Towards a Systematic Development of RIAs: Taxonomy for classifying Rich-User-Interface Components

Rosa Romero[1], David Díez[1], Susana Montero[1], Paloma Díaz[1], and Ignacio Aedo[1]

[1] DEI Lab - Computer Science Department
Universidad Carlos III de Madrid, Spain
rmromero@inf.uc3m.es, david.diez@uc3m.es, smontero@inf.uc3m.es, pdp@inf.uc3m.es, aedo@ia.uc3m.es

Abstract. The development of Rich Internet Applications (RIAs) is based on the selection, assembly, and tailoring of Rich-User-Interface (RUI) components. Since RUI component libraries have been widely spreading, the selection of appropriate components has become an essential and complex task. While the user interface design is usually guided by principles, guidelines, and heuristics, there are not artifacts or mechanisms for selecting RUI components in a systematic way. Moreover, there is a lack of homogeneous classification criteria that hinders the selection of components and increases the relevance of the experience designing web applications. To ease the search and choosing of components by web-developers, this paper presents a taxonomy for classifying RUI components. The development of such a taxonomy has been based on both the study of relevant resources from the UI domain and on the opinions of Subject-Matter Experts (SMEs).

Keywords: Rich Internet Application, component libraries, subject-based classification, taxonomy

1 Introduction

Rich Internet Applications (RIAs) are *'web applications that offer the responsiveness, rich features, and functionality approaching of that desktop application'* [8]. These capabilities are achieved by using Rich-User-Interface (RUI) components organized into libraries – defined as '*reusable components collection for applications development*' [7]- that have been widely spreading. Libraries from Java Server Faces (JSF) technology such as Richfaces[1] and Icefaces[2]; JavaScript libraries such as Dijit[3] and jQuery[4]; or components libraries belonging to Adobe Flex technology are examples of this proliferation.

[1] Website: http://www.jboss.org/richfaces
[2] Website: http://www.icefaces.org
[3] Website: http://api.dojotoolkit.org/jsdoc/1.2/dijit
[4] Website: http://docs.jquery.com/Main_Page

These libraries offer tens of components grouped by diverse criteria and related to different functionalities. Consequently, the search and choosing of the suitable component for a specific problem is not a trivial matter. As an example of this situation, we can highlight two different well-known components: *Progressbar*, a control component conceived to inform the user the progress of a task, and *Collapsible Panel*, a structural component that allow users to add collapsible sections of a web page. The *Progressbar* is classified into the '*Widget'* category in Dojo and into '*UI control'* category in Flex; similarly, JSF libraries do not use a homogenous criterion and classify this component into '*Component'* category in Icefaces and '*Rich output'* category in Richfaces. As far as *Collapsible Panel* is concerned, this component is classified as a container in Dojo – '*Accordion container*' category- and Icefaces – '*Layout panels*' category. On the contrary, Flex, JQuery and Richfaces consider the *Collapsible Panel* as a control, classifying it into '*UI control*', '*Widgets*' and '*Rich Output*' categories, respectively. This situation is reproduced in other widely used components such as *Dialog*, *Combobox*, *Menu*, *Calendar*, etc.

The component selection process aims at determining which component, among those available, is the most suitable to fulfill a set of requirements of the UI design. Therefore, the existence of classification schemes is required to be more efficient when using component libraries [3] [7] [9] [22], both during creation and maintenance, as well as in the selection of components. However, the experience in the use of RUI component libraries highlights, as aforementioned, the lack not only of homogeneous criteria for classifying RUI components but also of meaningful terminology and appropriate classification mechanisms. With the purpose of overcoming these limitations, we propose a taxonomy as such a classification artifact that allows categorizing objects in a hierarchical way as a manner of easing the search and choosing of RUI components.

This paper presents a taxonomy based on both the study of relevant resources from UI domain and the opinions of Subject-Matter Experts (SMEs hereafter). This taxonomy allows categorizing structural and control elements that may be part of Web User Interfaces (WUI). The rest of the paper is organized as follows. In section two, we deepen in the problem that has motivated of our work by studying different RIA platforms. Section three is focused on reviewing the proposals related to classification artifacts for UI components. In section four, we deepen in the concept of taxonomy and its characteristics. The development process of our taxonomy is described in section five; this process is divided into two levels – empirical and operational– in order to provide an iterative process that guarantees the appropriate definition of the taxonomy. The result of this development process, the taxonomy itself, is presented in section six. Finally, conclusions and intentions for further work are drawn in the last section.

2 Related Works

The continuous use of RUI component libraries has shown us the lack of either specifications or homogeneous classification mechanisms. Nevertheless, there are approaches in other domains - such as graphical-user interfaces (GUI) and desktop applications– that may be used as a reference to define RUI component classification.

A classification of controls based on user's goals is proposed in [6]. This classification defines four basic categories: *imperative controls,* used to initiate a function; *selection controls,* used to select options or data; *entry controls,* used to enter data; and *display controls,* used to display and manage the visual presentation of information on the screen. These categories therefore allows classifying controls but does not include other types of interface components, such as structural components – UI elements that designers use to organize the information- or advanced contextual components – for example, *breadCrumbs,* a navigation component that provides information about page hierarchy.

A generic dedicated widget vocabulary is proposed in [23]. This vocabulary includes four different types of elements as determined by the particular role they play within the UI: *basic elements* tag includes well-defined interface objects such as a buttons, scroll-bars, menu, etc.; *input elements* tag collects UI elements that allow entering data; *output elements* tag groups widgets that show information to the user in the form of text, graphics or audio; and *collection elements* tag collects conceptually related elements. This vocabulary provides helpful categories to classify basic UI components; however, it does not consider such rich and advanced components that characterize RIAs.

In the desktop applications domain, such as Java desktop community – including libraries as Swing[5]- it is commonly accepted to classify UI components as *controls*, which are interface elements users can manipulate to perform an action; and *containers* that are used to organize the information. These desktop-component libraries are the inspiration for web technologies such as JSF. JSF technology establishes the standard for building Java server-side user interfaces within the Java EE Platform[6]. Otherwise, Flex platform focuses its classification on *controls* and *media* components, which allow incorporating image and video in RIAs. The main agreement among these informal classification schemas is categorizing the most complex components such as 3D, audio, and charts components into little descriptive categories such as *Miscellaneous* or *Other Components*.

Based on this review of the literature, we have identified two kinds of classification artifacts that have been mainly applied with UI components: simple controlled vocabularies, '*which are compound of a closed list of named subjects, used for classification'* [10], and dedicated vocabularies, which are reserved for a specific use or purpose. Such kinds of schemes provide closed lists of terms that do not allow establishing hierarchical relationships between them. Taking into account that the UI is composed hierarchically [16], we propose the use of a taxonomy as a classification mechanism that support the representation of hierarchical relationships.

[5] Website: http://download.oracle.com/javase/1.5.0/docs/guide/swing
[6] Website: http://java.sun.com/j2ee/1.4/docs/index.html

3 Background

Subject-based classification refers to '*any form of content classification that groups objects by the subjects they are about*' [10]. Such a technique is based on selecting a set of keywords and organizing them according to a criterion that is significant for the users. This can take many forms, such as controlled vocabularies, taxonomies, thesaurus or ontologies. Compared to taxonomies, controlled vocabularies do not allow designers to establish relationships between terms; thesaurus makes it possible to include hierarchical and associative relationships. As far as ontologies are concerned, their purpose is to capture domain knowledge in a generic way and provide a commonly agreed understanding of a domain; however, its use is not intuitive for non-expert users. Consequently, we chose the taxonomy since it provides semantics enough to classify the components and they are relatively easy to use, as far as they are properly designed as pointed out below.

Taxonomies are defined as '*a hierarchical system of classification representing structural differences'* [11]. These artifacts are effective to help users to choose the right term between two closely related terms [17] as long as aspects as the naming of the concepts – *terminology*-, the source materials– *corpus*- and the structural model were carefully considered.

4 The Taxonomy Development Process

The taxonomy development process aims at developing a taxonomy with a set of dimensions each consisting of a set of characteristics that sufficiently describes the objects in a specific domain [19]. To accomplish this goal, such procedure should have the following qualities [19]:

– *To take into consideration alternative approaches to taxonomy development*. Our procedure is based on the model defined by Nickerson [19] for the development of taxonomies in information systems. This model follows Bailey´s deductive to empirical approach [2], which has been applied for the development of taxonomies in social sciences.

– *To reduce the possibility of including arbitrary or ad hoc dimensions and characteristics in the taxonomy*. The definition of our taxonomy, and as a consequence of its dimensions, will be based on existing standards, design pattern libraries, and empirical grounds such as UI component libraries.

– *To be completed in a reasonable period of time*. The development of our taxonomy will follow an iterative process controlled by web-developers and UI-design experts. In order to avoid an endless process, the taxonomy will be considered as correct when a majority of experts rate it as valid.

– *It must be straightforward to apply*. Since we are not interesting in drawing additional conceptualizations that might have not been presented in the empirical data, deductive level has not been included, easing the development process.

– *It must lead to a useful taxonomy*. The taxonomy has been conceived as a design artifact that may address the development process of RIAs. In this way, the taxonomy would be a useful mechanism for web-developers without experience creating this kind of web applications. Nevertheless, in this work we will not validate its usefulness but only its quality.

In keeping with these principles, we have defined a development process (see Fig.1) divided into two phases: the empirical phase, aimed at identifying general characteristics of the objects, and the operational phase, used to identify missing objects in order to create an object with the specific missing characteristics. As aforementioned, deductive level has not been included in our proposal. The following subsections explain these two phases.

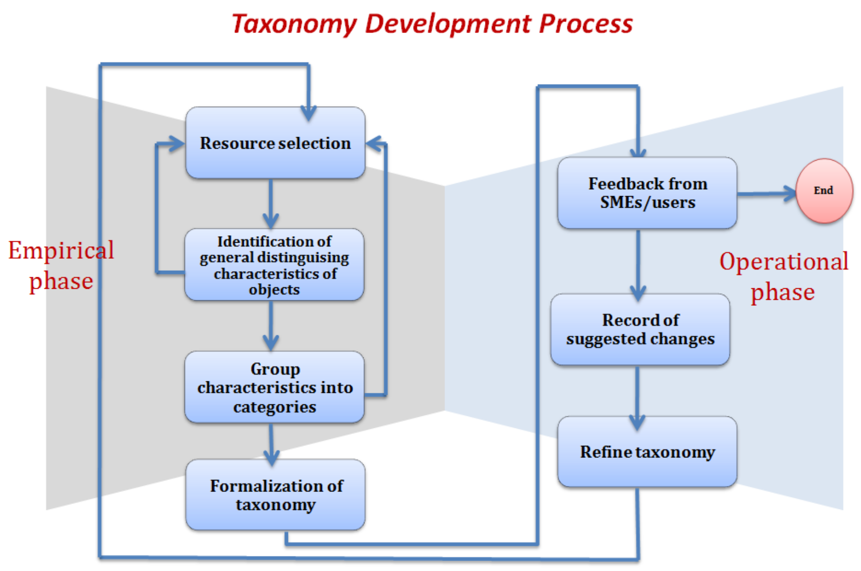
Taxonomy Development Process
Empirical phase
Resource selection
Identification of general distinguising characteristics of objects
Group characteristics into categories
Formalization of taxonomy
Feedback from SMEs/users
End
Operational phase
Record of suggested changes
Refine taxonomy

Fig. 1. The taxonomy development process.

4.1 Empirical Phase

The empirical phase aims at identifying general characteristics of UI components in order to define a preliminary version of the taxonomy. To achieve this goal, the most direct factors contributing to the quality of the taxonomy, such as corpus, the coverage of those source materials and its structural model, has been considered.

The quality of the corpus is expressed in terms of *multiplicity* referred to the number of resources involved; *relevance* defined as the significance of the source collection to a given matter; and *homogeneity*, which should yield high quality results within the area of coverage, with deep and consistent results [25]. In keeping with the multiplicity indicator, the number of reviewed resources amounts to twelve, quantity validated by SMEs in the operational phase. *Standards and guidelines*, *interaction design pattern libraries,* and *development technologies* have been selected due to their relevance and wide use for both, UI development and UI design. Furthermore, in order to determine the UI domain coverage for each of these resources a set of subjects, as shown in Table 2, has been collected.

Table 1. Main subjects identified for each resource reviewed.qu

Type of resource	Name of resource	Concepts for the taxonomy
Standards and guidelines	Document Object Model (DOM)[7]	Hierarchy of basic objects in documents divided in: *navigation*, *structure* and *control* elements.
	WAI-ARIA[8]	
Interaction design pattern libraries	Welie Design Pattern Library[9]	Knowledge of how a design should work, under preferred conditions (but may suggest how to cope with tradeoffs)
	Yahoo! Design Pattern Library[10]	
	Van Duyne Design Pattern Library	
Web UI technologies	Java Server Faces	Existence of basic controls and introduction of components that provide capabilities such as direct manipulation, rich graphics and streaming media scenarios.
	Javascript	
	Adobe Flex	
Desktop UI technologies	WindowsClient.NET	Existence of basic controls that are used to get input from the user, interactive displays of highly formatted information that can be modified by the user and containers with special and general purpose in the UI.
	JavaFX	

This first effort at the taxonomy development process leads to group common characteristics of the components into dimensions that form the initial taxonomy. Because of the disparity of characteristics possessed by complex RUI components –

[7] Website: http://www.w3.org/DOM/
[8] Website: http://www.w3.org/TR/wai-aria/
[9] Website: http://www.welie.com/index.php
[10] Website: http://api.dojotoolkit.org/jsdoc/1.2/dijit

components that may be composed by other RUI components and provide more advanced functionalities, such as direct manipulation-, their categorization was the main challenge during the definition of this initial version of the taxonomy. Finally, these complex RUI components were mainly classified under the *Templates* category (see Table 2).

Table 2. Preliminar version of the taxonomy. Templates category.

- Entry	- Navigation
- Editor in place	- Breadcrumbs
- Autocompleter	- Paginator
- Rating	- Display
- Selection	- Local zooming
- Drag & drop manager	- Overview plus detail
- Parts selector	
	- Carousel

This level ends with the definition of a draft taxonomy composed by four main categories – *Controls*, *Widgets*, *Containers,* and *Templates*- and a total of 81 subcategories. Due to binary taxonomies are not desirable [25], the maximum deepness of our taxonomy is three. With the same purpose, the width, which indicates the average number of children inside the taxonomy, is 3.12.

4.2 Operational phase

The purpose of the operational phase is twofold. On the one hand, it is conceived to identify missing concepts, misunderstandings, or ambiguous terminology in order to refine our preliminary taxonomy. On the other hand, it allows us to validate the taxonomy and finalize the development process according to the agreement of experts over the relevance of the source materials, the definition of non-ambiguous and meaningful terms, and the consistency of hierarchical relationships throughout the taxonomy. Both activities are based on the feedback provided by SMEs. In our case, SMEs should have a background in either UI design or web development with an experience longer than five years. Table 4 summarizes the experience of the group of experts who participated in our work.

Table 3. Percentages of SMEs‘ years experience.

Experience	UI design	Web development
From 5 to 7 years	66,6%	62,5%
From 7 to 10 years	33,3%	25%
More than 10 years	0%	12,5%

Regarding the evaluation technique, we used a mix-questionnaire, i.e. the questionnaire combines both closed-ended questions and open-ended questions. Closed-ended questions make it possible to validate some attributes of the taxonomy by using a five-level *summated rating scale* ranging from *strongly disagree* to *strongly agree*. Open questions are used to collect opinions and findings from SMEs. Since *summated rating scales* do not provide concrete values but categories, we used the *median* to identify the agreement of experts: the taxonomy will be considered as valid if the median of experts' opinion is equal or higher than four (agreement level). Finally, as far as conducting the evaluation is concerned, the taxonomy was made available to experts through a public website. Such a website provided information about our solution, a detailed description of the taxonomy, and access to the questionnaire.

The operational phase compiles two iterations or evaluations. Eleven experts carried out the first one. According to the judgment of experts, the median of agreement score for both meaningful and non-ambiguous terminology was set lower than four; in particular, this initial evaluation suggested that some categories of the taxonomy were ambiguous or misleading – for instance, *Templates* category (see Table 2) was considered more suitable to designate the general structure of the UI not specific components. Otherwise, the agreement of experts over the relevance of the corpus of the taxonomy, and the consistency of relationships throughout the taxonomy was validated. In addition to the grade of the taxonomy, SMEs provided a set of opinions of it gathered in Table 5. At this point, we made some changes and additions in our taxonomy, such as the replacement of *Templates* category for *Interaction Design Patterns* category. The final version of the taxonomy is showed on the Appendix.

Table 4. Findings from SMEs.

Number	Finding
1	The distinction between "navigation button" and "push button" is unclear.
2	Is "navigation button" referred to a hyperlink?
3	"Templates category" is more suitable to designate the general structure of web page not specific components.
4	I could not understand the distinction between "menu bar" and "tool bar".
5	I cannot see the difference between "navigation button" and "push button". Additionally, in "Containers category", "Layout" falls at the same level than "Grid category", which is misleading due to grid element is a specific type of layout. Finally, in "Widgets category" is not clear which "Color Picker category" means.
6	Just a detail. "Date picker" term would be more suitable to designate such kind of components than "Calendar" term because it allows selecting a date not shows information related to dates (which would be a calendar).

7	"Parts Selector category" was the only term that makes me doubt. I suppose that is a "selector" between some "options" but I am not sure.

According to the iterative character of the development process, a second review was carried out. This second evaluation aimed to review the changes incorporated at the first evaluation in order to validate the new version of the taxonomy. As regards the evaluators, we considered two options: evaluate the taxonomy by the same group of SMEs or use a different group of SMEs with similar characteristics than the first ones. The first option might produce a bias because of the memory of the previous evaluation; on the other hand, a different group of SMEs may increase the *estimation error* of the evaluation. Since the applied evaluation technique was aimed to collect opinions and findings from SMEs based on their experience and knowledge, the bias caused by a test-retest evaluation could consider not significant. Thus, the second evaluation was carried out by the same experts. Table 6 summarizes the results of both evaluations. Due to the median of all indicators has achieved the agreement level, the taxonomy has been considered as valid after the second round of evaluation.

Table 5. Summary of results (by using the median) of the operational phase

Evaluation	The relevance of the corpus	Non-ambiguous	Meaningful terminology	Hierarchical relationships
1st eval.	4	3	3	5
2nd eval.	4	4	4	5

5 Taxonomy for Rich-User-Interface Components

The final taxonomy consists of four main categories and a total of 89 subcategories. Since the definition of the preliminary taxonomy, its deepness has not change but its width has increased to 3.4. Its purpose is to classify RUI components in a hierarchical way in order to ease their search and choosing by web-developers. In keeping with this purpose, the definition of the taxonomy was based on intrinsic properties of the components and the common process of choosing components. On the one hand, we identified meta-characteristics that usually serve as basis for the search process: the *structure* – the provision of organization of both interface elements and contents and their relationships to each other- and the *behavior* – the interaction events available- of UI components. On the other hand, the systematic design of the UI [24] is supported by the order of the categories defined in this hierarchical-classification mechanism. Following, the main categories that make up the taxonomy are described (full taxonomy is presented in the Appendix):

– **Containers**: it includes the RUI components used to organize the information. These components add support for *modality* – this property enables the developer to scope, or limit, a dialog box's modality blocking-, *drag and drop* – including moving, copying, or linking selected objects by dragging them from one location and dropping them over another- and default *look and feel characteristics*. Every component must be part of a containment *hierarchy* that has a top-level container as its root [4]. In keeping with that, containers have been divided into two subcategories: *top-level containers* that can hold other UI components; and *intermediate-level containers* that can hold and be held by other UI components.

- **Controls**: it includes the RUI components that allow users to carry out its tasks on the interface. Such category is divided into four subcategories: *imperative controls* used to initiate a function; *selection controls* used to select options or data; *entry controls* used to enter data; and *display controls* used to display the visual representation of information.
- **Widgets**: it includes the RUI components that provide a specific solution to a common design problem. These components use static, *predefined set of look and feel characteristics* in order to ease its configuration. Such category is divided into four subcategories: *entry widgets*; *selection widgets*; *display widgets*; and *navigation widgets*.
- **Interaction Design Patterns**: it includes the RUI components that provide a global solution to a common design problem. These components are usually constructed from aggregates of other RUI components and support setting the *input elements* according to each context of use. The subcategories defined are: *entry patterns; selection patterns*; *display patterns;* and *navigation patterns*.

6 Conclusions and future works

The existence of classification schemes is essential to be efficient when using component libraries. Nevertheless, existing RUI component libraries do not provide homogeneous classification criteria, resulting in the need of high previous experience as RIA developer to search and choosing suitable components. With the purpose of overcoming these limitations, we have defined a taxonomy that allows us to categorize in a hierarchical way the structural and control elements of the WUIs.

The development process of the taxonomy has been divided into two phases: empirical and operational. As a result of the empirical phase, and based on the review of relevant resources in the UI domain – such as standards and guidelines, interaction design pattern libraries, and development technologies- a preliminary version of the taxonomy was defined. The use of this wide corpus of information allowed us to identify essential controls and containers commonly used in web applications, as well as complex components and templates specifically conceived to resolve concrete web-design problems. Afterwards, we carried out two evaluation iterations in order to collect the findings and opinions of SMEs about the taxonomy. This assessment demonstrates the validity of both the development process applied to define our taxonomy and the taxonomy itself.

Further work will be guided to refine our taxonomy and prove its utility. First of all, based on the opinions of SMEs, we will review other RIA technologies and standards. Secondly, we will carry out the empirical evaluation of both the quality of the taxonomy and its usefulness to select RUI components by means of the provision of a software tool. This tool will allow us to assess other desirables factors for taxonomies, including: *evolution*, which represents its stability over time; *display* that is the way the taxonomy is shown to the user; *feedback mechanism* that is the method used to communicate with the user; and *performance,* which represents the performance compared with other taxonomy solutions. Finally, based on specific web-based interactive design scenarios, users testing will be carried out in order to assess the usefulness of the taxonomy to select RUI components. This evaluation may be performed by different groups of web-developers with different background in RIA development. Since our final goal is to provide artifacts that systematize RIA development, the target audience of the taxonomy should be as broader as possible, bringing the gap between the level of expertise and the way of expressing their needs.

Acknowledgements

This work has been partly supported by the RIA – User Interface with Patterns and Components Reuse project funded by the Regional Government of Madrid (CAM) and Universidad Carlos III de Madrid and the urTHEY grant (TIN2009-09687) funded by the Spanish Ministry of Science and Innovation (MICINN).

References

1. Allaire, J.: Macromedia Flash MX - A next-generation rich client. Macromedia - White Paper. (2002) Retrieved 15/11/2010 from http://download.macromedia.com/pub/flash/whitepapers/richclient.pdf
2. Bailey, K.D.: A Three-Level Measurement Model. In Quality and Quantity. (18) 225-245 (1994)
3. Barros, J.L.: Técnicas para la clasificación/recuperación de componentes software reutilizables y su impacto en la calidad. In Sistemas de Informaçao. ISSN 0872-7031 (1998)
4. Brereton, P., Budgen, D.: Component-based systems: A classification of issues. IEEE Computer, vol.33.no 11 (2000)
5. Brown, A. W.: Component-Based Software Engineering: Selected Papers from the Software Engineering Institute, IEEE Computer Society Press, Los Alamos (1996)
6. Cooper, A., Reimann, R., Cronin, D.: About Face 3: The Essentials of Interaction Design. Wiley, 3rd edition (2007)
7. Curtis, N.: Modular Web Design: Creating Reusable Components for User Experience Design and Documentation. New Riders (2009)
8. Deitel, P.J, Deitel, Deitel, H.M.: Ajax, Rich Internet Applications and Web Development for Programmers. Deitel Developers Series (2008)
9. Frakes, W.B., Pole, T.P.: An empirical study of representation methods for reusable software components. IEEE Transactions on Software Engineering, 20(8), (1994)
10. Garshol, L. M.: Metadata? thesauri? taxonomies? Topic maps! Making sense of it all. Journal of Information Science, 20(4), pp. 378—391 (2004)
11. Guidelines for the Construction, Format, and Management of Monolingual Controlled Vocabularies Construction of Controlled Vocabularies, Based on ANSI/NISO Z39.19-2005 ISBN: 1-880124-65-3 Retrieved 16/11/2010 from http://www.slis.kent.edu/~mzeng/Z3919/index.htm
12. Jagerman, E.J.: Creating, maintaining and applying quality taxonomies. Published by Evert Jagerman (2006)

13. Java Look and Feel Design Guidelines. Sun Microsystems Inc. Version 2.0 (2001) Retrieved 15/11/2011 from http://java.sun.com/products/jlf/ed2/book/index.htm
14. Jeetendra, P., Bisht RK.., Pant, D., Pathak, V.K.: On Some Quality Issues of Component Selection in CBSD. Journal of Software Engineering and Applications. 3(6). (2010)
15. Kaindl, H., Jezek, R.: From Usage Scenarios to User Interface Elements in a Few Steps. In: Proceedings of the Fourth International Conference on Computer-Aided Design of User Interfaces (CADUI 2002), Valenciennes, France, May, pp. 91-102. Kluwer Academic Publishers, Dordrecht (2002)
16. Levin, M.S.: Hierarchical design of User Interfaces. Human-Computer Interaction .pp. 140-151. Springer Berlin / Heidelberg. (1994)
17. McKelvey, B.: Organizational Systematics: Taxonomy, Evolution, Classification. University of California Press. (1982)
18. Naumann, J.D., Jenkins, A.M.: Prototyping: The New Paradigm for Systems Development. MIS Quarterly, Vol.6, No.3 (1982)
19. Nickerson, R.C., Muntermann, J., Varshney, U.: Taxonomy Development in Information Systems: A Literature Survey and Problem Statement. In Proceedings of the 16th Americas Conference on Information Systems (AMCIS 2010)
20. Nickerson, R.C., Varshney, U., Muntermann, J., Isaac, H.: Taxonomy Development in Information Systems: Developing a Taxonomy of Mobile Applications. In 17th European Conference on Information Systems. (2009)
21. Pérez, S., Díaz, O., Meliá, S., Gómez, J.:Facing Interaction-Rich RIAs: The Orchestration Model. In Eighth International Conference on Web Engineering, pp.24-37. Yorktown Heights, New York (2008)
22. Pressman, R. S.: Software Engineering: A Practitioner´s Approach. McGraw-Hill. (2005)
23. Plomp, C.J., Mayora-Ibarra, O.: A Generic Widget Vocabulary for the Generation of Graphical and Speech-Driven User Interfaces. Springer Netherlands (2002)
24. Stone, D., Jarrett, C., Woodroffe, M., Minocha, S.:User Interface Design and Evaluation (Interactive Technologies). Morgan Kauffman Series (2005)
25. Vogel, C., Powers, J.: Quality Metrics: How to Ensure Quality Taxonomies. In Information Today, Medford (2000)

Appendix: Taxonomy for Rich-User-Interface Components

Containers

- **Top-level Containers**
  - Primary Window
    - Layout
  - Secondary Window
    - Modal dialog
    - Modeless dialog
- **Intermediate-level Containers**
  - Pane
  - Scroll pane
  - Tabbed pane
  - Split pane
  - Collapsible pane
  - Movable pane
  - Two-panes selector

Widgets

- **Entry Widgets**
  - Text editor
  - Form
  - File uploader
- **Output Widgets**
  - File downloader
  - Data exporter
- **Selection Widgets**
  - Date picker
  - Color picker
  - Dock menu
  - Schedule
  - Spreadsheet
- **Navigation Widgets**
  - Tag cloud
- **Display Widgets**
  - Chart
  - Graph
  - Map
  - Dashboard
  - Treemap
  - Sparklines
  - Media
    - Video
    - Image

Controls

- **Imperative Controls**
  - Push button
  - Hyperlink
- **Selection Controls**
  - Checkbox
  - Radio button
  - Toggle button
  - Listbox
  - Combobox
  - Tree
  - Pick List
  - Menu
    - Context menu
    - Drop-down menu
    - Menu bar
    - Tool bar
- **Entry Controls**
  - Bounded-entry controls
    - Slider
    - Spinner
  - Text entry controls
    - Text area
    - Text field
    - Password field
- **Display Controls**
  - Progressbar
  - List
  - Label
  - Tooltip
  - Table

Interaction Design Patterns

- **Entry Patterns**
  - Editor in place
  - Autocompleter
  - Rating
- **Selection Patterns**
  - Drag & drop manager
  - Parts selector
- **Navigation Patterns**
  - Breadcrumbs
  - Paginator
- **Display Patterns**
  - Local zooming
  - Overview plus detail
  - Carousel
  - Image Comparer
  - Image Cropper