

Data Management Information Architecture A Page Flow Pattern Language

White Paper produced by
Mphasis Corporation

March 2005
Version 1.1

© Mphasis. All rights reserved

Document Contact Information

Name	Bert Hooyman, Senior Architect Mphasis UK Ltd. Southbank House Black Prince Road London SE1 7SJ, United Kingdom
Email	bert.hooyman@mphasis.com
Business Phone	+31 618 780 559
Business Fax	+31 302 967 801

Table 1 **Contact Details**

Note From The Author

The page flow pattern described in this white paper is the result of exposure to numerous recurrences of the same data management requirements. We believe the pattern language covers all the essentials of data management. Having said that, we find that reality is always more complex than we are led to believe. Many of these 'reality checks' have found their way in this white paper. We invite all readers to actively contribute additional reality checks and contradictory findings, with the purpose of further enhancing the quality of this work.

Acknowledgements

I have had much help and valuable feedback from the following Mphasis UET Practice members: Pre eti Shenoy, Rohitashwa Jain, Malay Nagda and Ramesh Krishnan. I thank them all for helping put this white paper together.

Changes in this version

For version 1.1 of the white paper, the original use case title "Browse Whatever" has been replaced by "Filter Whatever". As it turned out, the verb *browsing* caused more confusion than expected, and the alternative *filtering* was suggested as a more intuitive description. As a result, some changes have been made throughout this white paper, including a rename from Browse Page to Filter Page.

Furthermore, the 'clear' request, coming from the Edit page, has been omitted. In talking with the usability engineers of Mphasis, it was found that there is no sensible business case for a form clear feature, despite the availability of a standard HTML clear button element.

Contents

Document Contact Information	i
Note From The Author	i
Acknowledgements	i
Changes in this version	i
1 Executive Summary	1
Disclaimer	1
Audience & Objectives	1
Prerequisites	2
Further Reading	2
2 Business Requirements	3
3 Solution Outline	4
3.1 Creation & Modification of Whatever	4
3.2 Viewing Whatever	5
3.3 Searching for Whatever	8
3.4 A Page Flow Pattern for Manage a Whatever	9
3.5 Page Flows for Pattern Variations	10
3.6 Relevant Commands for Manage a Whatever	13
4 Mphasis Case Study - User Profile	15
4.1 Business Objectives	15
4.2 Requirements Overview	15
4.3 Mphasis Approach	16
4.4 Application of the Pattern Language	16
4.5 Bottom Line	21
5 Conclusions	22
Appendix A - References	23
Appendix B - Use Case Synopsis for Manage a Whatever	24

List of Figures

Figure 1	Page Flow for Create, Copy & Update Use Cases.....	5
Figure 2	Page Flow for List Whatever	6
Figure 3	Page Flow for View a Whatever	7
Figure 4	Page flow for Find a Whatever	8
Figure 5	Page Flow for Filter Whatever	9
Figure 6	A Filter page	9
Figure 7	Page Flows for Manage a Whatever	10
Figure 8	Page Flow Variation for Limited Domains.....	10
Figure 9	Another Page Flow Variation for Limited Domains.....	11
Figure 10	Page Flow Variation for Limited Final Domains.....	11
Figure 11	Page Flow Variation for Unbrowsable Domains.....	11

Figure 12	Page Flow Variation for Read-only Domains.....	12
Figure 13	Page Flow Extension for Cascaded Filtering	12
Figure 14	Page Flow Extension for Master-Detail Domains	13
Figure 15	Page Flow for Manage a User Profile	17
Figure 16	Organization attributes for Employee Edit page.....	18
Figure 17	Contact details for Employee Edit page.....	18
Figure 18	Intranet Site Preferences for Employee.....	19
Figure 19	Entitlements for Employee	19
Figure 20	Edit Page with open and closed sections.....	20

1 Executive Summary

This MphasiS white paper addresses the management of data in web-enabled business applications. The scope of this research is all those parts of a web-enabled application that are light on behavior and heavy on data. This includes, but is not limited to, screen population, data validation and entitlement management. In many cases, the primary business function of an application is indeed little more than data management. An example of this is the web-enabling of corporate ERP solutions, where the business process automation, the workflows and the resource planning activities are centralized in the ERP system but data entry and query is pushed out to the corporate intranet.

In an earlier whitepaper¹, we presented a requirements analysis and generally applicable use cases. These use case together describe a pattern which we refer to as "Manage a Whatever". Now, we turn to Information Architecture and identify canonical web pages and a Page Flow Pattern that applies to Manage a Whatever. Once a recurring pattern of pages and page flows is discovered, the derivation of implementation effort is linked directly to the Manage a Whatever pattern.

Disclaimer

This paper does not cover any recurring patterns in GUI design. Firstly, the pattern language refers to "whatevers", and we really do not know what these whatevers are or what attributes make a whatever. Hence, we do not attempt to discover any patterns at that level. Secondly, a very useful collection of GUI design patterns has been presented earlier, for instance in *Patterns in Java, Volume 2* (see ref. [8]), or on the web (see for example Martijn van Welie's patterns in interaction design, ref. [9]). We believe we do not need to go into GUI design patterns again. Having said that, there is still a relevant role for such patterns in the context of Manage a Whatever, especially when the pattern language is used as a basis for code generation or parameterized design.

Audience & Objectives

This white paper is written for MphasiS clients. It describes a pattern language for defining the high-level information architecture (the canonical web pages as well as the page flow) of data management tasks. Using the pattern language, MphasiS information architects can quickly establish a baseline user interface for data management applications, trigger a creative process and identify opportunities for enhanced interaction. Through usability engineering based on a recurring pattern of user interactions, MphasiS' information architects are better equipped to deliver satisfying user experiences in an early stage of a client project. The use of a pattern language allows enhanced collaboration with MphasiS business analysts, IT architects and software designers.

¹ See reference [1]

The goal of this white paper is to introduce a formal analysis & design method for the recurring interaction styles around data management. The method is based on a pattern language that helps defining the precise business requirements. Using the pattern language, we hope that MphasiS business analysts and information architects find a common medium to express their findings and thoughts.

Prerequisites

This white paper uses a page flow diagramming method that we have found very useful – expressive and compact. It is elaborated in more detail in reference [2]. We don't expect that any of MphasiS' clients would have any difficulties understanding the diagrams.

The use cases of Manage a Whatever are described in detail in reference [1]. They are included in a brief format in this white paper as Appendix B - "Use Case Synopsis for Manage a Whatever" (page 24 onwards).

Further Reading

In Appendix A - References, we include some literature references that we have found valuable.

The page flow of the case study elaborated in chapter 4, as presented in Figure 15 on page 17, is available as an HTML mockup for reference purposes.

2 Business Requirements

Requirements for data management frequently show up as 'screen population' requirements in complex data-entry environments. To support the data entry task at hand, reducing data entry errors and ensuring valid data entry, many user interface designs focus on selecting values from enumerations, instead of letting users type in codes, names or numbers.

This approach is further encouraged by the desire to derive management information from the data that was entered. For the purpose of MIS, it is essential to work with enumerated lists. Reports that show "Downloads by country", or "Purchases by segment" require that the lists of countries and segments, respectively, are under control of the application used to enter the information and not under control of the person entering the data.

Implicit in the above requirements analysis is the assumption that the system under study will provide a means to manage the list of countries and the list of segments. Some of these lists are of a static nature and really do not require any administrative support at all. A list of countries is a good example of this².

In the case of consumer segmentation, the list of segments is usually under control of a marketing organization. Changes to the segmentation may come with the season. Administrative access to the list of segments becomes part of the functional requirements of the system under study. Now, the functional requirement that 'visitors can select a profile from a given list' suddenly implies 'marketing staff can add, modify and remove profiles from the segment list'. During the requirements analysis phase, this explodes into one or more additional use cases that cover the administrative access to the segments list.

Look for data management requirements wherever the system under study identifies enumerations. Screen population requirements and data validation requirements are two natural places to find enumerated values. More master data may be present in the access control part of the system under study. Roles, entitlements, resources, users and groups can be qualified as master data in many cases. The source of master data is not really important at this point - it can either be stored in a database designed to support the system under study, or in an existing (legacy) database, a registry, a directory or made available as a Web Service. The requirements for data management may exist for any of those.

Beyond master data as identified above, many web-based applications are little more than data management frontends to (legacy) systems. In these cases, all the primary business objects are further candidates for the Data Management pattern language.

-
- 2 Many (web) applications let users identify themselves and register with address details. A list of countries is typically provided as part of the data entry screen. The list does not change very often at all – it is either encoded directly in the application logic or pulled from a database table that is managed through the administrative interface of the database application itself. The task of updating the countries list is delegated to the application support team or the database administrator, respectively.

3 Solution Outline

In chapter 2, we have seen why many applications require data management functionality beyond their primary application goal. Once the need is established, there is actually a *pattern of use cases* that may occur. The pattern encompasses the full lifecycle of the data and includes creation, query, update and delete (CRUD) as well as various listings, sorting, printing, exporting and more.

Interestingly, the pattern applies almost universally to all types of data, and indeed in many cases also to the primary *Domain Objects* of the system under study. In terms of object-oriented design, the pattern applies to all those objects that are light on behavior and heavy on data. Given the general applicability of the pattern, we refer to it as *Manage a Whatever*. In his excellent book "Writing Effective Use Cases" (see reference [3]), Alistair Cockburn identifies these same use cases as Manage a Frizzle. We have discussed the use cases of Manage a Whatever in a separate white paper, see reference [1]. I will use the work on Use Cases for Manage a Whatever as a basis and look for a pattern in information architecture here.

In this chapter, I will look at the three principle areas of user interaction, which are:

- data modification
- data viewing
- data search and selection

In all, five canonical web pages will be identified that cover all the functional requirements expressed in the use cases of Manage a Whatever. In section 3.4, the flow of control from page to page will be studied, resulting in a *recurring page flow pattern* for data management.

3.1 Creation & Modification of Whatever

Three use cases apply to instance creation and modification; these are "Create a New Whatever", "Copy a Whatever" and "Update a Whatever". A single web page addresses the needs of all functional requirements around these use cases.

The Edit Page

The Create, Copy and Update use cases have one thing in common – they all provide form-based data entry functionality. There is hardly any difference between updating a Publication or creating a new Publication³.

³ If there is any difference at all, it is in how a unique identifier is associated with each new instance. In some scenarios, the unique identifier is a natural key, which means it is part of the data entered by the user. ISBN is a good example of a natural key (for Publication). In other cases, the instance identifier is created by the underlying database when the new instance is inserted in the database. The value of the identifier doesn't play any role for the user. This is the case for Publisher, Keyword, Author, Category and Subject. For none of those the instance identifier is ever shown to the library users.

It makes sense to design a shared data entry page, the Edit page, for all three use cases. The scope of the Edit page is to assist the user in creating or modifying entity attributes for one single entity instance at a time. Natural commands that are part of the Edit page include:

Save – store the new or updated entity in the underlying database (or other persistence mechanism). When the operation fails, revert to the Edit page with details on the problem that has occurred. When the operation succeeds, show the new data on either the View page or the List page.

Cancel – ignore any data entry on this page and revert to the previous page in the user interaction.

Clear – erase all data entry fields and continue on the current Edit page.

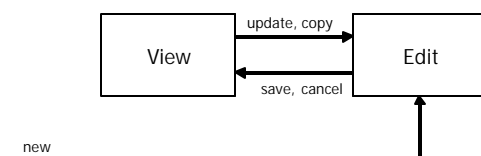


Figure 1 Page Flow for Create, Copy & Update Use Cases

The vocabulary used in Figure 1 is developed by Jesse James Garrett (see ref. [2]). The labels on all arrows are *user requests*, also known as *commands*. In many cases, these commands relate one on one to some use case. In a web-based data management application, the commands are the server-side actions that direct the flow of control and execute the business logic of the application. In section 3.6, the full command set of Manage a Whatever will be discussed in more detail.

3.2 Viewing Whatever

There are two principle ways to view information in a data-intensive web site; these are the list format and the single-instance display format. I present both page types in this section.

There are many variations on the theme of data viewing; including combined list/instance views, multi-page instance detail views, master/parent relationships with one parent instance combined with a list of detail summaries and single-instance display pages that are folded into the edit page.

The List Page

The result of “Find a Whatever” and “Filter Whatever” in many cases will be a *result set*. The List Page is used to display multiple instances of an entity in a list format. The same page also implements the “List Whatever” use case. A versatile and complete List page is really the heart of all data management – it is an element of the user interface that is part of all variations on the “Manage a Whatever” theme. Commands that apply to the List page are:

View – drill down from list view to entity view. The selected item is shown in more detail on the View page.

List – redisplay the same result set. Sort attribute, sort order and paging options may be used to control the content as follows:

Sort – sort the list in ascending or descending order by the selected column.

Re-display the List page, starting on page 1.

Page – apply one of the paging commands and re-display the List page:

- First Page – show the first page of items
- Last Page – show the last page of items
- Previous Page – show the previous page of items
- Next Page – show the next page of items

Note that Page commands are only available when there are more items in the List than what is shown on the page.

Value – attributes displayed in a list can be hyperlinks used to navigate to another selection of entities sharing the attribute value.

Note that this style of navigation always leads to success scenarios because the implicit Find will definitely return at least one match. This behavior makes it a very 'friendly' navigation style.

Service – apply a service to selected items on the list. This requires a selection mechanism on the List page (checkboxes). Upon completion, the List page is displayed again.

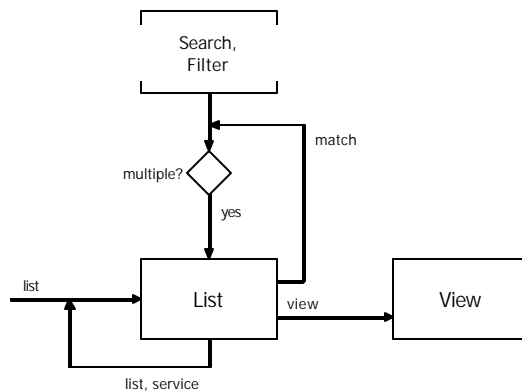


Figure 2 Page Flow for List Whatever

The View Page

The View page appears in most of the page flow diagrams in this chapter. It is the common display page for a single entity and as such it used to display the results of a search or filter when only a single instance matches the search. The View page may also be the target for any hyperlinked item in a List page. Commands on the View page are:

Copy – copy (part of) the entity attributes into an Edit page and initiate the creation of a new entity instance from there. Upon save of the new entity data, use the View page to display the new instance details. Upon cancel, revert to the View page with the previously shown entity instance.

Update – navigate to the Edit page with the current instance’s data for update.

Upon save, use the View page to display the updated details of the entity instance. Upon cancel, revert to the View page with the previously shown entity instance.

Delete – delete the currently displayed instance, possibly after a confirmation dialog (are you sure?..). Upon successful delete, revert to the View page with the next instance displayed in case a natural next instance is available, or with any other instance in case no natural ordering applies. Upon a cancelled or failed delete, re-display the original entity instance.

View – re-display the same page. Paging options may be used to control the content as follows:

Page – apply one of the paging commands and re-display the View page:

- First Page – show the first item
- Last Page – show the last item
- Previous Page – show the previous item
- Next Page – show the next item

Note that Page commands are optional – they should not be used in cases where there is no natural ordering of instances.

Value – attributes can be hyperlinks used to navigate to a selection of entities sharing the attribute value.

Note that this style of navigation always leads to success scenarios because the implicit Find will definitely return at least one match. This behavior makes it a very ‘friendly’ navigation style.

Service – apply a service to the instance. Upon completion, the View page is displayed again.

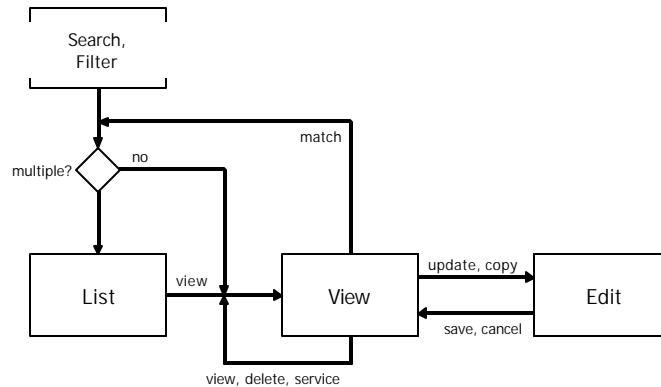


Figure 3 Page Flow for View a Whatever

Note that for very complex data types, there may be more than one View Page. This aligns with the well-known paradigm of Model-View-Controller; multiple views are possible on the same model. A familiar appearance of this phenomenon is the organization of entity data across multiple tabs in a *tabbed GUI*. The views visually

appear as one whole, but the implementation is really a group of individual view pages.

3.3 Searching for Whatever

There are two principle ways to search for a data item of interest, either through a targeted (form fill-in) search or by filtering the data space, drilling down on information categories. The associated use cases are “Find a Whatever” and “Filter Whatever”. In this section, I introduce the web pages that deliver on these use cases.

The Search Page

The Search page provides data entry fields for all searchable attributes of the entity. A form-based search is the most natural way to specify search parameters; when more than one search attribute is provided, a logical AND is applied.

There is only one command associated with Search which is:

Find – apply the search. A search results in either zero, one, or more than one matching entities. When no matching entities are found, revert to the search page informing the user that no matching entities were found. When a single matching entity was found, proceed to the View page (described in section 3.2 above). When more than one matching entity is found, proceed to the List page.

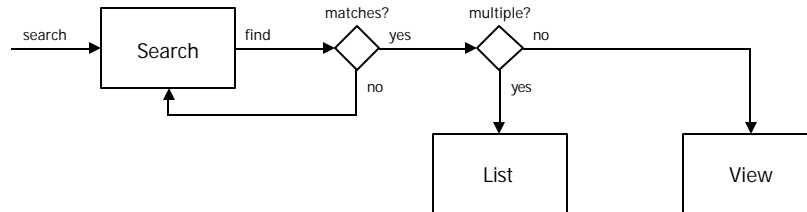


Figure 4 Page flow for Find a Whatever

The Filter Page

The Filter page provides, for one attribute, all values of that attribute that are referenced by at least one entity instance. It should *not* provide a full list of all defined values but only those that are actually used. We’re not listing the attribute domain here, we’re filtering an entity and displaying the result of a matching operation.

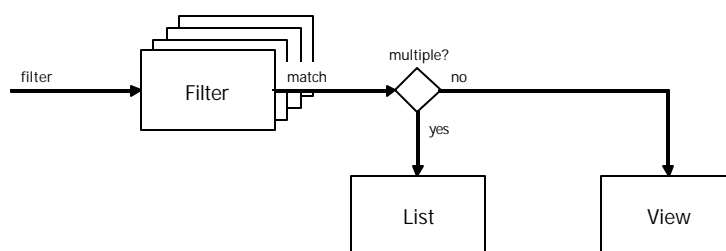


Figure 5 Page Flow for Filter Whatever

The display of attribute values typically uses a multicolumn table, because attribute values are often short (10-20 character) labels. In Figure 6 below, the “Filter Catalog by Category” page from the Library project is shown.

Catalog index by category			
To browse the catalog, select a category from the list below.			
Algorithms	Compilers	Computer-Aided Design	CRM
Database Technologies	Distributed Computing	E-Commerce	Enterprise Resource Planning
Financial Services	Graphics	Hardware	ITIL
Modeling & Design	Operating Systems	Operations Research	Other non-fiction
Physics	Product Workbook	Programming Languages	Social Impact
Software Engineering	Supply Chain Mgmt & Procurement	Telephony & ISDN	Typography
User Interfaces	Web Technology	Webdesign	Workflow Management

Figure 6 A Filter page

On the Filter page, all individual attribute values are hyperlinks that trigger the display of a (list of) matching entity(-ies). In Figure 6, hyperlinked words are shown in blue, not underlined.

There are no ‘commands’ on the filter page; the only hyperlinks are the attribute values themselves. For a *cascaded filter*, a hyperlink on a Filter page may lead to another Filter page. In the Library, this is the case for “Filter Catalog by Subject”. When a subject is chosen, all Categories related to that subject are shown. When a category is subsequently selected, the list of publications matching that category is displayed.

3.4 A Page Flow Pattern for Manage a Whatever

In sections 3.1-3.3, the principle web pages for data management have been identified. The diagrams in these sections have shown how the various user requests (the *commands*) dictate a flow of control from one page to another. The combined page flow for the five data management pages is shown in Figure 7.

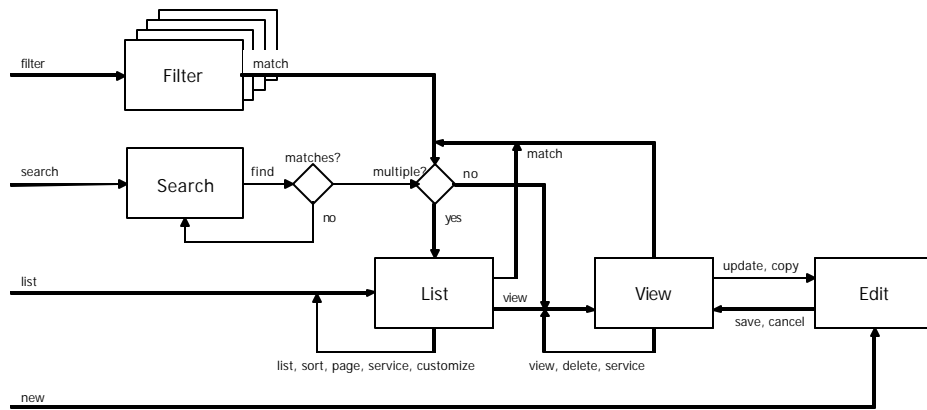


Figure 7 Page Flows for Manage a Whatever

There are four commands that come in from the left-hand side of the diagram. These commands are valid throughout all pages; these commands are generally implemented as a set of buttons or menu options that are always visible. All other commands are associated with one or more arrows coming out of the pages – these are contextual commands. An example is update, which is a command that is only available on an arrow coming out of the View page. The reason for this is that the Update a Whatever use case requires a single instance of whatever to be in context, and View is the page that satisfies that criterion. So from the View page, any command that applies to a single instance is valid. Besides update, this includes copy, delete and service. The names of these commands relate directly to the associated use cases.

3.5 Page Flows for Pattern Variations

The page flow shown in Figure 7 represents the ‘standard’ page flow for the Manage a Whatever pattern. In reference [1], a number of variations on the baseline pattern are described. Here, I show these variations as page flows.

Limited Domain

A limited domain is characterized by the absence of Search and Filter pages. The Copy, Next and Service use cases are not implemented, either.

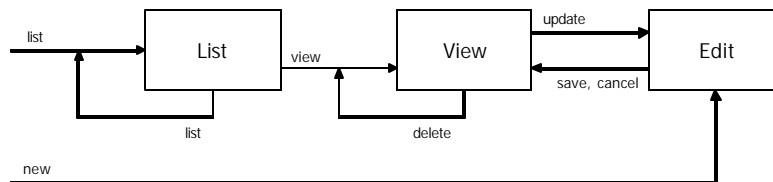


Figure 8 Page Flow Variation for Limited Domains

Limited Domain without View Page

This is an even simpler implementation of a Limited Domain – there is no View page. This leaves only two pages for the implementation of the Create, Update, Delete, List and Sort use cases.

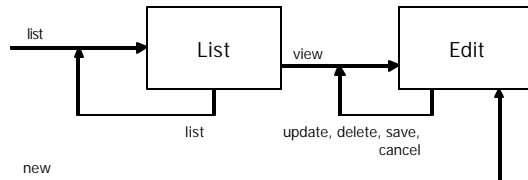


Figure 9 Another Page Flow Variation for Limited Domains

Final Domain

A Final Domain is very much a complete implementation in terms of the required pages, but it lacks the Update and Delete commands.

Limited Final Domain

A limited final domain is implemented using three pages, same as the Limited Domain. The difference is that this variety lacks the Update and Delete commands.

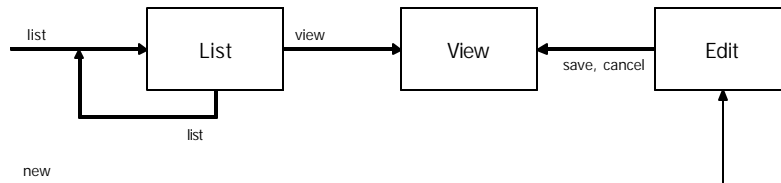


Figure 10 Page Flow Variation for Limited Final Domains

Unbrowsable Domain

This is a four-page implementation of Manage a Whatever, lacking the Filter pages. This variation also lacks the hyperlinked attribute values in the List and View page.

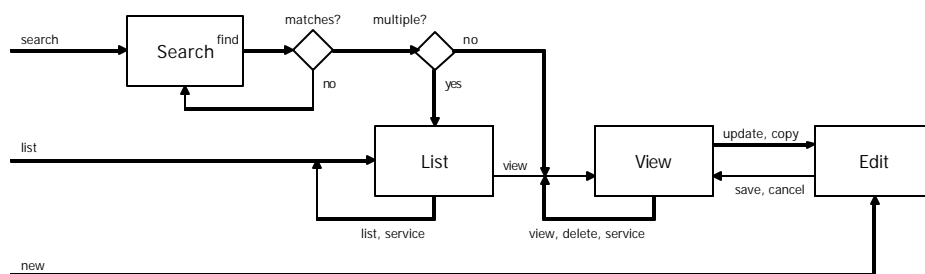


Figure 11 Page Flow Variation for Unbrowsable Domains

Read-only Domain

A read-only domain has all the list, search and filter capabilities, but lacks the Edit page and the New, Update, Copy and Delete commands.

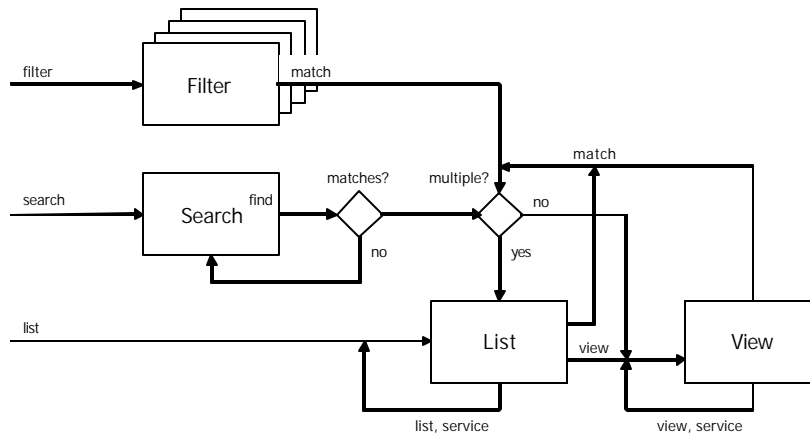


Figure 12 Page Flow Variation for Read-only Domains

Cascaded Filtering

This is an extension of Filter that applies to all 'filterable' variations of Manage a Whatever.

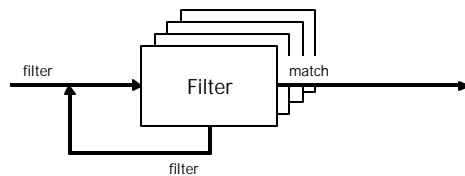


Figure 13 Page Flow Extension for Cascaded Filtering

Master-Detail Domain

This variation requires the linking of some of the Detail use cases with one or more Master use cases, as shown in Figure 14. Here, I show how a Limited Domain without View Page is used for the Detail entity. It is linked to the View page of the Master entity.

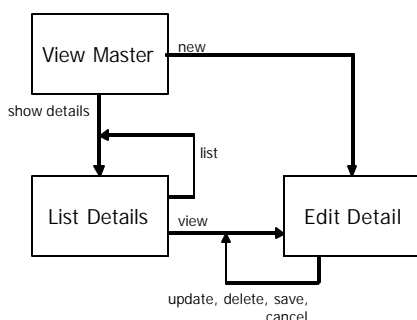


Figure 14 Page Flow Extension for Master-Detail Domains

3.6 Relevant Commands for Manage a Whatever

As part of my analysis of web-enabled data management, I have identified five canonical web pages that together realize all required user interaction features for Manage a Whatever. In the web page description sections (in sections 3.1 through 3.3), I have made reference to *relevant commands*, and the page flow diagrams in previous sections show many of those commands. In the context of a web-enabled application, these commands or *user requests* are sent from the user's browser to the application (server). They are processed there and as a result another web page is displayed.

I will now tally all these requests and provide a more formal definition of each one of them.

- list** – show entities on the List page. This command is valid anywhere in the context of the entity being managed. The default command would list all entities, applying paging if needed. Additional qualifiers can be used to control the attribute to sort by, the sort order, the page size and the page number to show.
- view** – show an entity on the View page. This command is used on hyperlinks in the list page and following a filter or search action that resulted in a single matching entity. Additional qualifiers can be used to control which entity to display; this realizes paging on the View page. Paging commands may be used on the View page.
- search** – display the Search page. This command is valid anywhere in the context of the entity being managed.
- find** – apply search criteria as provided. When no matching entities are found, re-display the Search page. If a single match is found, display the View page (or the Edit page when no View page is used). When multiple matching entities are found, use the List page.
- filter** – display the Filter page using a specified attribute. Show all values of the attribute that are in use. This command is valid anywhere in the context of the entity being managed. Additional qualifiers can be used to control the attribute selected for filtering. This supports cascaded filtering.

- match** – select entities matching a given attribute's value. If a single match is found, display the View page (or the Edit page when no View page is used). When multiple matching entities are found, use the List page.
- new** – start creation of a new instance. Display the Edit page without any data population. This command is valid anywhere in the context of the entity being managed.
- copy** – display the Edit page pre-populated with data copied from the current entity instance. Knowledge of which attributes should or should not be copied is entity-specific. This command is only valid in the context of a single instance (i.e. only one instance shown or selected).
- update** – start modification of an existing instance. Display the edit page populated with the instance data. This command is only valid in the context of a single instance (i.e. only one instance shown or selected).
- save** – persist the data entered on the Edit page. This is either an update or an insert, depending on how the Edit page got displayed. New and copy lead to insert, update leads to update. The save command is only valid on the Edit page. Data is verified for completeness and correctness before committing to the physical storage. Any validation errors lead to a re-display of the Edit page with the errors shown.
- cancel** – interrupt the ongoing edit action. Revert to previous page. This command is specific to the Edit page.
- delete** – remove the current instance from persistent storage. A confirmation dialog may be used before the physical removal is started. Referential integrity constraints may result in failure to delete. This command is only valid in the context of a single instance (i.e. only one instance shown or selected).
- service** – execute an application (or entity) -specific command. Valid on the List page, the View page and the Edit page. Command execution is typically a side-effect; upon completion the original page is redisplayed, possibly with data changes.

4 Mphasis Case Study - User Profile

4.1 Business Objectives

Our client wants to realize a corporate intranet (a Business-to-Employee or B2E intranet). The company has decided to use an elaborate *User Profile* as the basis for information dissemination to its employees.

Through the profile, employees can mark their interest in InfoStreams, which are channels of company information, and in Communities, which are teams of employees that span across organizational units (think sports groups, first-aid units, ride-sharing communities). The company offers a multitude of information delivery capabilities, including office email, instant messaging, voice messaging, short text messages (SMS) and text pagers. Employees can use their profile to indicate their preferred delivery channel(s).

Employees shall use the User Profile to maintain their contact details, including office details, mail stop, phone extension and fax number.

The user profile is also used to register entitlement of employees for access to restricted areas of the intranet. A role-based access control system is used for this, and team leads can assign or withdraw roles to their direct reports (and down).

Employee ID's are used to link the user profile to the corporate HR system. The user profile is an extension of the employee data held in the HR system. Data held in the HR system, which is not editable in the user profile, includes:

- Employee ID – a globally unique employee identifier
- Employee Name – name of the employee as held in the HR system
- Job Function – one of an enumerated list of functions as managed in the HR system
- Division – top-level organizational units of the enterprise; these may span across multiple geographies and are further subdivided in Departments
- Department – one of an enumerated list of functional departments (e.g. sales, marketing, support, r&d, production, finance, operations....)
- Direct Reports (if any) – references to employees that report to this employee (optional, can be many)
- Report-to (manager) – reference to immediate report -to, i.e. the direct manager.

4.2 Requirements Overview

- [1] The User Profile holds user-editable profile attributes as well as non-editable attributes
- [2] Users of the application include all employees of the enterprise
- [3] Managers can update the entitlements of their staff (direct reports as well as below). Entitlements are based on predefined roles.
- [4] Any user can update her/his preferences, including preferred notification channel, InfoStreams subscriptions, Community memberships and language.

- [5] Any user can browse the organizational hierarchy, search for any other employee by any attribute
- [6] Users shall manage their own office location details, including postal and delivery address. The system shall provide a list of predefined office locations to choose from (users extend those locations with in-building coordinates, phone extensions and mail stop numbers)
- [7] Site administrators create new user records. Only site administrators can update user email addresses and username/password details for access to the B2E intranet.

4.3 MphasiS Approach

Based on the above requirements, MphasiS identified the following (Manage a Whatever) use cases:

Create a Profile – restricted to site administrators

Copy a Profile – restricted to site administrators

Delete a Profile – restricted to site administrators

Update a Profile – depending on the user's role and the relationship of the user to the employee whose data is being updated, some attributes may or may not be editable (site administrators have access to all attributes held in the profile but cannot edit their own roles, managers can update their own profile plus the roles of their staff, other users can update only their own profile)

Filter by Geography – Cascaded Filter from Region via Country and City to Office Building

Filter by Organization – Cascaded Filter from Division to Department

Filter by Function – A straight filter

Filter by Role – Another cascaded filter, this time from Application to Role. These are the entitlements-driven ACL roles

Search Employee – Search on any attribute or combination thereof

List Employees – Defaults to 'My Department', i.e. all employees that are in the same department of the division this user works for. Requires paging and sorting.

View a Profile – Folded into Update a Profile. Requires paging.

4.4 Application of the Pattern Language

Based on the identified use cases, there is a need for:

- An Edit page, which doubles as a View page. Given the fact that Profile is part of a master/detail relationship with Direct Reports as well as with Roles, the Edit page might include 'List Direct Reports' and 'List Roles'. Given the large number of profile attributes, the Edit page might be implemented using multiple pages (tabbed GUI) or through some means of progressive disclosure. A paging function to step through a list of employees is desirable.
- A List page, which shows lists of employee summaries. Candidate attributes for the summary include name, division, department, office city, office country. The list page must provide paging as well as sorting.

- A Search page, which allows a form -based search on any combination of profile attributes
- Nine Filter Pages, five of which result in another filter (i.e. cascading). The other four result in a list of matching employees (using the List page) or a display of one single employee profile (using the Edit page).

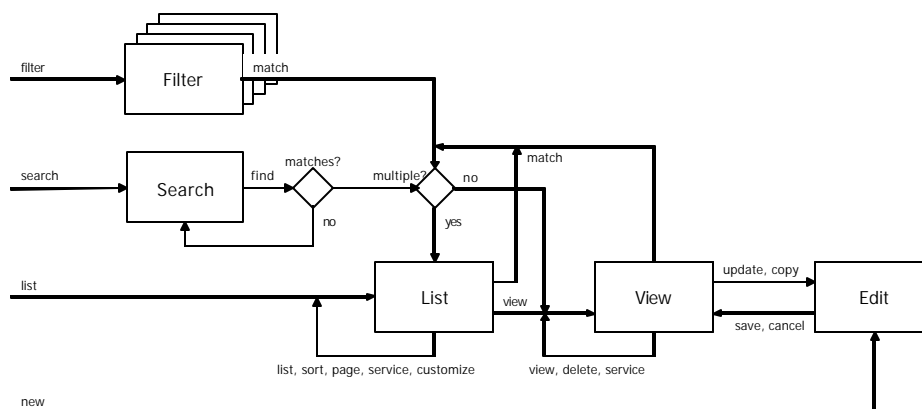


Figure 15 Page Flow for Manage a User Profile

Any filter action ultimately results in the display of either a single employee's details or a list of matching employees. Likewise, any search action results in either a detail or a list display. There are four globally available navigation links (global in the sense that these are present on all pages of *Manage a User Profile*). The *list* link does not show a full list of all corporate employees, but instead goes through a search for all employees in the same department ("My Department").

From the List page, users can navigate to previous and next pages of the list, they can click on an attribute value (e.g. Job Function) which results in a list of all employees sharing that same value, or they can select one employee from the list, resulting in a display of the Edit (User Profile) page.

Editing the User Profile

As indicated above, the Edit page of User Profile is by far the most complex part of the information architecture. The page must support various levels of edit entitlements (ranging from access to almost all attributes to none at all), there is a substantial amount of information on the page, and there are two master/detail relationships to be taken care of (Role and Report-To). In addition, there are at least eight distinct commands that apply to a User (labeled as clear, copy, cancel, save, delete, previous, next, value in the diagram).

In a sample implementation of the User Profile GUI, we have used a progressive disclosure form of display. We group the attributes in four subsections which are dynamically disclosed and hidden.

Organization Attributes of Employee

The report-to details are non-editable, as these relationships are maintained exclusively in the HR system. Together with all other read-only attributes from the HR system, this chunk of information is grouped under “Organization”.

- Organization

Name: Alwyn J. Roches
 Employee ID: 46-5522183
 Division: Printing Paper
 Department: Procurement
 Job Function: Contracts Negotiator
 Manager: Jocelyn McReddy

Reports:

Name	Function	Email	Extension
Fred White	Jr. Negotiator	f.white@ourcorp.com	766-2432
Mary J. Lopez	Jr. Negotiator	m.j.lopez@ourcorp.com	766-2455
Susan James	Jr. Negotiator	s.james@ourcorp.com	766-2487

Figure 16 Organization attributes for Employee Edit page

Although the page is an Edit page, the Organization attributes are all non-editable. This part of the page is a View page for all users, irrespective of their role.

Contact Details of Employee

The contact details of an employee are editable by the employee her/himself, and by the system administrator, but not by any other user. Hence, this part of the Edit page is either shown as an editable form or as a read-only View page. We show the editable form in Figure 17.

- Contact Details

email: a.j.roches@ourcorp.com

extension:

mobile:

fax:

pager:

assistant: Valery Thurston, v.thurston@ourcorp.com, ext. 766-2401

office:

region:

country/state:

city:

office:

office mail address:

floor/location:

Figure 17 Contact details for Employee Edit page

For the specification of Office, a top-down approach has been used. The employee first enters the region, and the interface populates the country/state select box with values matching the selected region. This continues down to the office level,

which provides a street address indication. Once that is selected, the associated mail address is set by the application (non-editable). The employee completes the office information with an indication of specific floor/location details within the building.

Site Preferences for Employee

In the third section, employees can specify their preferences with respect to the use of the B2E intranet. Again, only the employee herself or the site administrator can update these attributes.

The screenshot shows the 'Intranet Site Preferences' form. It has a title bar '- Intranet Site Preferences'. The form contains several sections:

- Language:** A dropdown menu set to 'English UK'.
- Preferred Notification:** Three radio buttons: 'Email' (selected), 'Voice or voicemail', and 'Instant Messaging'.
- Communities:** A list box containing 'ride-sharing', 'first-aid', and 'site designers'. Below it is the text '(select as many as you wish)'.
- InfoStreams:** A list box containing 'Reviews', 'Refurbishments', 'Demonstration Events', and 'In the press'. Below it is the text '(select as many as you wish)'.

Figure 18 Intranet Site Preferences for Employee

Entitlements

The Entitlements section of the Employee Edit page includes the user entitlements for all applications accessible from the intranet as well as the user's login identity and activation switch. As the list of applications is finite, the design for entitlements is based on a full listing of all applications, and for each application all possible roles are shown as radio-button choices. The advantage is that administrators and line managers (who are entitled to make changes to employee entitlements) are exposed to all the options and settings in one single view.

The screenshot shows the 'Entitlements' form. It has a title bar '- Entitlements'. The form contains the following fields and sections:

- Username:** a.j.roches
- Password:** A text box with masked characters (dots).
- Active:** A checked checkbox with the text 'This employee is an active intranet user'.
- Roles:** A table with columns 'Application' and 'Roles'.

Application	Roles
Application 1	<input type="radio"/> No Access <input type="radio"/> Regular User <input type="radio"/> Application Administrator <input checked="" type="radio"/> Business Manager
Application 2	<input type="radio"/> No Access <input type="radio"/> Regular User <input type="radio"/> Application Administrator <input checked="" type="radio"/> Business Manager
Application 3	<input checked="" type="radio"/> No Access <input type="radio"/> Regular User <input type="radio"/> Application Administrator <input type="radio"/> Business Manager
Application 4	<input type="radio"/> No Access <input checked="" type="radio"/> Regular User <input type="radio"/> Application Administrator <input type="radio"/> Business Manager

Figure 19 Entitlements for Employee

The details on this section of the Edit page are only editable for system administrators and line managers. Employees cannot edit their own entitlements.

Combining the Edit Sections

As stated, the sample design has selected progressive disclosure to combine all the edit sections on a single page. Plus signs and minus signs on the section labels are used to indicate the option of disclosure. In the closed state, an instructional text is added to highlight this feature. In Figure 20 the Edit page is shown with three of the four sections closed. The other page elements included on the edit page are:

- Navigation aids to search, filter, list or create employees,
- Paging aids to view the previous, next, first or last employee of a current list of employees (which may itself be the result of a search or a filter), plus a navigational aid to revert to that list,
- A help button in a standard location,
- Command buttons to delete or copy the employee details,
- Command buttons to save any updates or revert to the previous unaltered version,
- A command button to clear all editable attributes
- Two services, accessible through command buttons (print and download).

Not clearly shown is the fact that as a rule, any displayed value that is non-editable is a candidate for hyper-linking. As an example, consider the Department attribute of the Organization section. Clicking on the word 'Procurement' would result in a list of employees that all work for that same department. Likewise for Division, Job Function and many other attributes.

Corporate Intranet - Employee Profile for Alwyn J. Rochas

Organization

Name: Alwyn J. Rochas
 Employee ID: 46-5522183
 Division: Printing Paper
 Department: Procurement
 Job Function: Contracts Negotiator
 Manager: Jocelyn McReddy

Name	Function	Email	Extension
Fred White	Jr. Negotiator	f.white@ourcorp.com	766-2422
Mary J. Lopez	Jr. Negotiator	m.j.lopez@ourcorp.com	766-2455
Susan James	Jr. Negotiator	s.james@ourcorp.com	766-2467

+ Contact Details
 Click on the "Contact Details" label to display employee contact details

+ Intranet Site Preferences
 Click on the "Intranet Site Preferences" label to display user preferences for the corporate intranet

+ Entitlements
 Click on the "Entitlements" label to display this user's roles and entitlements

View list | First | Previous | Employee 7 of 32 | Next | Last

Clear Form | Data... | Copy | Save | Revert | Print | Download | Help

Figure 20 Edit Page with open and closed sections

4.5 Bottom Line

Based on the Use Case Pattern Language, business analysts can translate functional data management requirements into standard use cases. Through the Page Flow Pattern Language, the information architect can map use cases to canonical pages (list, edit, view...). Based on the use cases and the selected pages, the information architect can design a page flow that is most appropriate to the problem domain. Subsequently, the pattern language helps the information architect completing the design, ensuring completeness and correctness of the final information architecture.

5 Conclusions

In this white paper, we have shown how the use cases of Manage a Whatever map to a page flow design that ties all the data management goals together. We have applied a simple diagramming method, developed by Jesse Garrett, for the purpose of designing and documenting web page flows.

In the context of an implementation project, the findings in this report can be used to drive an effort estimation. Based on the requirements matrix that is put together as part of the requirements analysis (see reference [1], Conclusions), we can now identify exactly which pages will be required for the realization of these use cases. Please refer to the referenced document for a complete description of the sample scenario.

Another advantage of applying a well-understood pattern language is the ease of deriving page mockups that realize the page flow. Independent of the desired look and feel, the page flow pattern determines how pages are hyperlinked, which commands must be present on the various pages, and what information must be present on which pages. This is particularly effective when designing data management interfaces for master/detail type information.

Appendix A - References

- [1] "Data Management Requirements – A Use Case Pattern Language", Version 1.1, White Paper by Bert Hooyman, MphasiS UK Ltd, March 2005
- [2] "A visual vocabulary for describing information architecture and interaction design", Jesse James Garrett, version 1.1b, March 6, 2002, on the web at <http://www.jjg.net/ia/visvocab/>
- [3] "Writing Effective Use Cases", Alistair Cockburn, Addison-Wesley, 2001. ISBN 0-201-70225-8
- [4] "Designing Easy-to-use Websites", Vanessa Donnelly, Addison-Wesley, 2001. ISBN 0-201-67468-8
- [5] "Designing Data-Intensive Web Applications", Stefano Ceri, Piero Fraternali, Aldo Bongio, Marco Brambilla, Sara Comai, Maristella Matera, Morgan Kaufmann Publishers, 2003. ISBN 1-558-60843-5
- [6] "Web Database Applications with PHP & MySQL – Building Effective Database-Driven Web Sites", Hugh E. Williams and Davis Lane, O'Reilly 2002. ISBN 0-596-00041-3
- [7] "Web Application Design Handbook : Best Practices for Web-Based Software", Susan L. Fowler and Victor R. Stanwick, Morgan Kaufmann Publishers, 2004. ISBN 1-558-60752-8
- [8] "Patterns in Java, Volume 2", Mark Grand, John Wiley & Sons, 1999. ISBN 0-471-25841-5
- [9] "...patterns in Interaction Design", Martijn van Welie, on the web at <http://www.welie.com/patterns/>

Appendix B - Use Case Synopsis for Manage a Whatever

B.1 Create a New Whatever

Use Case Brief

In the context of data management, the ability to create new instances of a data entity are an important user goal. This use case addresses that goal.

Success Guarantees

A new instance of Whatever is persisted and made available to other application users.

B.2 Copy a Whatever

Use Case Brief

In the context of data management, the ability to create new instances of a data entity are an important user goal. One way to satisfy that goal is to (partially) copy an existing instance. This goal typically only exists for entities with many attributes or extended attribute collections.

Success Guarantees

A new instance of Whatever is persisted and made available to other application users.

B.3 Update a Whatever

Use Case Brief

Application data items may require modifications. Users access the data repository to update individual instances of data entities.

Success Guarantees

The updated instance of Whatever is persisted and made available to other application users.

B.4 Delete a Whatever

Use Case Brief

Data items may become obsolete and serve no further purpose in an application. Users access the data repository to remove individual instances of data entities.

Success Guarantees

The selected instance is removed from persistent storage.

B.5 Find a Whatever

Use Case Brief

When there are many instances of a particular type of data, Manage a Whatever requires a search mechanism to locate an instance or instances of interest. This is even more relevant in situations where the entity does not provide any filtering features.

Success Guarantees

One or more matching items are found.

B.6 View a Whatever

Use Case Brief

For data items with more than just a few attributes, an entity details display is used to present all the attributes on a single screen, or indeed on multiple (tabbed) screens when there are many attributes involved. View a Whatever satisfies the goal of showing this entity details screen.

View a Whatever brings a single instance of Whatever in context. This is a result of:

- “Find a Whatever” matching only one single instance,
- Selecting a Whatever from “List Whatever”,
- “Filter Whatever”, matching only one instance, or
- Committing a newly created, copied or updated entity instance.

Note that there is no globally visible “View a Whatever” command because this would always require an identification of the Whatever to be displayed. Hence “View a Whatever...” would be an option, followed by a simple “Find a Whatever” to enter a unique identifier.

Success Guarantees

The instance details are displayed.

B.7 Next Whatever

Use Case Brief

As part of data management, users can filter or search a particular domain of data items. Once they have identified a subset of interest, they wish to step through the instances one by one. This is referred to as *paging* and includes typical commands such as Next, Previous, First and Last. In small domains, paging may also occur on the full set of data items.

Success Guarantees

The details of the next (previous, first, last) instance are displayed.

B.8 List Whatever

Use Case Brief

As part of data management, List Whatever plays a pivotal role. List Whatever is almost always a globally visible command, allowing users to jump from anywhere within Manage a Whatever to the List Whatever page. Also, List Whatever is invoked as a result of “Find a Whatever” and “Filter Whatever”. Finally, List Whatever is the starting point for several other use cases, including “View a Whatever”.

Success Guarantees

The instance summaries are displayed.

B.9 Sort a List of Whatever

Use Case Brief

As an extension of presenting a list of Whatever, it is oftentimes useful to sort the list in a particular order, possibly as a prelude to “View a Whatever”. Search or filter the domain first, then sort the result set and select the first instance. Use “Next Whatever” to page through the result set in the desired order.

Success Guarantees

The sorted instance summaries are displayed.

B.10 Filter Whatever

Use Case Brief

Filtering is the complement of Find a Whatever. Rather than searching for a particular entity instance, “Filter Whatever” implements a guided selection of entities. For a particular attribute, FilterWhatever shows all the attribute values that are in use. The user selects one value of interest and as a result the system displays all matching Whatever. The difference with Find a Whatever is that Find uses data entry, whereas Filter only uses selection from value sets.

Success Guarantees

The results are displayed.

B.11 Service a Whatever

Use Case Brief

The “Manage a Whatever” pattern focuses on CRUD-style tasks for application data. Many applications require *additional* tasks that apply to data. As long as such tasks are limited to a single instance of Whatever, this use case can be used as an extension point for implementation of these tasks.

A typical example of this is “mark as read” which applies to an email message. It changes the state of one particular email message. Another example is “check availability” on an order line item. A situation that is not well covered by “Service a

Whatever" is "find the difference between these two email messages", because it requires two email messages to be in context, not just one.

"Service a Whatever" can also be implemented as an extension of "List Whatever", but the service being invoked would still apply to single instances (it would iterate over the set of instances that is selected).

Success Guarantees

The service is successfully completed on all instances.