server

From the perspective of client applications, the server is a single entity providing virtual VCR-like control over multiple media sources. The server has the ability to simultaneously operate in two modes: in local mode, media sources are conventional analog devices, whose output may be routed to a number of available display devices using an RS-232 controlled switch. In remote mode, digital media is transmitted over the network to the client application from one or more remote sites, and is controlled using media playback applications present on the client's host.

The server is implemented as a series of increasingly abstract application programming interfaces (APIs). Each interface can be directly accessed by a client application, but typical operation of the server would only involve access from the highest (and most abstract) level. A higher level interface uses the services of a lower level in order to provide its own services. Figure 4 illustrates the relationship between these interfaces and the various components of the server.

The lowest level, called the device level, is accessible only through TCP socket communications and is composed of a series of device drivers. There is one device driver for each physical device, and one driver responsible for dispensing a class of digital data.

The next most abstract level (the class level) is accessible to client applications via an API to a library that directly communicates with the server on behalf of the client. This intermediate level provides completely separate interfaces for each class of device. Devices of the same class share characteristics particular to that class. Providing a separate interface for each class allows client applications to take advantage of these particular characteristics. Supported classes currently include digital video, digital audio, random-access analog video (optical video disc), and tape-based analog video. Future extensions to the server will support other device classes, such as text, MIDI or digital audio.

The top level, called the virtual-VCR level, is again accessible via an API and provides an interface that client applications can use to control any device. Characteristics of different device classes have been abstracted away in order to provide a single VCR-like interface.

Using an API to communicate with the more abstract levels of the server is advantageous for a number of reasons. The inclusion of a library into the client executable facilitates the distributed architecture of the server; each library can be considered to be an agent of the server executing on the client's host machine. The presence of the server on the client's machine allows the server to directly control applications on the client's host. Such applications would be used for the playback and manipulation of digital media. Secondly, the API itself has been designed to provide a uniform method of interacting with multiple media sources and formats, allowing the differences between classes of media to be partially abstracted away. The following is an excerpt from the C programming language API specification of the virtual-VCR level of the multimedia server:

```
resultType VVCR_Play(char *deviceName);
resultType VVCR_PlayFromTo(char *deviceName, int from,
                           int to);
resultType VVCR_Stop(char *deviceName);
resultType VVCR_Still(char *deviceName);
resultType VVCR_FF(char *deviceName);
resultType VVCR_Reverse(char *deviceName);
resultType VVCR_SetPosition(char *deviceName, int pos);
```

3.2 The Valhalla Client Application

The name of the client application to be examined in this section coincides with that of the multimedia framework - Valhalla. This client application currently serves as a hypermedia brochure (or hyperbrochure) designed to present information about the UBC Department of Computer Science to visiting students, staff, and faculty members.

Because the hyperbrochure was designed to present information to a wide variety of audiences, it becomes reasonable to adopt intent-based authoring as a method of constructing hyper-presentations customized to the needs of individual viewers. Since the purpose of the hyperbrochure is to present information, the communicational intent contained in the hyperbrochure's schema is that of "inform".

Figure 5 depicts the main interface to the hyperbrochure, which is currently implemented on a NeXT

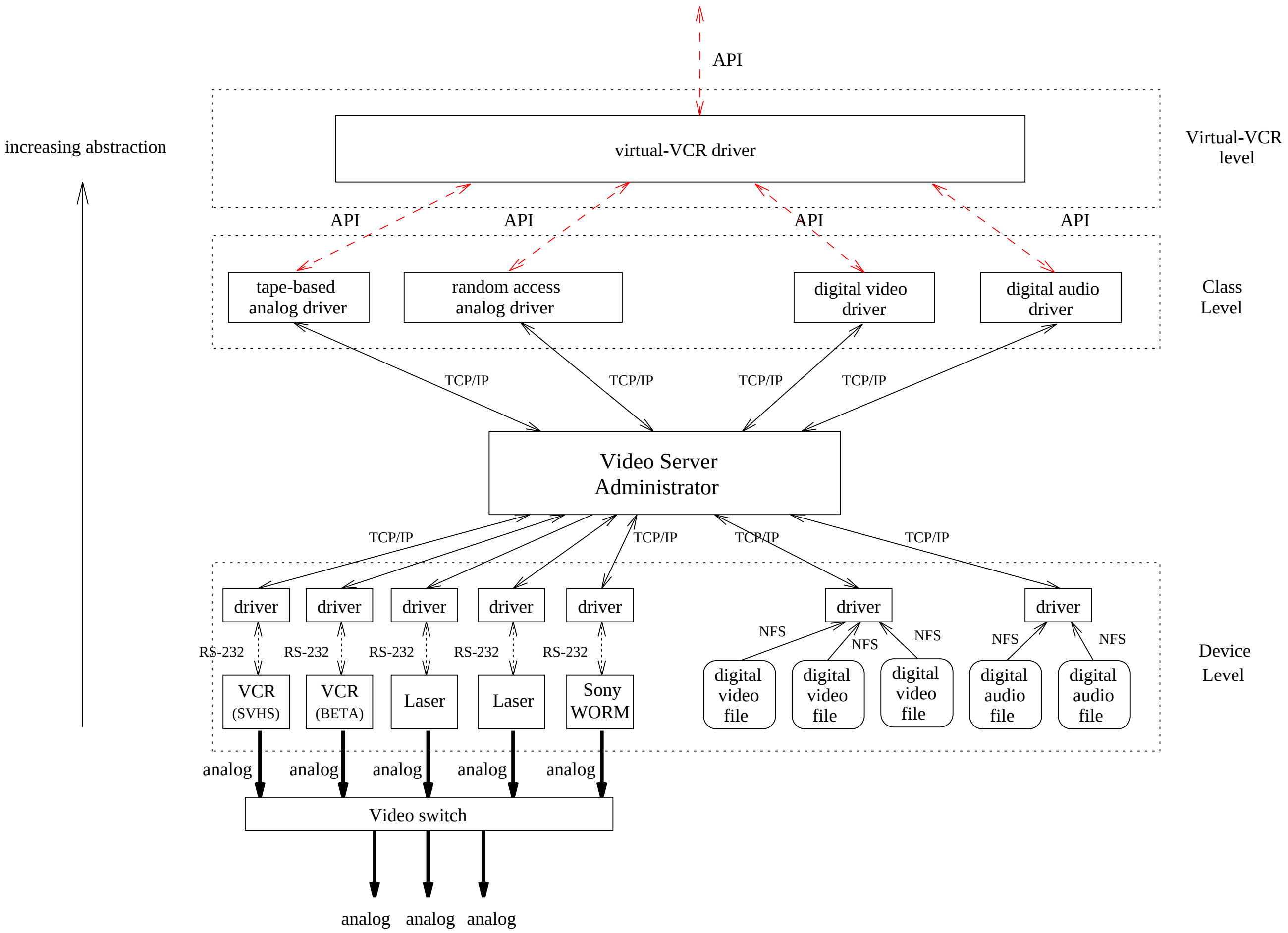

Figure 4: The hierarchy of interfaces to the video server

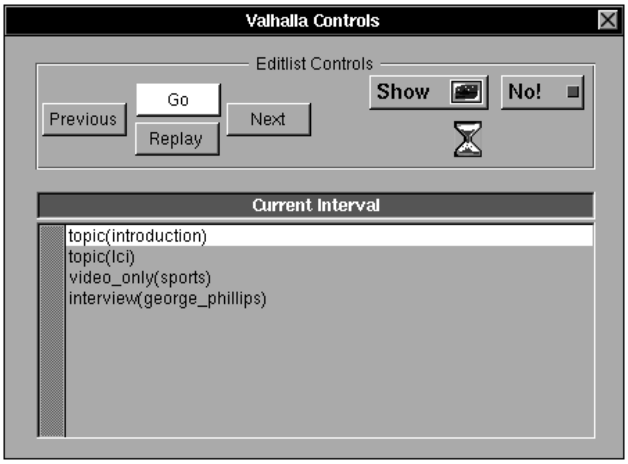

Figure 5: The hyperbrochure main interface

Figure 6: A frame from a video clip

cube. The Show and No! buttons provide a method for the user to communicate with the reasoning engine (although the engine is, for all intents and purposes, invisible to the user.) Pressing the Show button causes the engine to generate an editlist, which is an ordered sequence of media clips that describe a presentation. Pressing the No! button allows the user to express dissatisfaction with the provided presentation, causing the reasoning engine to supply the "next best" editlist.

The services of the multimedia server are used to display clips contained in an editlist. Currently, all media associated with the hyperbrochure is in the form of (synchronized) analog video and audio stored on CAV laserdiscs. Analog signals from two laserdisc players are routed to a video digitizer board within the NeXT computer, and the resulting digital video is displayed in a window on the NeXT's display. An example frame from a campus sporting event is shown in Figure 6.

We have not described how the reasoning engine acquires its assumptions about the hyperbrochure user. The initial user model is made up of hypotheses based on prior probabilities injected into the database by a knowledge engineer; subsequent interaction by the user causes an update of these hypotheses. As well, when the hyperbrochure client application starts a new session, it sends the user's name and host address to the reasoner. The reasoner then makes use of the Internet "finger" protocol (as well as other available local sources of information) to learn as much as it can about this user, updating its model accordingly.

Hypotheses can, in principle, be incorrect. Depending on how much importance the reasoning engine bestows on a particular hypothesis, the consequences of it being wrong (in terms of the appropriateness of generated presentations) could vary wildly. In order to compensate for this, and to give the user some indirect control over the content of the presentation, an appropriate subset of the user model is exported to the hyperbrochure and is made available for correction. Figure 7 contains an example view of a user model. The selection of an appropriate subset of the user model is a complex process; a cursory inspection of the problem suggests that those assumptions should be selected to which the presentation is most sensitive. Defining a suitable metric of sensitivity is not an easy task, however (refer to Csinger *et al.* [2]).

The pertinent elements of the user model selected by the reasoning engine are transferred to the hyperbrochure application in an abstract form. Instead of specifying a particular GUI "widget", a particular hypothesis may be specified as an element of a (finite) discrete set, as possessing a value within a particular range, or as having a boolean value. It is left to the hyperbrochure to determine an applicable "widget" to display this to the user. This method of providing abstract, indirect control over the reasoning process adds to the flexibility of the framework.

3.3 Benefits of the Valhalla Framework

Much like separating intent from content allows the delayed selection and binding of the content and form

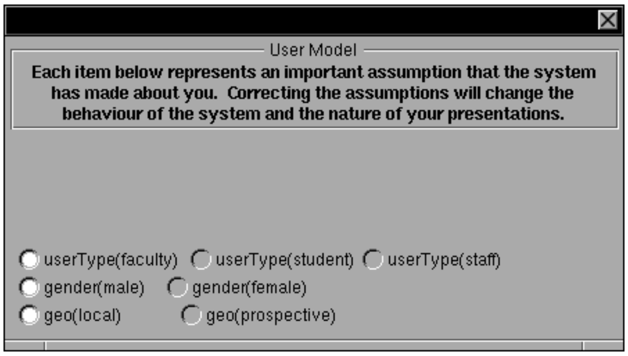

Figure 7: The hyperbrochure user model window

of authored documents, the distributed architecture and enforced compatibility between classes of applications in the Valhalla framework permit a single client application to be used for many purposes. In our example, the hyperbrochure client application, changing the schema provided to the reasoning engine and introducing different media clips into the multimedia server's collection could alter the nature of the client. For example, if the intent is changed to that of entertainment, and if presentation details are specific to the point of nearly being a script, the same client application used in the hyperbrochure could possibly be used as an interactive storytelling device.

The client applications are not the only agents whose nature can be changed. Reasoning engines, for example, may be thought of as modular elements that can be replaced depending on desired characteristics. It is conceivable to think of an author selecting from among a host of different reasoning engines, each with a unique "personality".

3.4 Future Development

A number of extensions to the hyperbrochure client are being planned. In addition to directly gaining knowledge via the user model window, details of the user's interaction with a presentation will be provided to the reasoning engine. For instance, if a user uses the navigation buttons in order to skip the viewing of the remainder of a particular clip, it may be deduced that the user doesn't have any interest in the contents of that clip and the user model can be updated accordingly.

As was previously mentioned, items in the user model window are chosen based on their degree of sensitivity. Sensitivity analysis is defined as the act of determining how much a presentation will be changed when the user modifies a given set of assumptions in the user model. Given a set of user model items, sensitivity analysis currently provides a quantitative measure of these items. This measure can then be used by the client application to present the user model items in a reasonable, sorted order.

Other visual clues from the reasoning engine can also be considered. The current reasoning engine associates a degree-of-belief metric with each assumption in order to judge how much weight to give it. Linking the degree-of-belief metric with colour (for instance) might help to persuade the user to notice and correct faulty assumptions. As an example, the client application might choose to present all assumptions with a low degree of belief in bright red, thereby providing the suggestion of uncertainty or danger.

One of the goals of the intent-based authoring paradigm was to save time and effort by reducing the impact of the temporal nature of the video medium. However, viewers must still watch entire presentations in order to gauge their relevancy and provide useful feedback to the reasoning engine. If the temporal portions of presentations could be summarized in a non-temporal format, the viewer may be able to form an opinion of the presentation in a more timely manner. One possibility is to construct a graphical representation of the presentation and use a fisheye view (Noik [5]) of the representation in order to highlight the relevant features. If the user is in the process of browsing multiple presentations, the differences between presentations may be candidates for relevancy.

4 Conclusions

Overcoming the well-known liabilities of the traditional authoring paradigm provided the motivation for an intent-based authoring paradigm. By requiring the encapsulation of intent and the ability to delay the binding of the content and form of an authored document, the intent-based authoring paradigm itself motivated the development of a more flexible multimedia system. The Valhalla distributed multimedia architecture has a number of key characteristics (media and platform independence and a distributed nature) that satisfy the requirements of intent-based authoring systems. This architecture was used to implement an intent-based hypermedia application, namely a departmental hyperbrochure.

References

[1] Andrew Csinger and Kellogg S. Booth. Reasoning about Video: Knowledge-based Transcription and Presentation. In Jay F. Nunamaker and Ralph H. Sprague, editors, *27th Annual Hawaii International Conference on System Sciences*, volume III: Information Systems: Decision Support and Knowledge-based Systems, pages 599–608, Maui, HI, January 1994.

[2] Andrew Csinger, Kellogg S. Booth, and David Poole. AI Meets Authoring: User Models for Intelligent Multimedia. *Artificial Intelligence Review*, 8, 1994.

[3] Peter Karp and Steven Feiner. Issues in the automated generation of animated presentations. In *Proceedings Graphics Interface*, pages 39–48, Halifax, May 1990.

[4] Alfred Kobsa. User modelling: Recent work, prospects and hazards. In *Proceedings of the Workshop on User Adapted Interaction*, Bari, Italy, May 1992. Also available as a June 1992 Technical Report from Universität Konstanz Informationswissenschaft.

[5] Emanuel G. Noik. Layout-independent fisheye views of nested graphs. In *Proceedings IEEE/CS Symposium on Visual Languages*, Bergen, Norway, August 1993.

[6] Dorée Duncan Seligmann and Steven Feiner. Automated generation of intent-based 3d illustrations. *Computer Graphics*, 25(4):123–132, July 1991. Proceedings of SIGGRAPH '91 (Las Vegas, Nevada, July 28-August 2, 1991).

[7] Wolfgang Wahlster and Alfred Kobsa. *User Models in Dialog Systems*. Springer-Verlag, 1990.