Modularization of Multimodal Interaction Specifications

Matthias Denecke, Kohji Dohsaka, Mikio Nakano

Communication Sciences Laboratories, NTT
NTT Corporation
Morinosato Wakamiya 3-1
Atsugi, Kanagawa 243-0198, Japan
denecke@cslab.kecl.ntt.co.jp

Abstract

In this paper, we discuss potential advantages of, and current impediments to, modular specifications of multimodal interfaces.

1. Introduction

In this paper, we address modularization of interaction management. Specification of interaction management is intended to be understood according to section 6 in [1].

The recurring theme of modularity in software engineering is to "expose things that remain constant, hide things that change". The goal is to manage complexity. We argue that there is a need to manage complexity in multimodal systems, due to the many different disciplines required to design, implement and deploy a successful multimodal system.

2. Background

2.1. Examples for Modularity

2.1.1. Component Based System Design

Modularity has long been a concern in software engineering. In practice, there are many different approaches to encapsulate functionality provided by components. We will not discuss in detail these approaches but just mention two well-known successful approaches, namely *Enterprise Java Beans*, part of the Java platform and the *Component Object Model (COM)*, mostly used in Microsoft applications. A newly emerging paradigm are *web services* which allow modularization of an applicaton across a network.

2.1.2. Dialogue Objects

Recently, several vendors of voice controlled systems have developed modular approaches to dialogue management. Put simply, what most of the different approaches to dialogue objects do is to take standard techniques as established in software engineering, to the voice application level. Following this approach, an application is divided into several components. For instance, an e commerce travel application might contain a component responsible to acquire dates, another one to acquire monetary amounts and so on. Depending on the application flow, one component is entered, the interaction remains under local control of this component until either the goal of that component is reached or the interaction is aborted, and moves on to the next component. An important characteristic of this kind of modularity is that the components act as black boxes and can only be influenced from the outside through interfaces specified at compile time. This is the same as is known from software engineering components.

2.1.3. Cascading Style Sheets

Cascading Style Sheets (CSS for short) provide a kind of modularity that is different from the one described in the previous section. More specifically, CSS allow to specify the visual appearance of XHTML markup separately from the web page. The CSS provides a set of rules how XHTML elements should appear in the web browser. The interface between XHTML and CSS is provided by *selectors* on the CSS side which specify alternatively to which type (such as `H1, H2`,...) or to which class of XHTML node a rule should be applied. A class can be assigned to an XHTML node by way of the `class` attribute in case nodes of the same type are to appear differently. The interesting aspect of this sort of modularity is that a style sheet can cut across a whole web page (pr web site, for that matter) once a set of selectors is agreed upon.

3. Related Standards

3.1. VoiceXML and SALT

SALT and VoiceXML both allow encapsulation of speech and TTS engines, thus enabling component based system design (to a degree) as described in section 2.1.1. This allows development of interactive voice response applications to be independent from the development of voice browsers.

In addition, VoiceXML allows the specification of simple voice based interactions by supporting so-called form filling dialogues. To this end, a VoiceXML compliant component implements a generic dialogue management algorithm. An application designer wanting to take advantage of this algorithm specifies the domain dependent bits and pieces of the application, such as which slots to fill and how to prompt for fillers, and leaves control to the VoiceXML component as appropriate to acquire the information from the user.

The interesting aspect of this approach is that it cleanly separates the dialogue logic (the dialogue management algorithm) from its domain dependent presentation (the prompts and slots to fill). Three questions naturally arise at this point:

1. Is it desirable to have such generic algorithms?

2. Is the provision of generic dialogue algorithms generic enough to develop any spoken dialogue application?

3. In the context of this workshop, if such an approach is desirable, can it be extended to multimodal dialogues?

The developers of the VoiceXML standard obviously answered the first question affirmatively, while the developers of the SALT standard answered that question negatively, emphasizing the simplicity of their markup language. Another interesting aspect of the VoiceXML approach is that the separation of generic dialogue management and application specific content does not relieve an application designer from understanding the dialogue algorithm. For development purposes, this knowledge is still required.

Second, the VoiceXML dialogue management algorithm is not the only way to implement dialogues in VoiceXML. Rather, VoiceXML application designers are free to explicitly specify (or dynamically generate on the fly, depending on the application logic) VoiceXML pages in which dialogue logic is encoded, for example, in VoiceXML compliant `<if>` `<else>` markup. This can be seen as a statement that generic algorithms are not sufficient, at least at the moment.

Before addressing multimodal concerns, it might be instructional to consider standards for conventional web pages. The XHTML, CSS and JAVASCRIPT provide a powerful combination of declarative and procedural specifications to graphical user interfaces. Simply put, XHTML and CSS together provide a representation of the content and the presentation, respectively. The interface that links the XHTML markup from the specification of its presentation in CSS is introduced by the `class` attribute. It is noteworthy that the XHTML specifications and the style sheets can be entirely orthogonal. In other words, once XHTML pages and style sheets agree on a set of `class` labels (and their meaning) XHTML pages and style sheets can be freely exchanged, since the style sheet specifications cut across the XHTML markup. This allows either a uniform user experience for different pages, or an individualized presentation of the same content, for instance, to address special user needs.

4. Challenges for Multimodal Systems

4.1. Examples

4.1.1. Channel Management

One of the most distinctive features of multimodal systems compared to other systems is their need to manage deficient input components. Suhm [2] showed that the use of input channels with different modalities leads to decreased task completion time and increased task completion rate as the input channels complement each other. In the same vein, Vo and Wood [3], among others, showed how the simultaneous use of multiple input channels leads to synergetic effects that decrease the (combined) concept error rate.

In human interaction research, Bhavnani and John [4] showed how users acquire interaction patterns Several studies show that despite experience, many users with basic command knowledge do not progress to an efficient use of complex computer applications. These studies suggest that knowledge of tasks and knowledge of tools are insufficient to lead users to become efficient. Assuming that these findings generalize to multimodal applications, the efficient use of multimodal applications is hampered by the fact that potential problem sources for perception based user interfaces are hidden from the user. This includes changing environment conditions (changing lighting conditions for camera based applications, changing noise levels for speech based applications, etc), but also insufficient working components, for example, due to lack of grammar coverage.

The corollary from the above is that perception based interfaces, if they are intended to be used efficiently, need to be actively managed. Based on the sensory input, the system state, possibly a environmental model and an estimate

of the users' intention, multimodal applications need to be capable of determining efficient user guidance. In particular, this *channel management* is necessary as deficiencies in the perception based input components are invisible to the user.

An example for such an active interaction management might be expressed in a rule of the following content: *If two subsequent uses of the same input channel result in rejected or otherwise failed interactions, the system should avtively suggest a more appropriate input channel to the user, if available.* We do not focus here on how the rules are to be expressed, we just assume that they can be formalized in one way or another.

Having determined a set of such interaction rules, it is desirable to apply them to an entire application, or even several applications. At this point, there is a need for a sort of modularity that allows crosscutting across different modules in much the same way style sheets can cut across multiple XHTML pages and modify their appearance. This sort of modularity is often not provided by component-based approaches.

4.1.2. Affective Interfaces

Once multimodal interfaces are capabale of determining the emotional state of their users [5], it is desirable to react to the users' state appropriately. While the previous section addressed the issue of modular management of deficient perception based input components, but a point along the same lines can be made for emotional and character based interfaces. As it is desirable to separate channel management from the original task, it is desirable all the same to specify the affective aspects of users interfaces separately from the original task.

4.1.3. Character based Interfaces and Impersonators

Similar to specifying interactions based on the users' emotional state, there are applications to specify interactions based on the users' knowledge and comprehension. For example, in interactive tutoring systems, one distiguishes between socratic and didactic interactions with the students. Similarly to channel manament, it is desirable to express character management in a form that is orthogonal to the specification of the content.

It probably did not go unnoticed that the above proposals of separating the task specification from the interaction specification has been motivated by the separation of content and presentation in XHTML and CSS. As in XHTML and CSS, an implementation of these suggestions, if possible, would lead to systems that can be composed

4.1.4. Summary

The examples in the previous section illustrate a need to separate the content of multimodal interaction from the style in which the interaction is carried out. We argue that this separation is different from the separation of dialogue management algorithms and application specific content as exemplified by VoiceXML (see questions 1 and 2 above). It is also different from the application logic level modularization as implemented by the various approaches to dialogue management (see section 2). Rather, it is more in line with the separation of content and presentation as exemplified by XHTML and CSS.

4.2. Potential Impediements

In the previous sections, the analogy between XHTML and CSS and multimodal interface specifications has been stretched a bit. There are several points in which the analogy breaks down.

In the previous section, we made the assumption that conversational strategies are orthogonal to specification of task models in the same way as XHTML and CSS are orthogonal. It needs to be determined to which degree this remains true in practice. Even if not always totally orthogonal, we are convinced that there is enough orthogonality so that a separate specification is desirable. Of course, this does not preclude explicit links where necessary.

Another problem is related to the combination of separate specifications into a coherent system behavior. Here, it is helpful again to look at the example of XHTML and CSS: conceptually, the style sheet can be considered as a complementary specification that fills in missing information in the XHTML page to yield a more fully specified XHTML page. Similarly, different interaction specifications can increase the informational content of declarative representations.

The third problem, finally, is related to developer effort. While the separation of specifications allows the reuse of tested, well tested components, the effect of combining these different specifications needs to be such that it either can be easily understood by developers *even if they don't understand the specifications*, or needs to be able to be evaluted by means of automated tools.

5. A Proposal for Modularization

Given that the functionality of multimodal dialogue managers varies depending on the task, it is unlikely that an at-

tempt to encapsulate functionality in an API similar to SAPI will be successful. In this paper, motivated by the discussion on HTML and CSS, we would therefore propose an alternative. More specifically, we propose to separate content from style (as in HTML and CSS) in separate specifications. In addition, we provide a third document in which the vocabulary of the interface is specified. In the HTML / CSS example, this corresponds to a list of the tag types and class labels for which CSS rules are specified. To summarize, we need three things:

1. **A Content Specification** which corresponds to the role of HTML in the above example,

2. **An Interface Declaration** which corresponds to the role of tags and class labels

3. **An Interaction Specification** which corresponds to the role of CSS

In the following, we discuss each specification in turn.

5.1. Content Specification

We assume that the content of the interactions be represented in a representation along the lines of EMMA. Put simply, EMMA can be considered a semantic representation (such as RDF) that is enhanced with multimodal annotations, such as confidence scores from recognition engines or time stamps. In contrast to EMMA, however, we propose an abstraction mechanism that assigns symbolic labels to numeric confidence scores. This is done along the lines of *facets*. See [6] for examples of how semantic representations can be annotated with temporal, spatial and input channel information using facets. Facets are symbolic multimodal annotations that, if chosen carefully, will allow unification and subsumption defined on the original representations to be extended to the annotated representations. The primary purpose of the facets is to abstract away information specific to recognition engines and to allow the vocabulary of the facets to be expressed in the interface declaration.

5.2. Interface Declaration

The purpose of the interface declaration is to introduce a common vocabulary by means of which the interaction specification can be expressed. The vocabulary contains entries for the facets, for example of the form *confidence: low, average, high* with which the abstraction mechanisms annotate the multimodal representations.

Since the actions of the interaction manager do not only depend on the input, but also on the internal state, the interface declaration needs to introduce vocabulary to describe the dialogue state as well. This description is referred to as *abstract dialogue state*. The abstract dialogue state can be seen as a collection of features that describe the dialogue state. In particular, the abstract dialogue state may contain aggregations of facets over time, for example, the number of times a part in a representation has been labeled with a confidence *low*.

5.3. Interaction Specification

The interaction specification defines how the interaction manager generates the output depending on the abstract dialogue state and the facets of the incoming representations. The important aspect is that the interaction specification be expressed only in terms of the vocabulary defined in the interface declaration. If a rule-based approach is adopted, it is also possible to have the interaction specification be generated by some form of statistical learning algorithm.

Given the vocabulary the interface declaration introduced, one can then define rules such as: *if speech has been used more than once in the past, and the current input contains at least two* confidence = low *annotations for information provided through the speech channel, then suggest not to use the speech channel for the next turn*. The first condition makes use of the abstract dialogue state, while the second conditions makes use of facets in the current input representation. As multimodal systems become more complex, this sort of specification becomes costly to develop; and thus is an interesting candidate for reusability.

In [7], an approach for (unimodal) spoken dialogue management, similar to the one outlined here, has been described.

6. Discussion

In this paper, we discussed modular specifications for multimodal interfaces. We argued for the necessity of modularization, due to (1) the complexity of multimodal interface specification and (2) the need for developer team members with different background. Modularization of specifications can alleviate both obstacles to a certain degree. We argued that existing approaches to modularity, as widely used in software engineering and in some voice applications, are not sufficient, because these approaches do not allow interaction specifications to cut across workflow specifications. We illustrated with the help of a set of examples how a complementary specifications along the lines of XHTML and CSS

can be desirable and outlined some potential problems with such an approach.

In this paper, we leave open the question whether any multimodal interaction standard should provide generic interaction management algorithms (as in VoiceXML) or not (as in SALT). Rather, we argue that, no matter which approach is followed, there is a need for modularization on the interaction management level, similar to the separation of content and presentation as exemplified by XHTM and CSS. Such a standard would allow a modularization in such a way that the ”how” aspect and the ”what” aspect of a multimodal application can be exchanged.

7. Acknowledgements

We would like to thank Shoji Makino and all members of the Dialogue Understanding Research Group for support and helpful discussions.

8. References

[1] J. A. Larson and T. V. Raman and D. Raggett. 2003. W3C Multimodal Interaction Framework, W3C Note 06, May 2003. www://www.w3c.org/TR/mmi-framework

[2] B. Suhm and B. Myers and A.H. Waibel. 1999. Model-Based and Empirical Evaluation of Multimodal Interactive Error Correction. In *Proceedings of the CHI 99, Pittsburgh, PA, USA.*.

[3] M. T. Vo and C. Wood. 1996. Building an Application Framework for Speech and Pen Input Integration in Multimodal Learning Interfaces. In *Proceedings of the International Conference on Acoustics, Signal and Speech Processing*.

[4] S. K. Bhavnani and B. E. John. 2000. The Strategic Use of Complex Computer Systems. *Human Computer Interaction*. 15:107–137.

[5] R. Picard. 1997. Affective Computing. The MIT Press.

[6] M. Denecke and J .Yang. 2000. Partial Information in Multimodal Dialogue Systems In *Proceedings of the International Conference on Multimodal Interfaces*. Available at http://www.is.cs.cmu.edu

[7] M. Denecke. 2003. Policies and Procedures for Spoken Dialogue Systems In *Proceedings of the EACL 2003 Workshop on Dialogue Systems, Interaction, Adaptation and Styles of Management*. Available at http://www.is.cs.cmu.edu