

Menu Hierarchy Generation Based on Syntactic Dependency Structures in Item Descriptions

Yukio Horiguchi, Shinsu An, Tetsuo Sawaragi, and Hiroaki Nakanishi

Dept. of Mechanical Engineering and Science, Kyoto University, Kyoto, Japan
horiguchi@me.kyoto-u.ac.jp

Abstract. The present paper proposes a procedural design method which makes use of dependency structures underlying menu item descriptions in order to generate well-structured and easily-learned menu hierarchies. A dependency structure captures syntactic relations among conceptual units that constitute a natural language description of what function the corresponding menu item stands for. The proposed method classifies computerized system functions after dependency structure prototypes and then serializes variable elements to be specified in each function class after phrase structure analysis. Its clear and consistent policy provides generated menu systems with high communicability to users. The effectiveness of the method is empirically investigated.

Keywords: Menu, Conceptual Dependency, Phrase structure analysis, Interface design, Human-Computer-Interaction, Usability.

1 Introduction

Organization of items in a hierarchy impacts the menu system usability [1]. Menus that are not designed consistent with the user's way of thinking would cause breakdowns in user-system communication [2]. They can easily misdirect and confuse users searching for items in the hierarchy. Regarding this issue, we have been studying menu interface design techniques to generate menu hierarchies based on "dependency structures" [3]. A dependency structure captures syntactic relations among conceptual units that constitute a natural language description of what function the corresponding menu item stands for.

According to Sperber and Wilson [4], a hearer of an utterance forms anticipatory hypotheses about the overall syntactic structure of that utterance on the basis of what s/he has already heard. Such an "anticipatory syntactic hypothesis" helps the hearer resolve ambiguities and referential indeterminacies contained in the utterance allowing her/him to access some of the encoded concepts before other utterances. The same can apply to the menu-based human-computer interaction. Menus structured for users to posit anticipatory hypotheses can help them easily develop mental image on the overall structure of the menu hierarchy through interaction.

The present paper proposes a procedural design method which makes use of such a syntactic feature underlying item descriptions in order to generate well-structured and

easily-learned menu hierarchies. The method classifies computerized system functions after dependency structure prototypes and then serializes variable elements to be specified in each function class after phrase structure analysis. Its clear and consistent policy is expected to provide generated menu systems with high communicability to users. The effectiveness of the proposed method is empirically investigated.

2 Dependency Structure

2.1 Conceptual Dependency

Schank has proposed the theory of Conceptual Dependency (CD) as a model of natural language understanding to be used in artificial intelligence systems [5,6]. In the theory, understanding of language is considered as the process of connecting words with certain conceptual constructions that exist in one's memory, which is named "conceptualization". There are two distinct levels of analysis in CD to represent concepts underlying utterances without relation to the language encoding them. One is the sentential level in which utterances of a given language are encoded within a syntactic structure of that language. The other is the conceptual level in which conceptualizations take place. A conceptualization consists of concepts and formal relations between the concepts. This level of analysis is conducted using formal representations of the conceptual base by which two sentences identical in their meaning have a single representation. A conceptual dependency network is the result of the analysis which is given by a linked network of concepts with dependencies between the concepts.

The proposed method extracts dependency structures from menu item descriptions and then visualizes them in diagrams like a CD network. According to CD, a large number of verbs can be rewritten into a small number of "primitive actions" [5,6] like MOVE. Different primitive actions give different structures of dependencies. Based on this idea, conceptual networks out of item descriptions will be classified into a limited number of structural patterns. The proposed method utilizes such patterns to organize menu items into a hierarchy. Bækgaard and Anderson's scheme [7] is employed to analyze typical dependency structures for computerized system functions.

2.2 Interaction Primitives

Bækgaard and Anderson [7] have proposed a description scheme to analyze and grammatically represent various human-related activities dealing with information systems. An interaction primitive is defined as a prototypical action which gives basic forms of interaction and combines other activity elements/participants in a specific way to represent a certain type of activity.

Table 1 presents a partial list of semantic roles of elements involving interaction primitives to be used in the present study. Bækgaard and Anderson have elaborated the following four primitives as the most basic set of actions:

- **SENSE** is the action that comes with three interaction roles of *Experiencer*, *Source*, and *Phenomenon*. It represents a situation in which the *Experiencer* senses the *Phenomenon* as an aspect(s) of the *Source*.
- **MOVE** is the action that comes with roles of *Agent*, *Object*, *Source*, and *Destination*. It represents a situation in which the *Agent* moves the *Object* from the *Source* to the *Destination*.
- **MODIFY** is the action that comes with roles of *Agent* and *Object*, representing a situation in which the *Agent* modifies the state of the *Object*.
- **CONTROL** is the action that comes with two interaction roles of *Agent* and *Object* or *Experiencer*, representing a situation in which the *Agent* requests the *Experiencer* to do something or the *Agent* physically controls the *Object*.

Bækgaard and Anderson [7] have also defined “mediation” in their scheme to represent activities which consists of two or more different interaction primitives. Primitives in a mediated activity share at least one element that is named a mediator. Mediated SENSE, Mediated MOVE, Mediated CONTROL, and Mediated MODIFY are instances of those mediated activities.

Table 1. Definition of roles of interaction elements (extracted from [7])

Role	Definition
Agent	The active participant that initiates and controls the event
Object	The passive participant that is most affected by the event
Experiencer	The participant affected by information about a Phenomenon
Phenomenon	That which is thought, felt, or sensed by the Experiencer
Instrument	The passive participant that enables the event
Location	The spatial boundary of the event
Source	The location from which an object is transported
Destination	The location to which an object is transported

2.3 Dependency Structure Prototypes

Prototypical dependency structures to classify computerized system functions have been acquired from our former study analyzing hundreds of electric appliances functions [3]. Table 2 presents linguistic and graphical expressions of typical dependency structures commonly found in the analyzed function set. Each structure represents a unique interaction pattern in which a certain type of action takes place involving associated roles of elements in order to run a particular type of functions within the device. For example, Structure #1 represents functions that the system moves an object from somewhere (i.e., the source) to another place (i.e., the destination). The proposed method utilizes these function patterns.

3 Menu Hierarchy Generation Method

3.1 Item Classification After Dependency Structure

The proposed method assumes that individual items in a menu hierarchy are given together with natural language descriptions of what functions they stand for. Processing these descriptions, they are classified into either one of the dependency structure prototypes defined in the previous section. After the classification, elements constituent of each structure separate into two types. One is the type of element that has a common value within that structure class while the other has variable values depending on instances. To designate a unique item in the function set, it is required to specify all variables in addition to its dependency structure type.

Table 2. Linguistic and graphical expressions of dependency structure prototypes

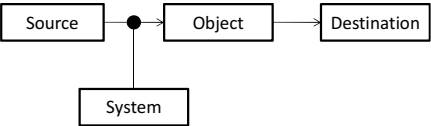
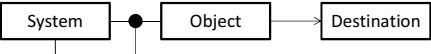
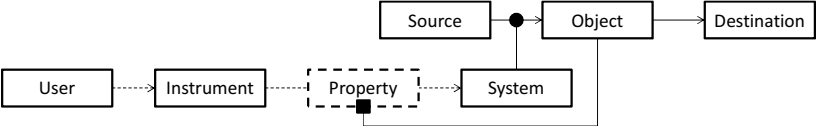
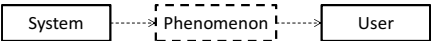

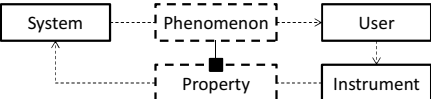
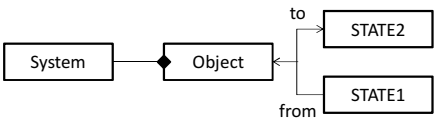
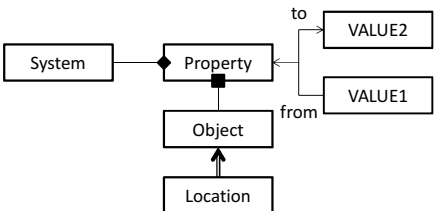
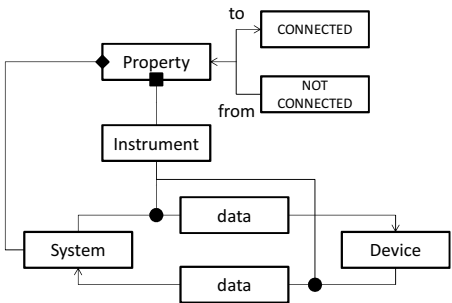
No.	Natural Language Expression
1	<p>The System Moves the Object from the Source to the Destination.</p> 
2	<p>The System Moves the Object from the System to the Destination.</p> 
3	<p>The System Moves the Object from the Source to the Destination. The System Senses the Property of the Object from the User through the Instrument.</p> 
4	<p>The User Senses the Phenomenon from the System.</p> 
5	<p>The User Senses the Phenomenon from the Source mediated by the System.</p> 
6	<p>The User Senses the Phenomenon from the System. The System Senses the Property of the Phenomenon from the User through the Instrument.</p> 

Table 2. (continued)

No.	Natural Language Expression
7	<p>The System Modifies the Object from one state to another.</p> 
8	<p>The System Modifies the Property of the Object at/in the Location from one value to another.</p> 
9	<p>The System Modifies the Property of the Instrument from Not Connected to Connected. The Instrument mediates communication between the System and the Device.</p> 

3.2 Serialization of Variable Elements in Dependencies

A set of rules will be referred to for serializing variable elements in a dependency structure after phrase structure analysis. For an instance of the rules, verbs ought to come first before any other elements. This rule was defined because the choice of verb decides the dependency structure to be focused and what elements to be specified afterward. The choice of dependency structure type followed by a series of selections for its subordinate variables shapes the basic structure of the generated menu hierarchy.

Figure 1 shows a tree diagram representing the phrase structure, i.e., phrase maker, of Structure #6. Alphabets in the diagram denote constituent parts of the sentence (S), i.e., verb (V), noun (N), relative (R), verb phrase (VP), noun phrase (NP),

prepositional phrase (PP), and relative phrase (RP). Numbers in parentheses besides the constituents specify the orders of the variable elements determined by the serialization rules.

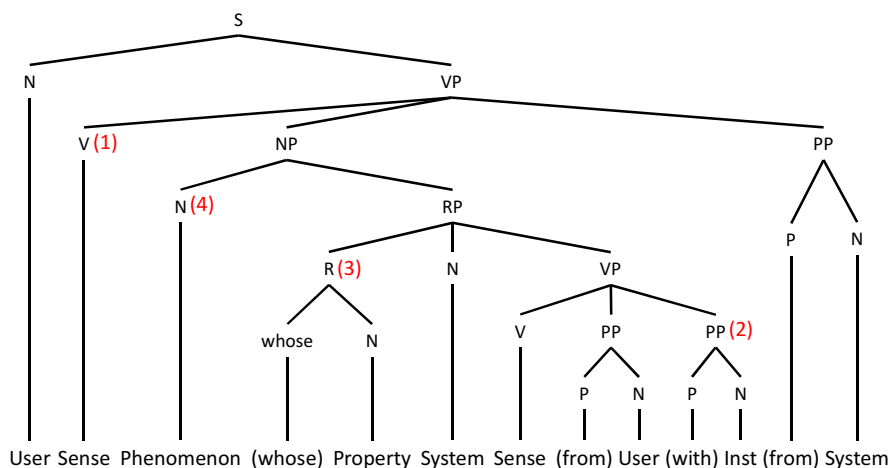


Fig. 1. Phrase marker of Structure #6 (S: sentence, VP: verb phrase, NP: noun phrase, PP: prepositional phrase, RP: relative phrase, V: verb, N: noun, R: relative)

3.3 Overall Procedure

The overall procedure to generate a menu hierarchy out of a set of menu items is defined as below:

1. Prepare a list of menu items of interest with their natural language descriptions.
2. Classify items into either one of the dependency structure types, filling in the structure's element slots with the corresponding words/phrases constituting their descriptions.
3. Sort out items in each structure to identify variable elements.
4. Arrange variables of each structure in sequence after the serialization rules.
5. Generate a menu hierarchy as follows.
 - (a) Determine one concrete verb per dependency structure to represent the set of items in each structure class. The resultant verbs will be used for item labels at the top level of the menu hierarchy.
 - (b) Items at each level below the top menu give options for a variable element of the corresponding structure class. The selection sequence conforms to the order of variables determined by Step 4.

Menu hierarchies resulting from the above procedure ask users to choose a verb first to specify a certain dependency type of functions. The users are then to fill in "blanks" in turn that correspond to variable elements in those functions. This principle gives consistency in the order of choices within the same type of functions.

In addition, interactions to be developed between user and menu system share the higher-level context even across different interaction types. The selection of action precedes selections of objects as well as modifiers. These characteristics are expected to facilitate users' development and application of their mental models on the usage of the menu system.

4 Experiment

An experiment was conducted to validate the effectiveness of the proposed method.

4.1 Setup and Procedure

A DVD recorder was employed as the target of operation whose menu interface was emulated on a personal computer (Figure 2). Participants in the experiment explored a menu hierarchy searching for particular items that could give solutions to their assigned tasks.

In the experiment, a menu hierarchy generated by the proposed method was compared to that resulting from alterations of its structure. The latter hierarchy was created by altering the orders of variable elements in the dependency structures in a random manner. Figure 3 shows a part of the menu hierarchy generated by the proposed method.

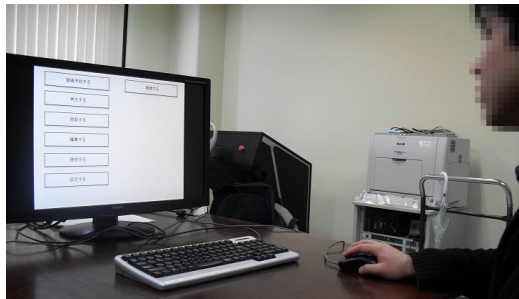


Fig. 2. Experimental setup

The following eight search tasks were given to participants:

- **Task 1:** Play back an audio file stored in the SD card.
- **Task 2:** Change the broadcast category for a programed recording onto the HDD to the Terrestrial Digital Broadcasting.
- **Task 3:** Search TV programs in which actor A will show up.
- **Task 4:** Import pictures stored in the SD card onto the HDD.
- **Task 5:** Program a timer recording of program A onto the HDD using the Electronic Program Guide.

- **Task 6:** Connect the recorder to the video intercom.
- **Task 7:** Dub a video stored on the HDD to the DVD.
- **Task 8:** Change the classification tag of a video stored in the DVD to “Drama”.

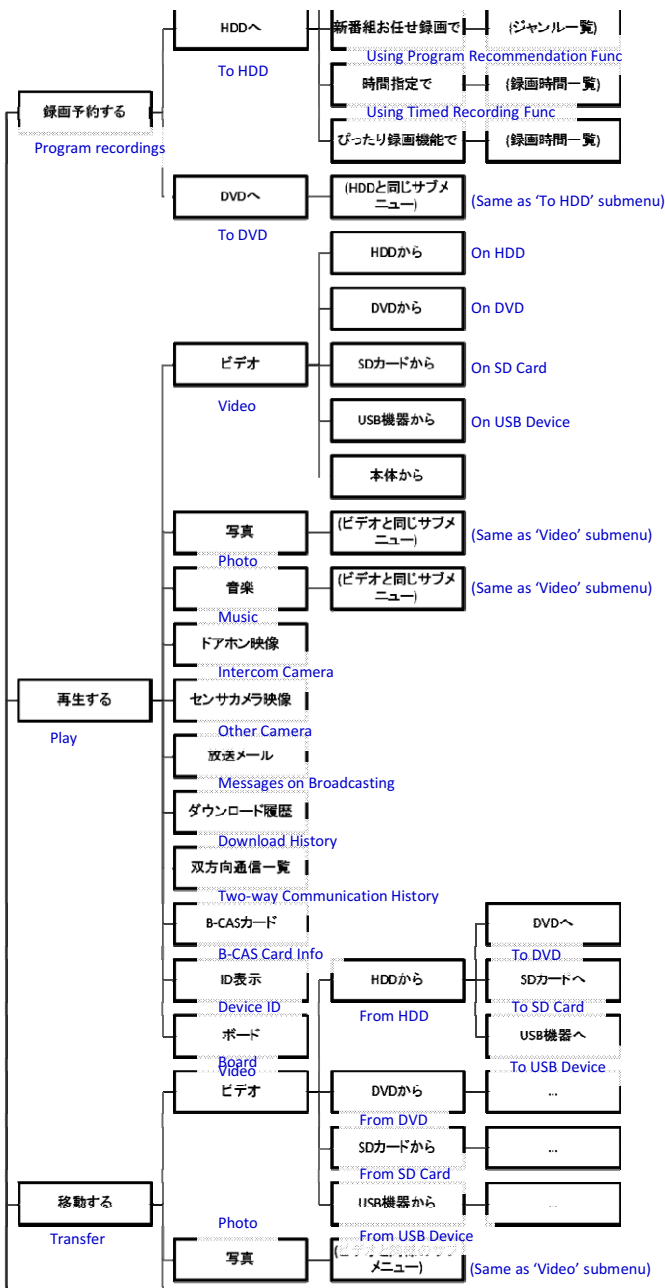


Fig. 3. Menu hierarchy generated by the proposed method

This task set was designed consisting of two blocks, one of which is from Task 1 to 4, and the other of which is from Task 5 to 8. These two blocks share same types of functions for solutions to their individual tasks. For instance, items that give solutions to Task 4 and 7 and those to Task 2 and 8, belong to Structure #1 and #8, respectively. Comparing these pairs in task performance gives data to look at effects of the previous search experiences onto the next search opportunities for menu items unsearched.

Twenty participants were recruited from a pool of faculty members, undergraduates, and graduate students at the department of mechanical engineering in Kyoto University. All of them gave informed consent to participation in this study. The study was designed as a between-subjects experiment. The participants separated into two groups of ten for the two different menu hierarchy conditions. In addition, in order to balance the difficulty between the two task blocks, the ten for each condition divided into two groups of five. Five participants performed Task 1 through 4 for the first half of the experiment and then went on Task 5 to 8 for the second half, while the other five did in the reverse order.

4.2 Result

Figure 4 presents (a) task time and (b) number of “back” button uses per task, comparing the first and the second half of the experiment as well as the two different menu hierarchy conditions. Whiskers represent standard deviations. Student’s *t*-test confirmed significant differences between the two menu hierarchies in both of the two performance metrics in the second half ($p < 0.05$). It was also observed that the proposed method group’s variations in them became much smaller in the second half, different from the other menu hierarchy group.

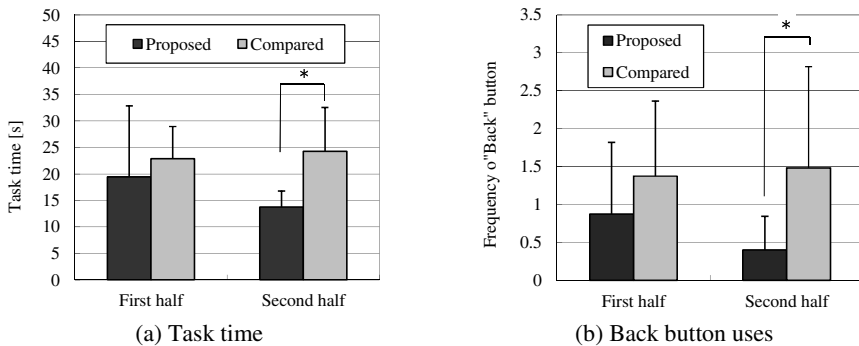


Fig. 4. Experimental result (* $p < 0.05$)

These results indicate that the participants in the proposed method group completed search tasks in the second half more quickly and accurately based on what they had learnt from their experience with tasks in the first half. The proposed method proves to generate menu hierarchies that can benefit users as it allows them to develop solid

mental images for navigating in the hierarchies. The structural regularity embedded in the menu hierarchies enable users to learn paths to items they have never experienced with.

5 Conclusion

This paper has proposed a procedural design method to generate a menu hierarchy based on dependency structures shared among menu item descriptions. A dependency structure captures syntactic relations among elements involving a set of interactions to realize a particular system function. This property gives generated menu hierarchies structural consistency so that, through interaction with them, users can easily develop solid mental images for navigating in the hierarchies.

A comparative experiment was conducted to validate the effectiveness of the proposed method. The method proved to generate menu hierarchies that enable the users to learn paths to items they have never experienced with from past experiences in search for “functionally” similar items.

References

1. Norman, K.L.: Better Design of Menu Selection Systems Through Cognitive Psychology and Human Factors. *Human Factors* 50(3), 556–559 (2008)
2. Horiguchi, Y., Nakanishi, H., Sawaragi, T., Kuroda, Y.: Analysis of Breakdowns in Menu-Based Interaction Based on Information Scent Model. In: Jacko, J.A. (ed.) *Human-Computer Interaction, HCII 2009, Part I. LNCS*, vol. 5610, pp. 438–445. Springer, Heidelberg (2009)
3. Horiguchi, Y., Gejo, W., Sawaragi, T., Nakanishi, H.: Menu System Design Based on Conceptual Dependency Structures between Functional Elements. In: *Proceedings on 2011 IEEE/SICE International Symposium on System Integration*, pp. 262–266 (2011)
4. Sperber, D., Wilson, D.: *Relevance: Communication and Cognition*, 2nd edn. Blackwell Publishers (1995)
5. Schank, R.C.: Conceptual Dependency: A Theory of Natural Language Understanding. *Cognitive Psychology* 3, 552–631 (1972)
6. Schank, R.C.: *Conceptual Information Processing*. North-Holland, Amsterdam (1975)
7. Bækgaard, L., Anderson, P.B.: *Using Interaction Scenarios to Model Information Systems*. Working Paper I-2008-04, Aarhus School of Business, Denmark (2008)