Resting Points

A User Interaction Pattern

Wolfgang Heinz

Introductiory Remarks

Motivation and History

The ideas for this pattern came forth when we had to redesign an interface for a banking system client consisting of many components. The components to be used ranged from legacy applications to newly designed parts. The emerging system was to exhibit a consistent and easy to comprehend interface and behavior to the user.

One of the design challenges proved to be the accomodation of parallelism in the work of the user. Different people came up with (or found in different systems) strikingly similar solutions, so that the request to write the solutions down in pattern form presented itself.

Although I am sure the pattern could be generalized much more, we stick to the domain of a business support system client as a starting point.

The pattern deals with HCI (human computer interaction) matters and thus is related to pattern languages in this area – we refer to *the Interaction Design Patterns Page* (see References) for a long list of . This pattern is especially indebted to the pattern language *Common Ground* (Tidwell (1999)), to which it could be considered an extension for a special common case.

From a rather technical point of view, our subject is an HCI analogon to work on distributed systems and concurrent programming.

Broader Context

Consider the client side of a business support system. A typical user is a clerk in a branch of a banking company. The system supports different tasks – teller's work as well as customer administration, consultant support, acquisition, etc.

The client system is to be written to an operating system with a graphical user interface and follows the respective interface guidelines (Windows, Apple, Motif,...).

Prototypical problems in the design of this type of system also center around the situation of parallel satisfaction of different customers' needs – e.g. while dealing with one customer's business a phone call must be taken to answer the needs of another customer. The clerk has to **shift attention** – how can this be achieved best? The accomodation of this shift of attention could be the focus of a pattern language from which we present the pattern that we consider the heart of that language.

Resting Points - The Pattern

Context

A clerk using a business support system with a graphical user interface.

Problem

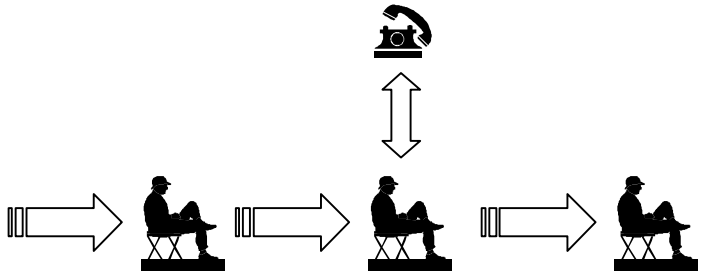
A business process takes a longer timespan for the user to accomplish, which means there may arise the need for interruption or suspension. How can we accomplish this in an orderly manner?

Forces

- The complete task is rather time consuming but should be performed to completion
- It is necessary to have a quick reaction to a secondary task request
- The user has a limited attention span to a suspended task – it can easily be forgotten
- Due to space limitations on the screen the suspended task and the secondary task cannot be shown together
- The objects and relations involved in the task have to be in a consistent state when stored in a database – the process forms a ”long transaction“
- The objects and relations involved usually are a combination of (independent) components
- The secondary task may involve (some) of the same components as the primary task

Solution

Provide a well defined set of situations *(resting points)* in which the user can safely suspend the business process, perform other actions, and afterwards resume the process in an orderly manner.



Make suspension and resumption easy and clear, and provide visual clues

- of how to suspend the process,
- of the process in the suspended state,
- of how to resume the process.

Discussion and Implementation Issues

Task Organisation

- By splitting up the task there are „small completions“ which are less time consuming.
- The reaction to the secondary task can be reduced to the timespan up to the next resting point
- If the business process can be split up into several independent steps this usually leads to a segmentation of the task into activities concerning different objects. The size of the object (types) has influence on the possible size of the step.
- If a process involves a sequence of updates to (partially) independent objects, these updates form natural steps between resting points.
- Such resting points at the borders of interactions with individual objects also ease the implementation of concurrency issues (as the secondary task may involve the same object as the suspended task).
- As the steps between resting points define consistent states, objects involved in the primary as well the secondary process should be in a consistent state as well. If an object may not be edited under certain circumstances this information should be provided by the object (the object has its own *Responsibility of State)* and the higher level business process should be notified.
- By defining the resting points as consistent intermediate states the transaction requirements can be met. [Don’t underestimate this as a trivial task – this can mean reconsideration of the whole business logic of the process. If this is too difficult or impossible to achieve consider making one or several intermediate steps larger.]
- Special consideration has to be given to the limited attention span of the user, therefore the granularity of the tasks should be chosen as small as possible as well as the suspended task should be marked unobstrusively but firmly.

User Interface Organization

[Where an aspect in the following discussion is already treated in a pattern in *Common Ground* (Tidwell 1999) or a pattern therein is related to a point made here we annotate this by *(CG: Name-of-the-Pattern).*]

- For showing the task in a suspended state use a *symbol (icon)* for the task in a display area *(CG: Status Display)*
- ...which means that the process or state should be *reified* by putting a visual bracket around it (e.g. a window frame) and be *named*.
- Consider to arrange the windows representing the tasks "on top of" each other (cascading windows, tabbed windows) and label the windows accordingly *(CG: Stack of Working Surfaces; Pile of Working Surfaces)*
- Use different techniques to provide feedback of a pending process including possible use of color highlighting, flashing of icons etc. *(CG: Color-coded Section)*

- Provide direct access to the suspended task using buttons, icons, dynamic menu entries etc., but always provide clues which task is current or active *(CG: Map of Navigable Spaces)*
- In addition, you can use a *list of pending activities* including reminders, or a history list, or (automatic) bookmarks *(CG: Bookmarks)*
- This list can also be navigated by use of "Back"-mechanisms provided by toolbar buttons, back-keys or menu entries *(CG: Go Back One Step)*
- The state of the process at the suspension point should be remembered and presented to the user at the resumption point *(CG: Remembered State)*
- If the smaller parts of the process between the resting points can not be interrupted under any circumstances, use a *modal window*. (Try to avoid this!)

Examples

- Applications supporting a workflow often exhibit the properties described above:

  A clerk is working on a loan entering product parameters and information on the people involved. The telephone rings and information is requested on the balance of a certain account. The clerk only has to identify the first participant in the loan and can then safely suspend this task by simply moving it into the background (or pressing a "Suspend"-Button). The clerk can work on the second task by selecting the account (even referring to the same cusomer he worked on in the suspended task). The loan is identified by its number and may be brougt into the foreground anytime by clicking an icon in a stack area. After completing the balance request the suspended task is brought to the foreground automatically and the work on the loan can be resumed at the same state the interruption was made.

  The Resting Points here are designed into the application

- Work involving knowledge acquisition in web-based applications can also be considered special cases of this pattern:

  The task is accomplished by following links in a browser. If a distracting task has to be opened, a link is simply taken and after the detour the first task is resumed by using the "History" or the "Back"-Buttons. Alternatively the detouring link can be opened in a separate window which will be opened after completion of the second task.

  The Resting Points here are implicit by reducing the small tasks to the content of one web page.

Related Patterns

Apart from the patterns already mentioned in the Discussion section, the (analytical) partitioning of the long task into smaller ones can be compared to a synthetical view using GOF patterns (thanks to J. Kopf for the hint): the smaller steps can be considered incarnations of the *command* pattern which give a macro command using the *composite* pattern.

Appendix

References

The Interaction Design Patterns Page
http://www.pliant.org/personal/Tom_Erickson/InteractionPatterns.html

Tidwell, Jenifer (1999) – *Common Ground: A Pattern Language for Human Computer Interface Design.* http://www.mit.edu/~jtidwell/common_ground.html

Acknowledgements

This paper draws ideas from many people – thanks to my colleagues from SparDat and ErsteBank Manfred Syrovatka, Albert Strimitzer, Gerhard Jarusch, Erich Gstrein, Alois Knotz, Erich Vaplon, Hans Felser, Michael Pendl, Stefan Kovalovsky; Andreas Schabus, Johann Kopf and the participants of the Oct. 1999 Vienna Pattern Writing Workshop group led by Christa Schwanninger.

Version 0.3; March 5th, 2000

Wolfgang Heinz

Baumgartenstr. 53
A-1140 Wien
Austria

Tel. & Fax: (+43-1) 912 42 21
mailto:Wolfgang.Heinz@acm.org