User Interface Design for Searching

A Pattern Language

Tim Wellhausen

kontakt@tim-wellhausen.de
http://www.tim-wellhausen.de

Jan 13, 2006

Abstract: The ability to perform search requests in business data is an important asset of any information system. To be accepted by its users, an information system needs a front-end user interface that leverages the features and complexity of the back-end search facilities. Intended for GUI designers and developers, this paper proposes a pattern language for the design of such user interfaces.

Introduction

A modern information system typically is connected to data sources that store a huge amount of structured data. To be of economic value, data must be easily accessible to the users of the information system. It is therefore important to provide means of searching that are both powerful and easy to use.

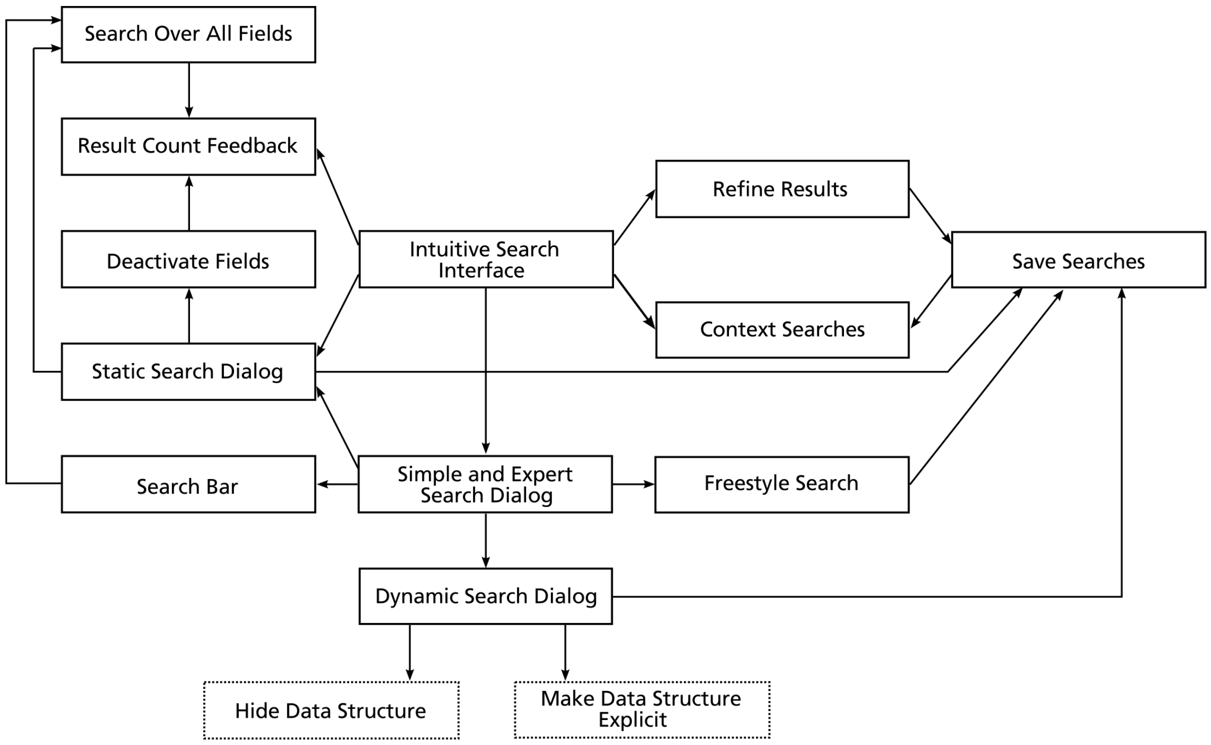
The design of a powerful back-end that actually executes search requests poses many problems in itself (see, for example, [Wellhausen04]). This paper focuses on design alternatives for front-end user interfaces that enable users to easily create search requests and efficiently search for business data.

The patterns apply to both rich-client and web applications. Because of their implementation complexity, however, some patterns (e.g. DYNAMIC SEARCH DIALOG) are mostly found in rich-client applications. The pattern language focuses on searching data that is stored in databases. Patterns for searching in documents such as “FIND AND REPLACE” and “FIND NEXT” are outside the scope of this paper. Key terms are explained in a glossary.

Overview

The *User Interfaces for Searching* pattern language consists of 14 patterns. The following diagram shows these patterns and outlines the structure of the language. The starting point is INTUITIVE SEARCH INTERFACE. The arrows in the diagram denote the order in which you may apply the patterns: the resulting context of one pattern is part of the problem of another pattern. Two of the patterns are not described in this paper; only their thumbnails are given.

The stars behind the pattern names indicate the author's confidence in the patterns: two stars are reserved for patterns that are known to work well, while zero stars indicates that a pattern may have alternative solutions.



Thumbnails

The following thumbnails contain the problems and the solutions of all patterns.

Pattern name	*Problem*	*Solution*
Intuitive Search Interface	Many search interfaces are difficult to comprehend because their usage requires knowledge about the data structures that are searched and the search capabilities at the data sources. How can we provide a search interface that users can intuitively understand?	Shield the users from technical details and choose search metaphors users are comfortable with instead of technical and mathematical vocabulary.
Static Search Dialog **	How can we support users to quickly create their individual search requests?	Provide a search dialog whose appearance is static, i.e. the dialog contains a fixed number of search fields whose positions on the dialog do not change.
Deactivate Fields **	When a search field is left empty, how can the user decide whether the system should search for empty values or whether it should exclude the respective search parameter from the search request?	Support an explicit activation and deactivation of search fields and include only activated search fields in a search request.
Result Count Feedback *	Databases in information systems may become huge. Searching in the stored data may therefore return big result sets that slow down the system. How can we avoid big result sets?	Provide feedback on the expected result set size of a search request before this search request is actually executed.
Search Over All Fields *	An information system may keep similar pieces of data at different places. How can the users easily search for occurrences of a particular value in all of these locations?	Let the system execute search requests that automatically include all business fields where the requested data may be stored.
Refine Results	Initial search requests often return result sets where it is difficult for users to quickly identify the results they are looking for. How can we support an iterative search request refinement?	Allow the users to perform searches within the results retrieved from earlier search requests. Allow them to backtrack to previous results to change earlier decisions.
Save Searches **	Some search requirements do not change. How can we avoid that users have to recreate their typical search requests over and over again?	Let the users save, load, manage, and re-execute their search requests.
Context Searches *	The same application context often requires the same search requests to be executed. How can we avoid that users have to create such search requests explicitly?	Provide predefined search requests and make them available in the appropriate application context.
Simple and Expert Search Dialog **	Most of the time, users need only few search features despite the richness of the available functionality. How can we ensure that average users can create their standard search requests easily and fast whereas expert users have access to the full search functionality?	Provide a simple search dialog that contains only the most important features. Additionally provide an expert search dialog that supports all features available.
Search Bar **	How can we avoid a search dialog that takes a lot of space on the screen if the search re-	Add a small search bar to the application that provides space for entering a

Pattern name	*Problem*	*Solution*
	quirements are simple?	value for one search parameter. Start searching when the users press the enter key.
Dynamic Search Dialog *	The domain model may contain a huge number of business fields that can be searched. How do we avoid a search dialog that has to provide a static search field for every searchable business field?	Provide a search dialog that lists all available search parameters and let the users explicitly choose the search parameters to be included in a request.
Hide Data Structure	The structure of the domain model may be complex and difficult to understand. How can we avoid that all users have to learn and understand the domain model?	Hide the structure of the domain model and present a flat view of it.
Make Data Structure Explicit	The domain model is complex and expert users need full control over formulating their search requests. How do we appropriately represent the domain model?	Provide a graphical view of the business domain model that includes all entities, their business fields and associations. Let the users navigate through the model and let them choose any business field as search parameter.
Freestyle Search **	Users of some information systems need to quickly execute complex search requests. How can we provide an appropriate user interface?	Provide a domain specific search language that supports all features of the back-end search facilities. Expert users may formulate their search requests by entering statements in the search language.

Intuitive Search Interface

The user interface of an information system has to support searching.

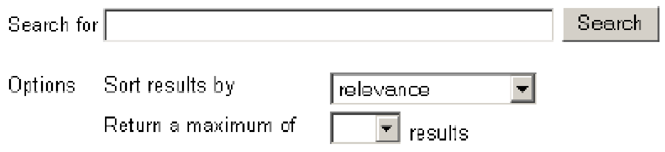
Many search interfaces require technical knowledge about the back-end search facilities. How can we provide a search interface that users can intuitively understand?

You cannot assume that all users understand the fundamentals of information retrieval and know the capabilities and restrictions of the back-end search facilities. However, the user interface for searching must be suitable for all user.

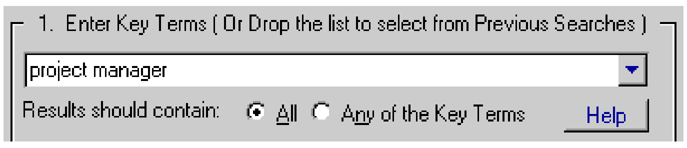
Therefore:

Shield the users from technical details and choose search metaphors users are comfortable with instead of technical and mathematical vocabulary.

You should not require the users to learn a technical search language like SQL. Instead, create a STATIC SEARCH DIALOG that explicitly provides GUI elements to control all features of the back-end search engine. Ensure consistent default settings among all GUI elements. For example, typical GUI elements let the users set the sort order and the maximum number of results like on the *Cisco Datasheet Website*.



Most users do not know the fundamentals of Boolean algebra such as the precedences of *ands* or *ors* in Boolean expressions. Instead, you may support a syntax for entering search phrases as it is provided by many Internet search engines ("+term1 -term2"). Alternatively, you may provide GUI elements to let the users control the interpretation of a search phrase. The resume finder *infoGIST*, for example, provides such GUI elements.


1. Enter Key Terms (Or Drop the list to select from Previous Searches)
project manager
Results should contain:
All
Any of the Key Terms
Help

If some expert users need advanced search capabilities, you may provide SIMPLE AND EXPERT SEARCH DIALOG. In that case, one search dialog provides a simplified user interface, whereas the other one exposes all features of the back-end search engine.

If the result set size of a typical search request is big, present RESULT COUNT FEEDBACK to your users and let them iteratively REFINE RESULTS. Provide predefined CONTEXT SEARCHES if in a specific application context, users always need to perform the same search requests.

Static Search Dialog **

We are developing an Intuitive Search Interface for an information system. We may be implementing the simple part of Simple and Expert Search Dialog.

How can we support users to quickly create their individual search requests?

The visual appearance of a search dialog strongly influences the usability of a search dialog. New users should easily find all features, whereas experienced users should be able to quickly create search requests. The average user should be able to open the search dialog, enter some values and execute a search request within a couple of seconds.

Therefore:

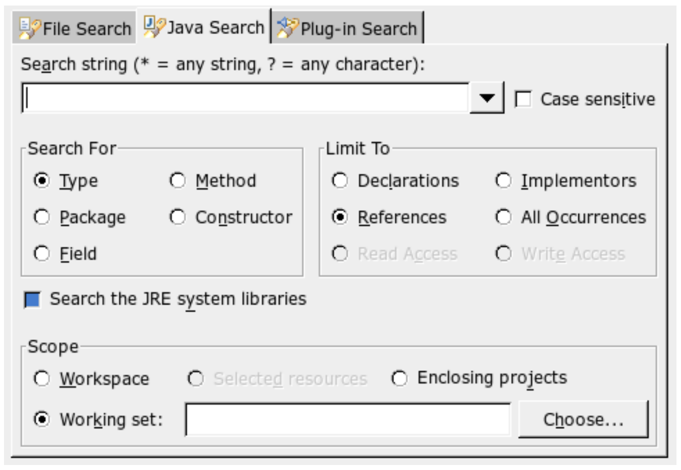
Provide a search dialog whose appearance is static, i.e. the dialog contains a fixed number of search fields whose positions on the dialog do not change.

A static search dialog provides an individual search field for every searchable business field. Additional GUI elements allow the users to specify search settings. The position of search fields on the dialog should be fixed so that users may recognize and find them easily. The order of the search fields and other GUI elements on the dialog should represent their respective importance for typical search requests.

Static search dialogs are very common. In *Eclipse*, the Java search dialog presents a typical example of such a dialog.

A static search dialog has limitations. Whenever the domain model changes, the search dialog has to be modified as well by adding or removing search fields.

If the users need to search for empty values, you should consider to let them Deactivate Fields. If they frequently execute standard search request, you may let them Save Searches.



Search string (* = any string, ? = any character):

Case sensitive

Search For: [✓ Type] Method Package Constructor Field

Limit To: Declarations Implementors [✓ References] All Occurrences Read Access Write Access

[✓ Search the JRE system libraries]

Scope: Workspace Selected resources Enclosing projects [✓ Working set:]

Deactivate Fields **

We are developing a STATIC SEARCH DIALOG.

If a search field is left empty, how can the user decide whether the system should search for empty values or whether it should exclude the respective search parameter from the search request?

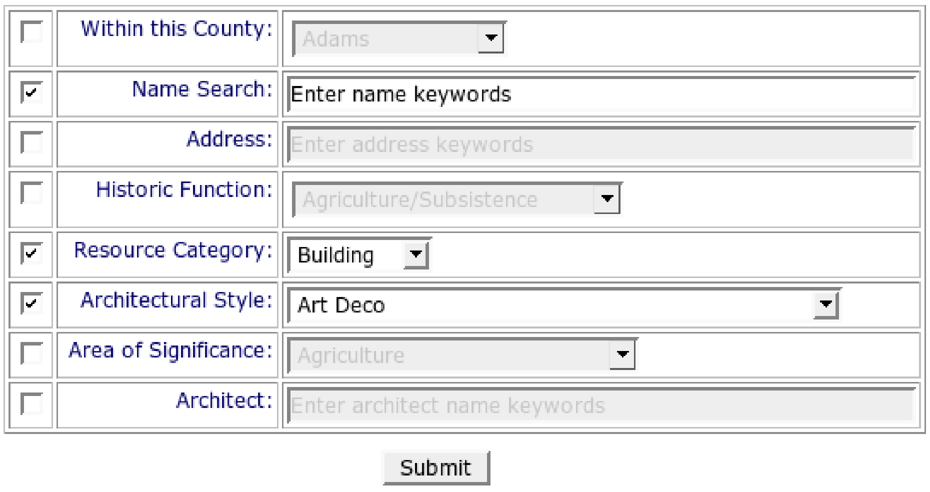
Depending on the semantics of the domain model, you may always exclude those search fields from a search request into which the users enter no values. If you have to support searching for empty business fields, however, you need to distinguish between searching for empty values and not including a search field in a search request.

Therefore:

Support an explicit activation and deactivation of search fields and include only activated search fields in a search request.

A check box in front of each search field is a convenient means to let the users enable and disable a search field without losing its current value. A search field is included in a search request only if the corresponding check box is selected. As downside, search dialogs look more messy and their control code might become more complex.

The search dialog of the *Pennsylvania's Historic Architecture & Archaeology* web site provides a check box in front of each search field. The search fields are enabled or disabled according to the states of their check boxes.



☐	Within this County:	Adams
✓	Name Search:	Enter name keywords
☐	Address:	Enter address keywords
☐	Historic Function:	Agriculture/Subsistence
✓	Resource Category:	Building
✓	Architectural Style:	Art Deco
☐	Area of Significance:	Agriculture
☐	Architect:	Enter architect name keywords

You may present RESULT COUNT FEEDBACK so that the users get immediate feedback on the consequences of adding or removing a search parameter.

Result Count Feedback *

We are developing an Intuitive Search Interface for an information system. Users may Deactivate Fields or Search Over All Fields.

Databases in information systems may become huge. Searching in the stored data may therefore return big result sets that slow down the system. How can we avoid big result sets?

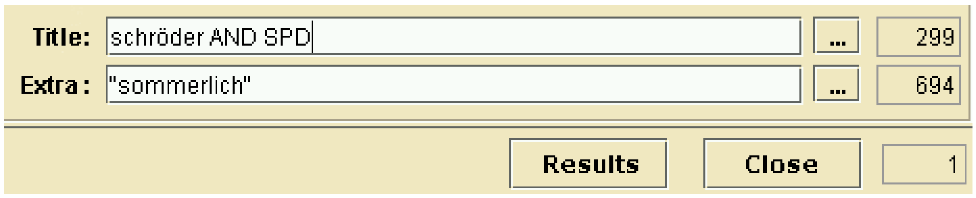
For a user, it is difficult to find the relevant data in a large result set. Even worse, transferring a large result set over the network may slow down the whole system. Users may become impatient and start their search requests several times in a row, thereby degrading the response times even more. If the users knew the number of hits for a search request, they could adjust the search parameters in advance to reduce the result set size.

Therefore:

Provide feedback on the expected result set size of a search request before this search request is actually executed.

You should try to reduce the users' effort to determine the result set size. Ideally, you present result counts without explicit user interaction. A rich-client application, for example, may retrieve result counts in the background while the users are entering values. An individual hit count may be displayed next to each search field. Whenever the users change a value, the result counts are updated automatically.

ProContent, an internally used information system of a television company, has a search dialog that shows both result counts for every search parameter and for the overall search request.



Title:	schröder AND SPD	...	299
Extra:	"sommerlich"	...	694

Depending on the data source, an information system may compute the result set size significantly faster than it may retrieve the complete result set. A full-text search engine with a complete index of all words, for example, may return the number of hits very fast, although it may take a considerable amount of time to fetch and return the results.

Result count feedback may degrade performance, on the other hand, if the database or search engine cannot efficiently retrieve result counts. Regardless of the efficiency of the back-end search engine, you have to trade off the decreased number of actually retrieved result sets against the increased number of result count queries.

Search Over All Fields *

We are developing a Static Search Dialog or a Search Bar.

An information system may keep similar pieces of data at different places. How can the users easily search for occurrences of a particular value in all of these locations?

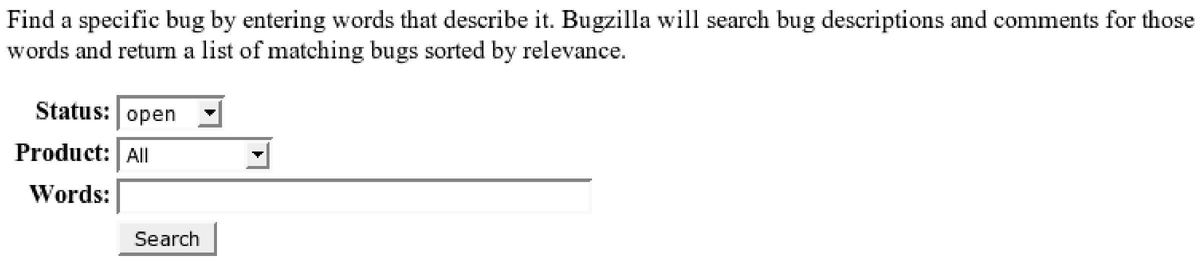
The domain model may not be consistent so that some pieces of information are redundantly stored at different places. Even if the domain model is consistent, similar pieces of information may be stored in different places. For instance, a content management system may store descriptions of different lengths for each content item.

Therefore:

Let the system execute search requests that automatically include all business fields where the requested data may be stored.

The users should enter a value in one search field without caring which additional business fields are searched by the system. The decision which business fields to add to a search request has to be taken during application development.

Bugzilla, a bug tracking system, supports a simplified search page. The following example shows how users can retrieve bug reports in which the phrase they enter may appear either in the description or in the comment of a report.


Find a specific bug by entering words that describe it. Bugzilla will search bug descriptions and comments for those words and return a list of matching bugs sorted by relevance.

Status: open

Product: All

Words:

Such a feature may yield a high system load if it is used extensively, in particular if a database does not contain indexes on the fields that are searched. It may cause huge result sets so that the users do not find what they are actually searching for. Consider providing Result Count Feedback to make the users aware of the number of results in advance.

Refine Results

We are developing an Intuitive Search Interface for an information system.

Initial search requests often return result sets where it is difficult for users to quickly identify the results they are looking for. How can we support an iterative search request refinement?

Search requests may return many results. Advanced users would like to modify their search requests iteratively to reduce the result set count, for example by adding additional search parameters. If users would like to try different search parameters they need to switch easily to earlier search requests which is only possible if the history of executed search requests is accessible.

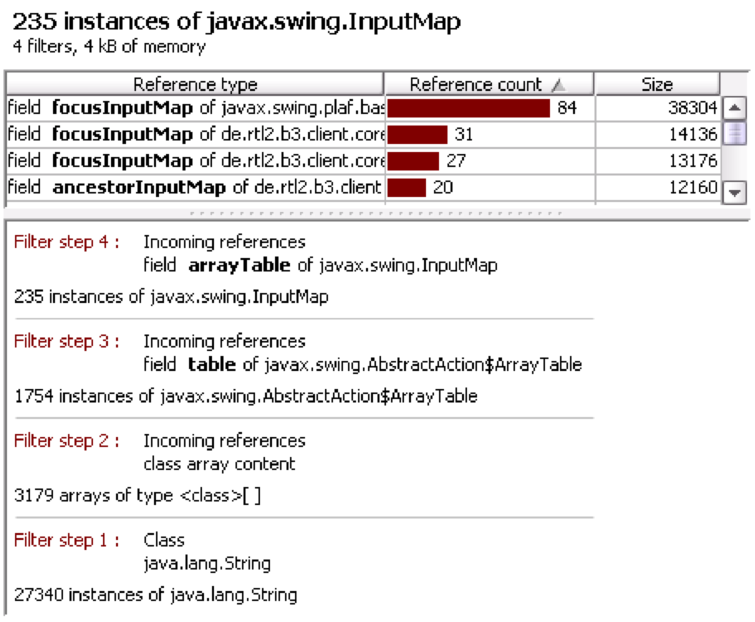
Therefore:

Allow the users to perform searches within the results retrieved from earlier search requests. Allow them to backtrack to previous results to change earlier decisions.

Step by step, users may decrease the result set size by applying further search parameters until the result set size is small enough to be closely examined. Keep information about each search request so that the users may backtrack if they find out that they are stuck in a dead end.

Searching within previous results may be implemented in different ways: the system may keep the identities of the resulting data records and narrow the next search request to those identities. Or it may keep the search request and enhance it with new search parameters. In the latter case, the system may automatically Save Searches so that users may resume refining their results at a later date.

JProfiler, a Java profiling tool, supports analyzing a snapshot of an object heap. It keeps information about every decision the users take and lets them backtrack to all previous steps.



235 instances of javax.swing.InputMap
4 filters, 4 kB of memory

Reference type	Reference count	Size
field **focusInputMap** of javax.swing.plaf.bas	84	38304
field **focusInputMap** of de.rtl2.b3.client.core	31	14136
field **focusInputMap** of de.rtl2.b3.client.core	27	13176
field **ancestorInputMap** of de.rtl2.b3.client	20	12160

Filter step 4 : Incoming references
field **arrayTable** of javax.swing.InputMap
235 instances of javax.swing.InputMap

Filter step 3 : Incoming references
field **table** of javax.swing.AbstractAction$ArrayTable
1754 instances of javax.swing.AbstractAction$ArrayTable

Filter step 2 : Incoming references
class array content
3179 arrays of type <class>[ ]

Filter step 1 : Class
java.lang.String
27340 instances of java.lang.String

Save Searches **

We are developing a Static Search Dialog, a Dynamic Search Dialog, or Freestyle Search. Users may Refine Results.

Some search requirements do not change. How can we avoid that users have to recreate their typical search requests over and over again?

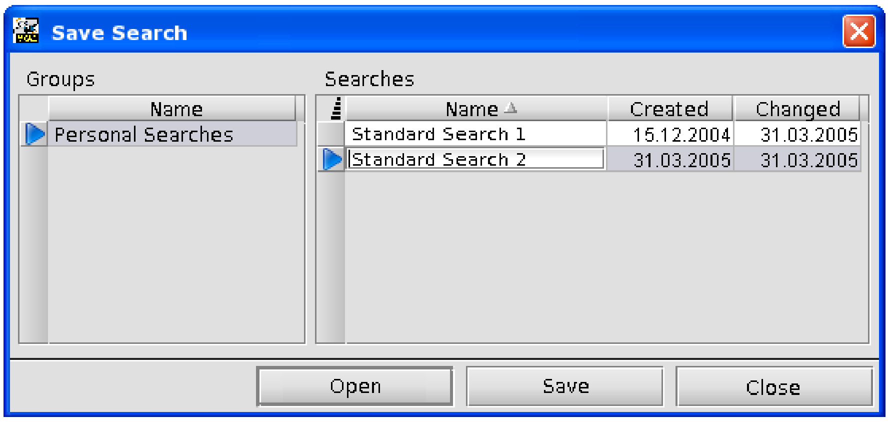
For many tasks, users need typical search requests that rarely change but that cannot be anticipated by application developers. Nevertheless, it is necessary to support users in avoiding the repetitive creation of frequently executed search requests.

Therefore:

Let the users save, load, manage, and re-execute their search requests.

The system has to be able to save the current state of a search dialog and store it persistently. The users may later choose any of their saved search requests for execution. If a saved search request is retrieved, the search dialog has to restore the saved values and let the users modify and execute the request.

Sometimes, a group of users have the same search requirements. In that case, grant key users the right to share their saved search requests with all group members. Group members may not have the permissions to modify saved search requests, only the execute them.



Name
Personal Searches

Name	Created	Changed
Standard Search 1	15.12.2004	31.03.2005
Standard Search 2	31.03.2005	31.03.2005

B3, an internally used information system of a media company, provides a search dialog that enables the users to manage and share their search requests.

To support such a feature, the system must be able to store and restore search requests. Consider creating an abstraction layer that separates the description of search requests from their execution.

Search requests that are saved by users may be made available as Context Searches.

Context Searches *

We are developing an Intuitive Search Interface for an information system. The users may Save Searches.

The same application context often requires the same search requests to be executed. How can we avoid that users have to create such search requests explicitly?

A search dialog gives users full flexibility in creating search requests. Sometimes, such flexibility is not necessary because the users need to execute standard search requests that do not change. In a library, for example, the librarian always needs the list of all books that are borrowed by a person when that person returns a book.

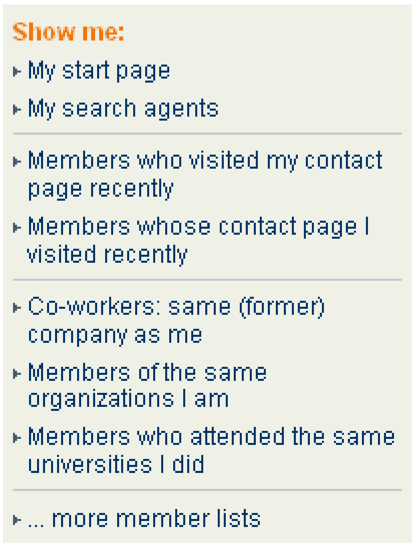
Therefore:

Provide predefined search requests and make them available in the appropriate application context.

A rich-client application may provide a context specific pop-up menu whose menu items directly execute search requests when chosen. A web application may list links to all predefined search requests that are available for the current user in the current context. Clicking on a link executes the search request and opens a new page with the results.

Search requests may be predefined by application developers if the developers know the users' search requirements in detail. Search requests may also be predefined by the users themselves if they can Save Searches and provide a hint to which application context a saved search request applies.

OpenBC, a platform for establishing business contacts, provides links to search for persons who have the same background or similar interests.


Show me:
My start page
My search agents
Members who visited my contact page recently
Members whose contact page I visited recently
Co-workers: same (former) company as me
Members of the same organizations I am
Members who attended the same universities I did
... more member lists

Simple and Expert Search Dialog **

We are developing an INTUITIVE SEARCH INTERFACE for an information system.

Most of the time, users need only few search features out of the richness of the available functionality. How can we ensure that average users can create their standard search requests easily and fast whereas expert users have access to the full search functionality?

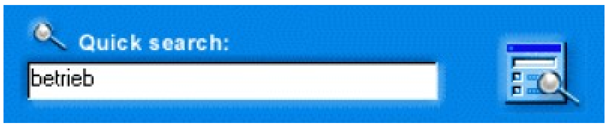
Most of the time, most users only need a limited sub set of all features available. If a system only provides one search dialog that contains the complete search functionality, using such a dialog is difficult for average users. If some features are left out, on the other hand, some search requests cannot be created any more.

Therefore:

Provide a simple search dialog that contains only the most important features. Additionally provide an expert search dialog that supports all features available.

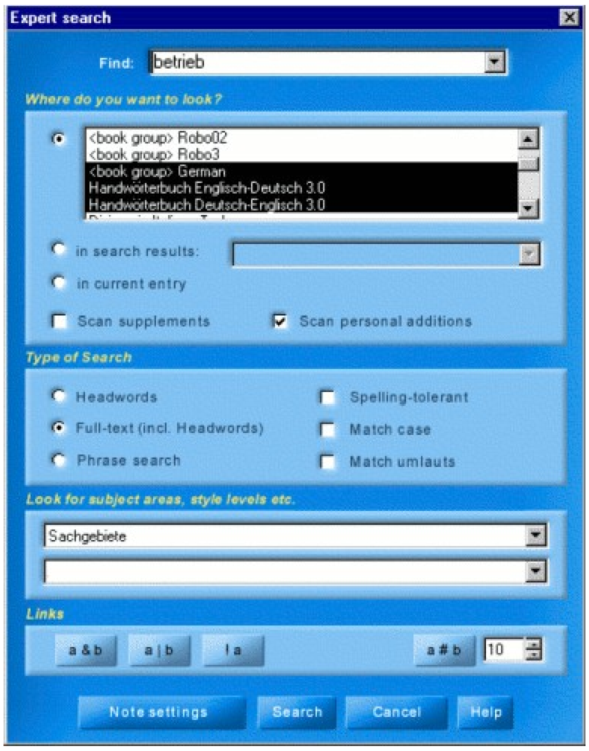
A simple search dialog makes it easier for novices to learn the basics of searching and lets the more experienced users search fast and conveniently. An expert search dialog is made for advanced users who have a broad understanding of the domain model and of the search capabilities of the back-end.

You should only include those features in a simple search dialog that are useful to most users. A feature that is needed by one group of users may not be important to another group. In case of doubt, restrict these features to the expert search dialog. Typical features that should only be provided in an expert search dialog include sorting and an advanced query syntax.


Quick search: betrieb

The *Langenscheidt e-dictionary* provides both a simple and an expert search dialog. The simple search dialog is an example of a SEARCH BAR that contains only one search field whereas the expert search dialog makes the complete search functionality available.

A simple search dialog typically is a STATIC SEARCH DIALOG. An expert search dialog may be a DYNAMIC SEARCH DIALOG that allows to add search fields to a search interface dynamically. It may as well provide FREESTYLE SEARCH, i.e. the users may enter search requests using a dedicated search language.


Expert search

Find: betrieb

Where do you want to look?

[✓] <book group> Robo02
<book group> Robo3
<book group> German
Handwörterbuch Englisch-Deutsch 3.0
Handwörterbuch Deutsch-Englisch 3.0

in search results:

in current entry

Scan supplements [✓ Scan personal additions]

Type of Search

Headwords [✓ Full-text (incl. Headwords)] Phrase search

Spelling-tolerant Match case Match umlauts

Look for subject areas, style levels etc.

Sachgebiete

Links

a # b 10

Search Bar **

We are developing a simple search dialog as part of Simple and Expert Search Dialog.

How can we avoid a search dialog that takes a lot of space on the screen if the search requirements are simple?

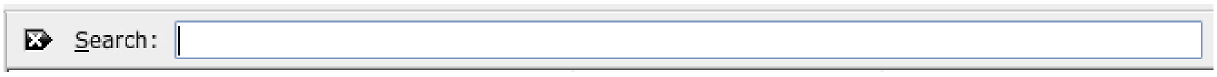
A typical search dialog takes up permanent space on the screen so that the users have less space available for other dialogs. If there is an expert search dialog available, the requirements for a simple search dialog are low. It may provide only one search field.

Therefore:

Add a small search bar to the application that provides space for entering a value for one search parameter. Start searching when the users press the enter key.

A search bar should always be visible, for example as part of the application tool bar. By pressing a specific key combination, the keyboard focus should move to the search field so that the users may immediately fill in a value and start searching by pressing the enter key.

KMail, an e-mail client application, provides a search bar that simplifies searching in e-mail bodies for a given phrase.



A drop down list box in front of the search field may allow users to choose among different search parameters. The *Firefox* web browser, for example, provides a search bar with which users may send search requests to a configurable Internet search engine.

A search bar becomes more powerful if users may Search Over All Fields, i.e. if they only have to fill in one value and several business fields are searched at once.

The Search Bar pattern mainly applies to rich-client applications. For web applications, it is known as Search Box ([Graham02]).

Dynamic Search Dialog *

We are developing the expert part of SIMPLE AND EXPERT SEARCH DIALOG.

The domain model may contain a huge number of business fields that can be searched. How do we avoid a search dialog that has to provide a static search field for every searchable business field?

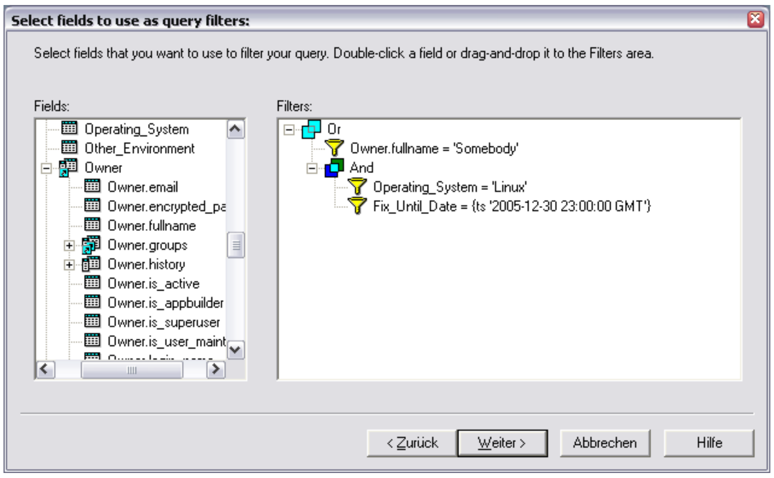
The domain models of some information systems change often because of new business requirements. If you provide a STATIC SEARCH DIALOG, you need to update the search dialogs frequently, which may be expensive. A STATIC SEARCH DIALOG may also be difficult to use because of its complexity. It is hard to arrange all search fields in a way that the users can easily recognize and find them.

Therefore:

Provide a search dialog that lists all available search parameters and lets the users explicitly choose the search parameters to be included in a request.

Most expert users have a good understanding of the domain model so that you should MAKE DATA STRUCTURE EXPLICIT. In that case, a tree or graph of search parameters may represent the structure of the domain model. Otherwise, you may HIDE DATA STRUCTURE and provide a flat list of all search parameters to choose from.

The chosen search parameters have to be combined before their execution as a search request. Typically, all search parameters are combined by a Boolean *and* operator. You may as well allow the users to explicitly define the combining operator. You may additionally support hierarchically combined search parameters (see the example below).


Select fields to use as query filters:

Select fields that you want to use to filter your query. Double-click a field or drag-and-drop it to the Filters area.

Fields:
```
Operating_System
Other_Environment
Owner
  Owner.email
  Owner.encrypted_pa
  Owner.fullname
  Owner.groups
  Owner.history
  Owner.is_active
  Owner.is_appbuilder
  Owner.is_superuser
  Owner.is_user_maint
```

Filters:
```
Or
  Owner.fullname = 'Somebody'
  And
    Operating_System = 'Linux'
    Fix_Until_Date = {ts '2005-12-30 23:00:00 GMT'}
```

ClearQuest, a bug tracking application, contains a dynamic search dialog. On the left hand side, all available search parameters are shown. On the right hand side, a user may create a hierarchically structured search request by dragging and dropping search parameters.

To support a dynamic search dialog, you need a back-end search engine that supports dynamic search requests. A QUERY ENGINE (see [Wellhausen04]) is a pattern that describes a solution for executing dynamic search requests.

A dynamic search dialog is a powerful means to search for business data. It may likewise be employed to create business reports. However, it is expensive to built and it usage is quite difficult to understand since it is designed mainly for expert users.

If the users frequently execute standard search requests, you may let them SAVE SEARCHES.

Freestyle Search **

We are developing the expert part of SIMPLE AND EXPERT SEARCH DIALOG.

Users of some information systems need to quickly execute complex search requests. How can we provide an appropriate user interface?

Some expert users don't need a graphical user interface to formulate their search requests. They know exactly what they are searching for, they know the domain model, and they know the fundamentals of information retrieval.

For any given system, there will always be another requirement that does not fit the current implementation. Extending an existing search dialog is expensive and may disturb the users that are familiar with the current version.

Therefore:

Provide a domain specific search language that supports all features of the back-end search facilities. Expert users may formulate their search requests by entering statements in the search language.

The search language may be the native query language of the data source. If a relational database is used, experts may formulate their search requests in SQL. If the users should be shielded from the data sources, however, you need an abstraction from the native search language.

In that case, you need to invent a domain specific search language that is transformed at runtime into the native query language. You may implement arbitrarily complex search features by providing appropriate search expressions. You are not limited by a user interface toolkit. If new search features have to be included, you just need to extend the search language and the transformations.

If you provide FREESTYLE SEARCH, you may save development costs because you don't need to provide a graphical user interface. On the other hand, the users have to be trained extensively.

Many European travel agencies are connected to the online booking system *Amadeus* that supports freestyle searching. Its search language is powerful but cryptic. Travel agents have to regularly update their knowledge as the search language continues to evolve. The following example is a search request to list all available economy cars in Frankfurt on April 04 to be rented for one day starting at 10am:

CA ZI FRA 04APR-1 / ARR-1000-0800 / VT-ECMN / ID-Y2XXXX

If the users frequently execute standard search requests, you may let them SAVE SEARCHES.

Acknowledgments

I would like to thank my shepherd Frank Buschmann for his encouragement during the review process of this paper. His many valuable comments helped a lot to improve the patterns. I'd also like to thank the participants of the EuroPLoP 2005 writers' workshop for their insightful feedback and suggestions, in particular Allan Kelly and Hans Wegener.

Glossary

This glossary explains terms that are used throughout the paper.

Domain model	A model of the data that is available within an information system.
Business field	An element of a domain model that can be searched.
Search field	A graphical component of a search dialog, for example a text field, into which users enter values to search for. Typically, every search field is connected to a business field.
Search request	An abstract description of what exactly is searched. A search request contains a search target and search parameters. The back-end search engine executes a search request on the appropriate data source(s) and returns a result set.
Result set	The result of executing a search request: a set of all data records in the data source that conform to the search parameters.

References

[Graham02] Graham, I.: *A Pattern Language for Web Usability*, Addison-Wesley, 2002, http://www.wupatterns.com/

[Laakso03] Laakso, Sari A.: *User Interface Design Patterns*, http://www.cs.helsinki.fi/u/salaakso/patterns/

[Tidwell99] Tidwell, J.: *Common ground: Pattern Language for Human-Computer Interface Design*, http://www.mit.edu/~jtidwell/interaction_patterns.html

[Tidwell04] Tidwell, J.: *UI Patterns and Techniques*, http://time-tripper.com/uipatterns/

[Tidwell05] Tidwell, J.: *Designing Interfaces*, O'Reilly, 2005

[Welie03] Welie, M.: *GUI Design patterns*, http://www.welie.com/patterns/gui/index.html

[Wellhausen04] Wellhausen, T.: *Query Engine – A Pattern for Performing Dynamic Searches in Information Systems*, in: Proceedings of the European Conference on Pattern Languages of Programming (EuroPLoP) 2004, http://www.tim-wellhausen.de/papers/QueryEngine.pdf