4 Task-Action Consistency in Multi-Device Systems

Anke Dittmar, University of Rostock, Albert-Einstein-Str. 21, Rostock, Germany, ad@informatik.uni-rostock.de

Peter Forbrig, University of Rostock, Albert-Einstein-Str. 21, Rostock, Germany, pforbrig@informatik.uni-rostock.de

This paper reconsiders task-action consistency with respect to multi-device systems. We show how this concept influences our understanding of an appropriate model-based design approach. Additionally, we argue that concepts like consistency, their evaluation and their integration into formal methods cannot replace user-centered design approaches but must be combined in a reasonable way.

INTRODUCTION

In [7], Grudin referred to a workshop where 15 experts were not able to supply a ‘consistent’ definition of consistency with respect to user interfaces. We know versions like

- A similar (user) input in the same situation will have a similar effect on the (computer) system.
- The same input in similar situations will cause similar effects.

However, when can we say that two inputs are similar? When can we consider two situations as being the same or similar? While Grudin suggested to follow a user-centered design approach instead of relying on formal concepts such as consistency for evaluating user interfaces the understanding of what consistency could mean is discussed in [14]. It is seen as a concept which relates the dialogue language of the interactive system and the task language of a (competent) user. This idea is reflected, for example, in the distinction between action-effect consistency and task-action consistency (see next section).

Of course, the emergence of portable computing devices and (hopefully not;) ubiquitous applications makes the design of user-friendly systems even more difficult. In face of this development, criteria of usability such as consistency have to be reconsidered too. So, we have to find answers to questions like

- What aspects of consistency matter in cross-platform design?
- How can consistency be evaluated in the multi-device context and how can such measures be integrated into the design process? (see this workshop proposal)

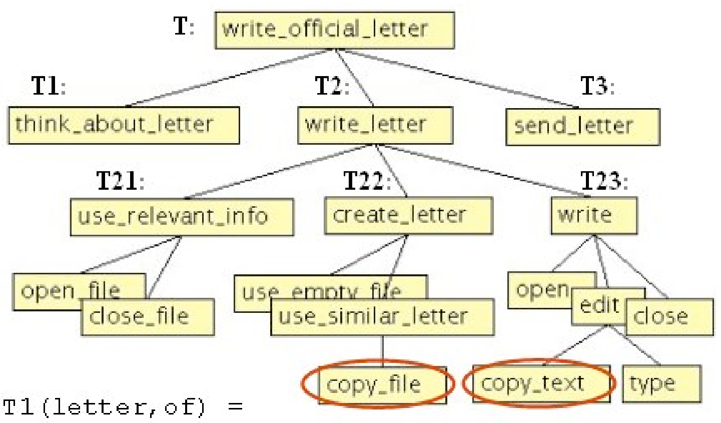
We think that it is necessary to ‘exploit’ both approaches mentioned above: the idea of a user-centered design process with an active user involvement, and the deepening of our understanding of formal concepts such as consistency in order to tackle the problem. This paper explores consistency, and particularly task-action consistency with respect to multi-device systems, within a model-based design and evaluation approach in combination with task scenarios.

TASK-ACTION CONSISTENCY

Dix et al. emphasize that consistency “is not a single property of an interactive system that is either satisfied or not satisfied. Instead, consistency must be applied relative to something” [[5]]. In [[7]], three types of consistency are distinguished: internal and external consistency (within and between applications), and consistency with respect to past real-world experiences. Consistency can be applied to different interface levels. It can support ease of learning, it can support ease of use, and this can be related to conflicting aims. Grudin argues that existing forms of consistency focus on general interface properties and that a shift to the consideration of users’ work and work context is needed. He further argues that a formally expressible consistency often lead designers to concentrate on the description of the software architecture. In [[10]], two types of consistency are proposed on the basis of a simple model of human-computer interaction called goal-action-effect cycle. If users want to complete a task by using an interactive system they have to “translate” the task goals into appropriate actions. By executing them they change the state of the system. These effects are presented and influence the further behavior of the user. Action-effect consistency guarantees that the same (low level) action has the same effect whatever the context is. For example, hitting control C should copy a marked object (e.g. text) into a buffer. Task-action consistency says that tasks which are considered as similar by users have to be completed by similar sets of actions. Task Action Grammars [[6]] can be seen as a formal approach to describe task-action consistency. However, only simple tasks are mapped to actions. Furthermore, it depends on the users and their specific situations what they would consider as similar tasks. So, this kind of consistency is even in the case of single applications problematic. One way to cope with it is to build systems which supply multiple task-action methods so that users have a choice [[10]]. However, to make task-action consistency a more useful concept for designing and evaluating interactive systems more elaborated task-action relations are needed which take into consideration complex user tasks and the context of use.

MBD: TASK MODELS AND DIALOG MODELS

In Model-Based Design approaches (MBD) different perspectives on the system under design or evaluation are described in different models on an abstract level. In this paper, we concentrate on task and dialog models. Task models are used to describe users’ task knowledge which is never independent of artifacts and means. A dialog model specifies possible orders of interactions, which are offered by an interactive system in order to support users in completing their tasks. We use TaOSpec as a specification formalism for both kinds of models (e.g. [[4]]). It allows the specification of tasks, task domain objects, dialogs, dialog objects, and actions along identical modeling principles. Models are described through elements (objects) and their mutual relations. A dedicated relation is the instance-pattern relation explaining abstractions. Objects are characterized by finite sets of attributes (name-value pairs). Furthermore, partial equations facilitate the description of subsets of instances of a pattern object. They can be used to define sub-tasks and sub-dialogs as well as certain states of domain and dialog objects. Figure 2 shows a fragment of a task model for writing an official letter. The task tree illustrates the hierarchical decomposition into sub-tasks. In addition, partial equations (as indicated below the tree) describe in which order sub-tasks can be performed. For example, in order to complete sub-task *write_letter* (T2), the iterative sub-task *use_relevant_info* is performed concurrently to *create_letter* and the iterative subtask write which have to be executed in sequence (* denotes the iterative operator, I the concurrent, and; the sequential one). A task tree is refined down to the level of basic tasks, which encapsulate actions. An action can create, manipulate, use, and destroy domain objects, which play the role of an artifact or of means with respect to this action. In Figure 2, letter is the artifact in *copy_text* and refers to the letter which has to be written. Object of (describing a list of open files with relevant information) is a means. A task model can be considered as a pattern object and its instance objects are descriptions of possible sequences of basic tasks/actions. To make a distinction between actions which are performed with or without the interactive system of interest we refer to the former kind of actions as *interactions*.



```
T1(letter,of) =
   ( *T1() | T2(letter,of) ) ; T3(letter),

T2(letter,of) =
    *T21(of) | (T22(letter) ; *T23(letter,of))
...
```

Figure 2. Part of a task model for writing an official letter.

Similar Tasks

As mentioned in the previous section, we need to be able to determine similar tasks to make task-(inter)action consistency a useful concept. Task models might help here. They allow to compare sets of sequences of basic tasks/interactions. They also allow to compare descriptions of domain objects and the way how they are manipulated or used. Let us assume a conceptual understanding of how to copy a file and a text as described by the following refinement of the task model in Figure 2.

```
[PRECOND bf IN FileBuf]
copy_file(tf,bf) =             copy_text(f,bt) =
  goto_source_folder()            goto_source_place()
  ; select_file(sf)             ; select_text(st)
  ; copy_to_buf(bf,sf)          ; copy_to_buf(bt,st)
  ; goto_target_folder()        ; goto_target_place()
  ; paste(tf,bf)                ; paste(f,bt)
  ; [ rename_file(tf) ]       % […] denotes an option
```

Objects *tf*, *sf*, *f* are instances of pattern *file*, *st* of *text*, and *bf*, *bt* of *buffer* where *bf* has to be a specific *buffer*, which additionally can store a file name (see precondition).

```
ELEMENT buffer                      ELEMENT file
ATTRIBUTES                          ATTRIBUTES
   content: string,                    name: string,
ADDITIONAL ATTRIBUTE                   location: string,
   file_name: string                   content: string
PARTIAL_EQU
   FileBuf = file_name!=nil  ...
```

Sub-tasks *copy_file* and *copy_text* could be considered as similar because the artifacts *tf* and *f* are manipulated in a similar way (the only difference is that *tf* has to be created while *f* already exists). In addition, the text of reused letters remains unchanged in both cases. Although *copy_file* describes two possible sequences of basic tasks (one ending with *rename_file* and one not), and *copy_text* specifies one these sets can be seen as similar.

In the following, a fragment of a dialog model specifying a word processing application is given.

```
dialog =
   open_file ; *edit ; close_file,
open_file = % + denotes the alternative operator
   create_file + open_existing_file,
edit =
   type + move_cursor + select_text + copy
   + cut + paste + safe + safe_as,
close_file =
   [ safe ] ; close,
open_existing_file =
   select_menu_entry_Open
   ; goto_folder ; select_file ; open,
safe_as =
   select_menu_entry_SafeAs ; goto_folder
   ; determine_file_name,
...
```

Similar to tasks, dialogs are hierarchically organized. Temporal constraints between sibling sub-dialogs and preconditions on the involved dialog objects serve to specify possible sequences of interactions. For simplicity, dialog objects and preconditions are omitted in the example. Take note that, in general, a dialog model has to ‘serve’ several task models. As one consequence, temporal constraints are often weaker.

Task-Action Consistency

As already mentioned, task-action consistency is given if similar tasks can be completed by similar sets of sequences of interactions. Task and dialog models allow the description of similarity on an abstract level. For example, sub-task *copy_text* could be performed by sequence SD1 of basic dialogs: *⟨move_cursor,select_text,copy,move_cursor,paste⟩*. The basic dialog *copy* could be realized by the interaction ‘press control C’ or by selecting an appropriate menu entry and so on. Sequence SD2=*⟨select_menu_entry_Open, goto_folder, select_file, open, select_menu_entry_SafeAs, goto_folder, determine_file_name⟩* might reflect a possible mapping from basic tasks of *copy_file*. Do SD1 and SD2 describe similar sets of sequences of interactions? This is certainly not the case. In fact, SD2 is even not a ‘correct realization’ of the conceptual task knowledge although it can be executed to copy a file. One reason is that there is no realization of the buffer. Instead, the source file is ‘open for accidental changes’.

TASK-ACTION CONSISTENCY IN CROSS-PLATFORM DESIGN

The distinction between internal and external consistency disappears with multi-device systems. Furthermore, the development of mobile devices and appropriate applications encouraged the discussion about context of use. In [[1]], context is defined as “any information that can be used to characterize the situation of entities…that are considered relevant to the interaction between a user and an application, including the user and the application themselves.” Here, we concentrate on task-relevant information.

Cross-platform consistency is characterized in [[15]] as follows. “A MUI [multiple user interface] can have a different look-and-feel while maintaining the same behavior over different platforms. For example, all user preferences must be preserved across the interfaces. If the end-user has specified a particular access mechanism using one interface, it should be used on all interfaces.” In [[10]], “the same abstract task-action mappings across interfaces” are proposed. Task and dialog models allow such a mapping. They can help to specify platform independent features of a system in a systematic way. However, we propose to demand a similar behavior with respect to tasks.

In many MBD approaches, task models serve to derive (mostly semi-automatically) dialog models (e.g. [[11]],[[9]]). Then, dialog models can be used to generate initial abstract UI prototypes. A prototype can be (“manually”) refined by adding concrete user interface elements without destroying references to the underlying task and dialog model (e.g. [28]). As for MUIs, it depends on the context (for example, on parameters of the physical devices such as screen size and storage, or on the location which also implies a certain social situation) which sub-task should be supported by a device (e.g. [[12]], [[3]]). Figure 3 illustrates an appropriate design approach. A general task model has to be restricted e.g. by filtering sub-tasks and restricting temporal relations between sub-tasks. These restricted task models are the starting point for deriving dialog models for different devices in use.

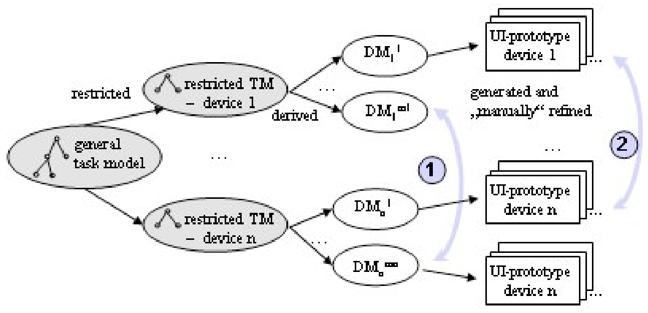


Figure 3 A task-related specification process for MUIs.

In this way, UI prototypes are guaranteed to obey the behavioral constraints specified in the underlying models. Hence, we can assume a certain degree of task-action consistency. However, we also need support for relating elements of dialogs of different devices (circle1, Figure 2) and for relating elements of UI prototypes (circle2). The integration of a pattern-based approach like [[8]] where patterns are used to redesign user interfaces could help. Pattern can be implemented on different platforms in different ways but they provide some kind of consistency.

SEAMLESS MOVE BETWEEN DEVICES: USING TASK SCENARIOS

In the context of cross-platform design, consistency should also serve as a criterion to measure how smooth users can switch between devices in accomplishing their tasks. It is stated in [[2]] that “usability of a multi-device system must be analyzed for each platform, taking into account each possible form of transition between the available devices”. In contrast, we propose not to strive for seamless switches between devices in general. However, people often develop habits and use single devices for certain purposes. For example, Ann could use her MP3 player to listen to music or audio books but Peter also uses the audio recording feature of his player. While Ann usually connects her player to their common PC to upload new songs or the next chapters of the actual book, Peter also wants to edit and manage recordings. Hence, it seems to be more realistic to develop automatic as well as interactive tool for supporting efficient task scenarios across devices. In Ann’s case, context information could be used to automatically open her uploading program and possibly the next chapters.

Task models which specify a set of task scenarios supply some kind of context with respect to an actual task. Say, for example, a user has accomplished sub-tasks A and B on device Dev1 and is using now device Dev2. According to the underlying task model he has to execute sub-task C or D in the next step. Then, provided that Dev2 supports these sub-tasks, the artifacts and means necessary for performing C or D could be presented. One could also imagine that a user applies (a perhaps simplified form of) task models to plan the execution of his work across different devices interactively. Take the example task of writing an official letter depicted in Figure 2. Perhaps, the user has compiled relevant information from several sources and created the letter. He wants to finish the letter while sitting in the train home. So, he ‘assigns’ task node T23 (*write*) to his notebook or PDA…

CONCLUSIONS AND FUTURE WORK

Although (or because) there are controversial discussions about consistency as a usability criterion the concept might help us to understand human-computer interaction more deeply. The focus here was on task-action consistency across MUIs. Task and dialog models specifying tasks and (inter)actions on an abstract level were used to support a more thorough discussion about similar tasks and similar actions. Some ideas for developing consistent interactive systems for different platforms or mobile devices within a model-based design process were presented. It was suggested to relate the concept of seamless moves between devices to task scenarios.

REFERENCES

[1] Dey, A.K., Abowd, G.D.: Support for the Adapting Applications and Interfaces to Context. In [[15]].

[2] Denis, C., Karsenty, L.: Inter-Usability of Multi-Device Systems – A Conceptual Framework. In [[15]].

[3] Dittmar, A., Forbrig, P., Reichart, D.: Model-based Development of Nomadic Applications . In Proc. of the IMC 2003, Rostock, 2003.

[4] Dittmar, A., Forbrig, P., Heftberger, S., Stary, C.: Tool Support for Task Modelling - A Constructive Exploration. Proc. EHCI-DSVIS'04, 2004.

[5] Dix, A., Finlay, J., Abowd, G.D., Beale, R.: Human-Computer Interaction, 3rd ed. Prentice Hall, 2004.

[6] Green, T., Payne, S.: Task-Action Grammar: the model and its developments. In Diaper, editor, Task Analysis For Human-Computer Interaction, Wiley, 1989.

[7] Grudin, J.: The Case Against User Interface Consistency. In Communications of the ACM, Vol. 32(10), 1989.

[8] Javahery, H., Seffah, A., Engelberg, D., Sinnig, D.: Migrating User Interfaces Across Platforms Using HCI Patterns. In [[15]].

[9] Luyten, K., Clerckx, T., Coninx, K., Vanderdonckt,J.: Derivation of a Dialog Model from a Task Model by Activity Chain Extraction, DSV-IS 2003 LNCS 2844, Springer, 2003.

[10] Monk, A.: Noddy’s guide to consistency. In Interfaces, No. 45, 2000.

[11] Paterno, F.: Model-Based Design and Evaluation of Interactive Applcations. Springer-Verlag, 2000.

[12] Paterno, F., Santoro, C.: One Model, Many Interfaces. In Proc. of the Fourth International Conference on Computer-Aided Design of User Interfaces, p. 143-154. Kluwer Academics Publishers, 2002.

[13] Reichart, D., Forbrig, P., Dittmar, A.: Task Models as Basis for Requirements Engineering and Software Execution. In Proc. of TAMODIA 2003, Prague, 2003.

[14] Reisner, P.: What is Inconsistency? In Diaper et.al., editors, Human-Computer Interaction - INTERACT'90, Elsevier, 1990.

[15] Seffah, A., Javahery, H., editors: Multiple user interfaces: cross-platform applications and context-aware interfaces. Wiley, 2004.