Model Driven Engineering of Rich Internet Applications Equipped with Zoomable User Interfaces

Francisco J. Martínez-Ruiz[1, 2], *Jean Vanderdonckt*[1] *Juan Manuel González-Calleros*[1], *and Jaime Muñoz Arteaga*[3]
[1]Université catholique de Louvain, Louvain School of Management, Information Systems Unit
Belgian Lab. of Computer-Human Interaction -
Place des Doyens, 1 – B-1348 Louvain-la-Neuve, Belgium
{francisco.martinez, jean.vanderdonckt}@uclouvain.be
[2]Universidad Autónoma de Zacatecas, México
[3]Universidad Autónoma de Aguascalientes, México – jmunozar@correo.uaa.mx

***Abstract*—The development of a Rich Internet Application is particularly challenging because its user interface can involve highly interactive techniques that require substantive programming that is mostly done by hand nowadays. This paper addresses this challenge by introducing a model-driven engineering approach where a zoomable user interface of such a Rich Internet Application is obtained successively by performing the following steps: the Computing Independent Model level consists of modeling a task for the future interface based on a list of canonical task types augmented by custom tasks, each task being mapped onto a domain model; the Platform Independent Model level consists of exploiting the structure and the temporal operators of this task model in order to generate one or many abstract user interfaces that will lead in turn to concrete user interfaces structured according to the principles of a zoomable user interface at the Platform Specific Model level.**

***Keywords:* Context of use, Rich Internet Application, Ubiquitous computing, User interfaces, User Interface Description Language, Vectorial User Interface, Zoomable User Interface.**

I. Introduction

Rich Internet Applications (RIAs) offer similar features and capabilities to the ones available in desktop applications, e.g., robustness, dynamic adaptation depending on users' profiles and multimedia. Also, their development introduces a series of problems. For instance, temporal constraints in terms of loading presentation elements or application logic (encapsulated in an ECMAScript dialect mostly Javascript). Besides that, most of the UI rendering is transferred to client-side [17]. In [5] a Model Driven approach is proposed. This method implies an iterative process of decomposition over a hierarchy of tasks. Here, the problem that emerges in non-trivial developments is the generation of enormous structures where developers are disoriented by the visual ambiguity of the repeating structures which observation could not lead us to the recovery of patterns nor semantic inferences. In the visualization of large information spaces, such as the Task Tree Diagrams (TTDs), it would be useful for a developer being able to scale and zooming in to obtain further details. Moreover, a developer could be interested about similarities in a section previously seen in another zone. This exploratory analysis could aid in an informal visual review of TTDs in order to recover hidden patterns or do comparisons that could be difficult to express in textual format [3]. Therefore, the synergy of mixing RIA development with ZUIs could boost the early step of task definition. The rest of this paper is organized as follows: Section 2 discuss the state of the art in the creation of ZUIs. Section 3 introduces some theory in Task models and model driven engineering domains. Then Section 4 covers the description of our method over a case study (see Section 5). And finally, Section 6 presents conclusions and future work.

II. Related work and problem description

This section includes three subsections: first, a review of the state of the art in the domain of ZUIs. Second, one dedicated to match RIAs features to ZUI ones. And third, an analysis of the problem.

A. Review of ZUIs

Zooming is utilized by users in multiple activities, for instance in the process of reading a newspaper since this activity includes a zooming out task (when titles are browsed) meanwhile a zooming in task is required in order to focus in a specific article or in order to review in detail schematic diagrams [10]. The benefits of ZUIs have been researched in [7]. There, the amount of incorrect selections was a measure in order to compare this approach to window based ones. In [14] the purpose of the study was to discover in which conditions zooming is more effective. According to [10] one of the advantages of zooming is the sense of location (geographical location) and object constancy.

The use of 2D structures with the goal of displaying hierarchies of information could be tracked to [9].There, the goal was to display in box-like elements, large directory structures of hard disks. Hierarchical information structures contain two kinds of information: first, information about the internal organization of each node and second, the content itself. ZUIs are a very well established study subject and they are applicable over large information sets (images or 3D scenarios) [3], vast collections of categories, for instance, music titles [1]. In [12], the idea is to combine fisheye views with compact overviews in a calendar application for mobile devices. Another interesting example is an interactive time-line component which includes vertically stacked time levels and zooming capabilities [13]. Finally, ZUIs are developed manually with the help of APIs such as JAZZ [6] but the task remains a complex one since the development does not follow a model based approach.

B. *RIA features related to ZUIs*

RIAs are complex at the development level for multiple reasons: (1) large number of objects has to be included in order to model the UI presentation and behavior. Also, (2) the management of non-traditional interactive widgets, (3) the navigation should be fluid and continuously animated and finally, (4) the TTDs should include code or weight schemes in order to support the work of developers. Note: many non-RIA applications have similar requirements and the use of TTDs is not exclusive of RIAs [5], [17].

C. *Problem description*

The explosive grow of Task tree nodes was the earliest motivation for this research. Since the initial step in the method [5] for developing RIAs includes this notation (see Fig. 1). The goal of this research is delivering: a solution with (1) Efficient Space Utilization, (2) Interactivity, i.e., the structure should be updatable and navigable in real time and (3) A reduced cognitive and perceptual load for the user (see Fig. 2). The construction of task models is not a trivial task since it is a time consuming process. The specification in [5] is based on ConcurTaskTree notation [15]. This approach has multiples benefits nevertheless there are a series of limitations: The first limitation is the inevitable growing size of the TTD for average projects. In current versions of the graphic tool for ConcurTaskTree the problem is treated with the inclusion of zooming in some of the TTD branches and techniques such as word wrapping and fisheye [23]. But the problematic remains because icons and text under certain size are unreadable.

D. *Monotonic Diagram representation*

On the other hand, current TTDs display a monotonic nature (see Fig. 1). Trees are structured by hierarchies of nodes where the only semantic information to be collected is relationship between children and parent nodes (i.e., deepness). Also, the repetitive structure of TTDs is not semantic-aware. That is, over a certain level of zooming the TTD representation does not rapport any relevant information. Finally, developers are not provided with a mechanism for visualize whole TTD in a useful way. Those inconveniences presented previously constitute the reasons to propose an alternative TTD representation (see Fig. 2).

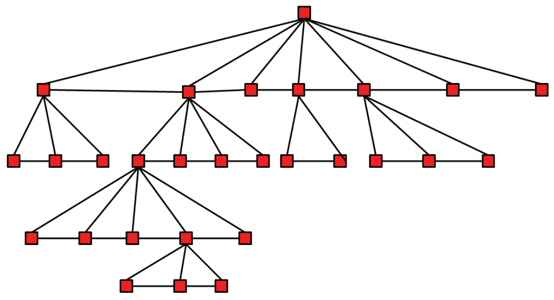


Figure 1. Example of a TTF with four levels.

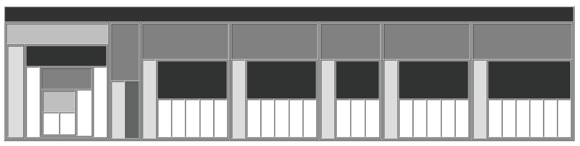


Figure 2. Example in ZUIT format.

III. Model driven engineering approach

Our methodology rests in a Model driven engineering approach [16]. The main elements are the following: the CAMELEON framework [5], UsiXML [4,24,25] and a variation of the CTT task model [15].

A. *CAMELEON Framework*

The design of UIs using a model based approach that includes characteristics such as Multi-level abstraction and Modality independence [16] suppose the use of a framework to deal with the complexity of the process. Our choice is the CAMELEON framework [21]. This framework breaks down the development process in four levels of abstraction: Task and concepts (T&D), Abstract User Interface (AUI), Concrete User Interface (CUI) and Final User Interface (FUI). The UI is represented in the User Interface Description Language (UIDL) UsiXML (User Interface eXtensible Markup Language). This UIDL covers all framework levels, in a design independent way and over multiple contexts (including vocal and graphical UIs).

B. CTT-based task models

The Concur Task Tree model (CTT) is a model proposed for modeling task trees [15]. The task model of UsiXML is implemented through CTTs. The objective of these models is representing the tasks that users are required to do in order to fulfill a goal. The process is iterative and each task is decomposed in sub tasks. Indeed, the final result is a hierarchy of tasks. The tasks are related through temporal operators which are described below. **Concurrent Operators**: These operators imply that Taks performed in any order, in a concurrent order: |=|, ||| and |[]|. **Sequential operators**: [>, |>, >> and []>> these operators imply a strict sequence in the order of execution of the tasks. **Selection operator**: [] exclusive choice between Tasks.

IV. Method outline

The following section describes the proposed method in order to generate Zoomable UIs for RIAs (ZUITs) at the Task Model and also at FUI level.

A. The general Method

The proposed method is a refinement of the first level of the development cycle proposed in [5] in which is a process of refactoring Computing Independent Models (CIM) as defined by OMG [16] to the concrete models: Platform Specific models. The process could be summarized in the following sections: In the first step (the focus of this paper) we create the **Task and Domain models** (T&D). The second step transforms the previous models into a UI definition independent from any modality, an **Abstract User Interface** (AUI). In the third step, the AUI is transformed again towards a **Concrete User Interface** (CUI). Here, the selection of modality and widgets (for a specific toolkit) is done. The final step is the delivery of **Final User Interfaces** (FUI) which are expressed in a specific platform (for instance, .NET, LZX, SWF among others).

B. The Task model updated

The focus of this paper is delivering an alternative TTD representation. In order to expand current TTD diagrams and solving the problematic presented in section 2.3. The general process to update this model is presented in Fig. 3.

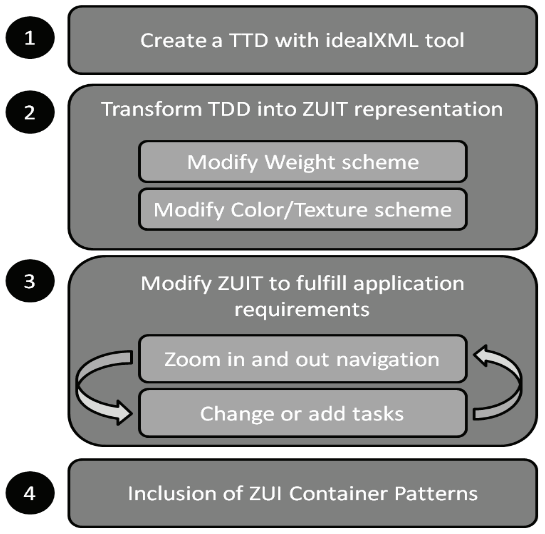

1
Create a TTD with idealXML tool
2
Transform TDD into ZUIT representation
Modify Weight scheme
Modify Color/Texture scheme
3
Modify ZUIT to fulfill application requirements
Zoom in and out navigation
Change or add tasks
4
Inclusion of ZUI Container Patterns

Figure 3. The proposed update to TTDs.

1) Create a TTD representation: This step requires the use of IDEALXML [20] to produce the UI model of this level. The process of creation is out of the scope of this paper and could be revised in [5].

2) Transform TTD into a ZUIT: The next section describes the proposed method for producing the alternative TTD structure. The key difference between this approach and the TTD is a better handling of the space. Because, even with big models, the area exposed to developers remains the same. However, the segmentation of a space, as any procedure of division over an area implies a bin-packing like problem (therefore, it is a NP-complete problem as well). For this first attempt, we are using a breadth-first traversal of the TTD in which our algorithm executes top-down (see Fig. 5) or in an informal way (see Fig. 4). The process of transformation is done by the aid of XSL transformations in order to translate the UsiXML definitions into nested box structures. XSLT has been estimated economic enough in order to produce the required transformations without relying on a transformation engine and transformation rules. On the other hand, XSLT is straightforward.

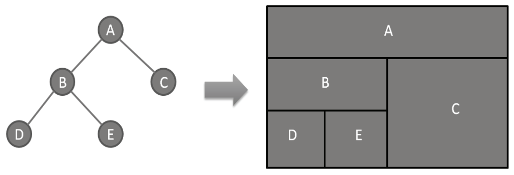


Figure 4. Transformation of TTD in ZUIT format.

```
Function CreateBNF structure(TaskTree tasks)
        returns ZuiStructure or failure
For each parent task GeneratBoxComponent()
    For each leaf task
        GeneratCellComponent()
        and
        DefineContainmentRelation(leaf, parent)
```

Figure 5. Algorithm for generating the BNF structure.

3) *Modify Weight and Color schemes*: This section deals with the definition of the color and weight schemes. The power of this alternative representation relays on these elements. Color schemes allow us to reduce the visual overload of managing tens of labels (or even more for standard applications). In a previous section (problem description) we already discussed about the intention of reducing the cognitive load and clarify the model. Therefore, instead of introducing textual definitions we select the use of a coding color scheme in order to identify the nature of temporal operators in each level of the TTD and also a color scheme is proposed to the nature of the tasks (see tables 1 and 2). For the sake of clearness, instead of gray scales, we use here texture schemes in tables and diagrams. The result of the application of these schemes could be seen in Fig. 6.

TABLE I. CODING SCHEME FOR HEADERS

color	**Type of Operator**	**Operators**
☰	Sequential	[>, |>, >>, []>>
‖‖‖	Concurrent	|=|, ||| , |[]|
▨	Choice	[]

TABLE II. CODING SCHEME FOR CELLS

Color	**Type of task**
▦	Interactive
▩	System
▧	Abstract

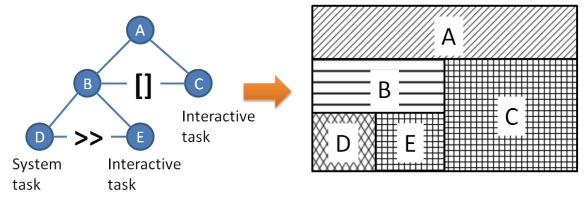


Figure 6. Application of color coding scheme.

On the other hand we have profit of the visual specification in order to introduce another dimension: the complexity of the task expressed as a weight. The decomposition of tasks into subtasks allows the detection and integration of visual help in terms of the complexity of tasks. That is, a complex task is represented with a relative bigger area than their neighbors. Note: These weights were proposed in [17] and there are shown here as an example (see table 3).

TABLE III. WEIGHT OF TASK TREE ELEMENTS

Weight	**Items to process**	
	Task Type	Operator Type
8	-	Concurrent
4	Interactive	Choice
2	Application	Sequence

The procedure defined in [17] could be used in order to calculate the different weights. But no explicit method is proposed here. Also, we have added features included by the task level definition of UsiXML (for instance: importance and complexity levels among others). With all this, it is possible to propose a weight measure in order to clarify the complexity of the task. For example, an interaction task which includes multiple validation and recovery of information processes is represented with a wider area (see in Fig. 7 where task E is depicted as more complex than task D).

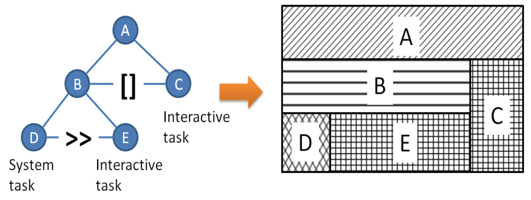
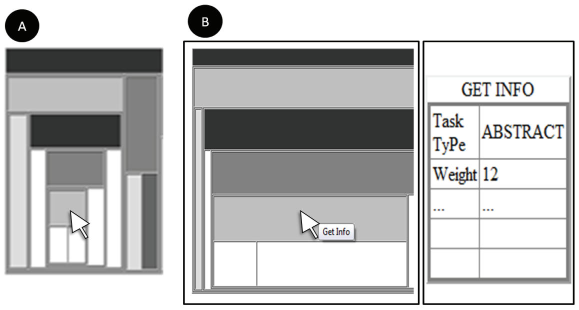


Figure 7. Application of weight coding scheme.

4) *Modifying the ZUIT:* The next step is modifying the Task model. This revision process is as follows: the developer navigates through the ZUIT (Zooming in and out, Fig. 8a and 8b) in order to update the structure of the application and also modifying task weights in order to change their importance (see Fig. 8b). Also is possible to make quick comparison between tasks in terms of position (also deepness) and weight. By doing this is possible to understand which areas are overloaded and could be divided, as well as, some tasks could be migrated to other sections. This updating process now is not automatic and requires the full regeneration of the ZUIT. Finally, Create, Read, Update and Delete operations (CRUD) are done over the tree tasks.

5) *Exploring ZUI patterns*: The final step is the introduction and application of Zoomable UI patterns (see Figs. 9 and 10). This classification could help us to deliver different FUI types for a RIA application.



GET INFO	
Task TyPe	ABSTRACT
Weight	12
...	...

Figure 8. Zooming into a specific task.

The patterns could be aligned according to a 2D coordinate system and then, a Zoomable UI over the horizontal axis could be represented with the Fig. 9a, a vertical one with 8.b and a mixed approach one with 9c. Also, the utilization of the available area could be expressed in alternative ways. For instance, there are three possible patterns: **expanded**, **closed**, and **preview**. This last one could be divided in two: **explained** and **outlined** (see Fig. 10). The last ones imply the inclusion of guidance elements for the users. For instance, a sort of content introduction or summary could be added. The behavior model for this type of ZUIs is presented in Fig. 11. The icon state is not always present. The preview alternative states are not a common practice now-a-days. Nevertheless, the provided model could handle all the possible states of this kind of UIs.

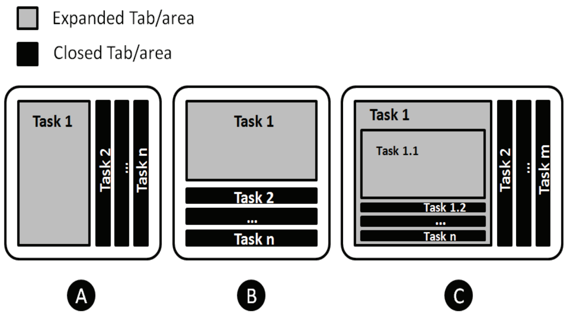


Figure 9. ZUI patterns.

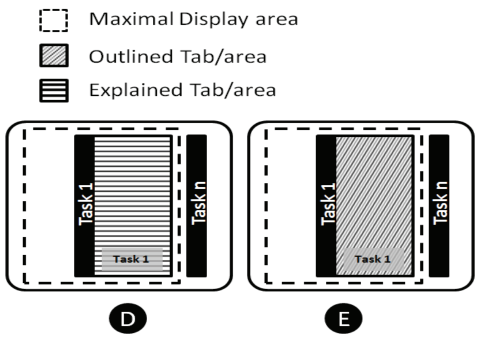


Figure 10. ZUI patterns (continued).

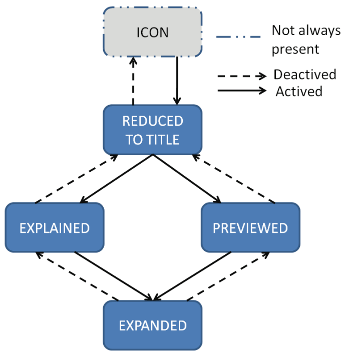


Figure 11. Behavior model of this type of ZUIs.

In order to arrive to a Zoomable FUI of the RIA UI we could use the Zoomable patterns depicted in Figs. 9 and 10 with a variation of the algorithm for the creation of menus proposed in [21]. In this case, the formal algorithm is presented in Fig. 11 where the general procedure is traversing the ZUIT (with a breadth-first based algorithm) in order to identify which set of tasks are related by chains of choice operators. Note: leaf nodes are excluded because they are related to selection operations of low level tasks, e.g., choose color, size or price range. Instead of selection of high level task that are related to the user's goal, e.g., pay for a product, request supplies, among others. In the case of a mixed pattern, the general algorithm contemplates a simple approach: including up to a predefined number of sub-items (Until the capacity is reached) in order to fulfill a specific level. Then, the rest of the sub-items are sent to other container.

```
function GenerateZoomUI (CTT tree, TypePattern Pattern)
returns Zoomable-structure or failure
initialize the search tree to root node
loop do
if there are no candidate nodes for expansion then return exit
   choose a node and expand its sons
   if the sons of candidate node include only choice operators then
          include it in ZoomList[ ] and its sons as zoom-items.
          Case Pattern is
              OUTLINED then
                 include candidate node name to zoomTitle
              EXPLAINED then
                 include candidate node name to zoomTitle
                include candidate node description to
                         zoomDescription
              MIXED then
                For each subset count of sons
                     Where mod capacity of sub-items equals 0
                     then
                        Generate new subzoom
                        ZoomList.orientation = Mixed
        if candidate node previously marked as zoom-item then
              change the label to subzoom
              associate to upper zoomable element
```

Figure 12. Algorithm for generating ZUIs.

V. Case Study

The chosen case study is a commercial web site for espresso coffee [18] where users could select and buy new machines and supplies (see Fig. 13). A possible TTD is presented in Fig. 15. Then, the algorithm described in Fig. 4 is applied and the final result of this transformation and some details (for instance, the location of different tasks) is presented in Fig. 14. Also, it is interesting to notice here the emergency of patterns in (some of the) subtasks (see Fig. 14a). This set of tasks has a similar structure which is going to be exploited in the next section.

A. Inclusion of Container Patterns

In Fig. 16a the first level is constituted by a series of choice operators. This set is transformed by the algorithm (Fig. 12) into a ZUI, if the orientation parameter is set to horizontal, vertical and mixed then the delivered FUI is depicted in Figs. 15a, 15b and 15c, respectively. Note. For the mixed approach we also take the next level in the TTD (a sub zoom) which corresponds to the sub tree depicted in Fig. 16b.

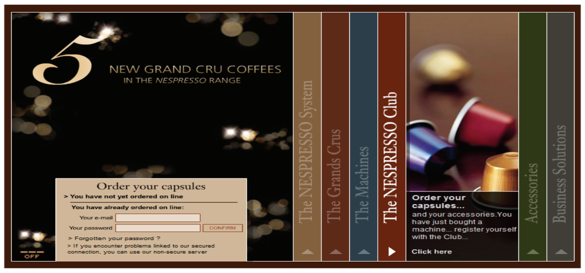

5
NEW GRAND CRU COFFEES
IN THE NESPRESSO RANGE
Order your capsules
> You have not yet ordered on line
You have already ordered on line:
Your e-mail
Your password
CONFIRM
> Forgotten your password ?
> If you encounter problems linked to our secured connection, you can use our non-secure server
The NESPRESSO System
The Grands Crus
The Machines
The NESPRESSO Club
Order your capsules...
and your accessories.You have just bought a machine... register yourself with the Club...
Click here
Accessories
Business Solutions

Figure 13. Coffee Shop example (fragment).

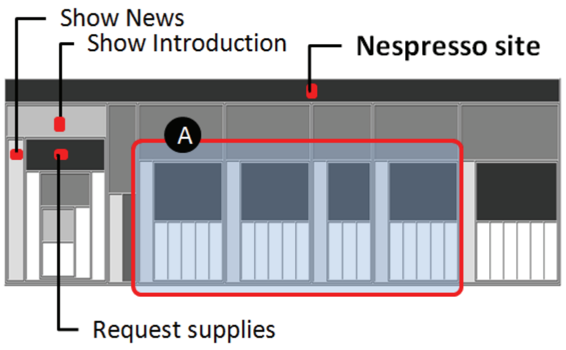


Figure 14. ZUIT of the Coffee Shop example.

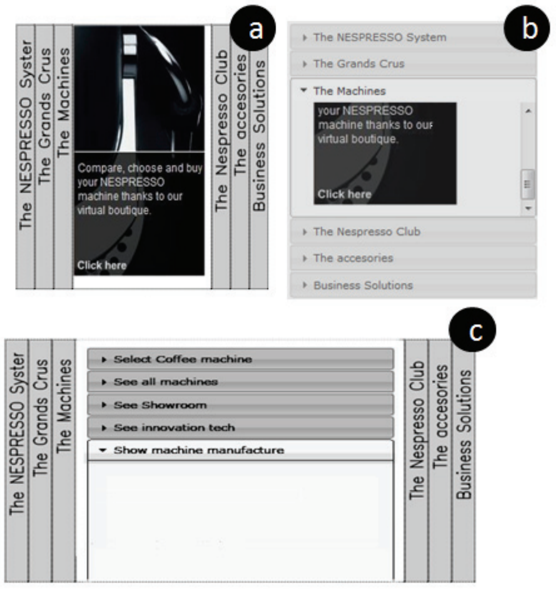


Figure 15. Possible Final User Interfaces.

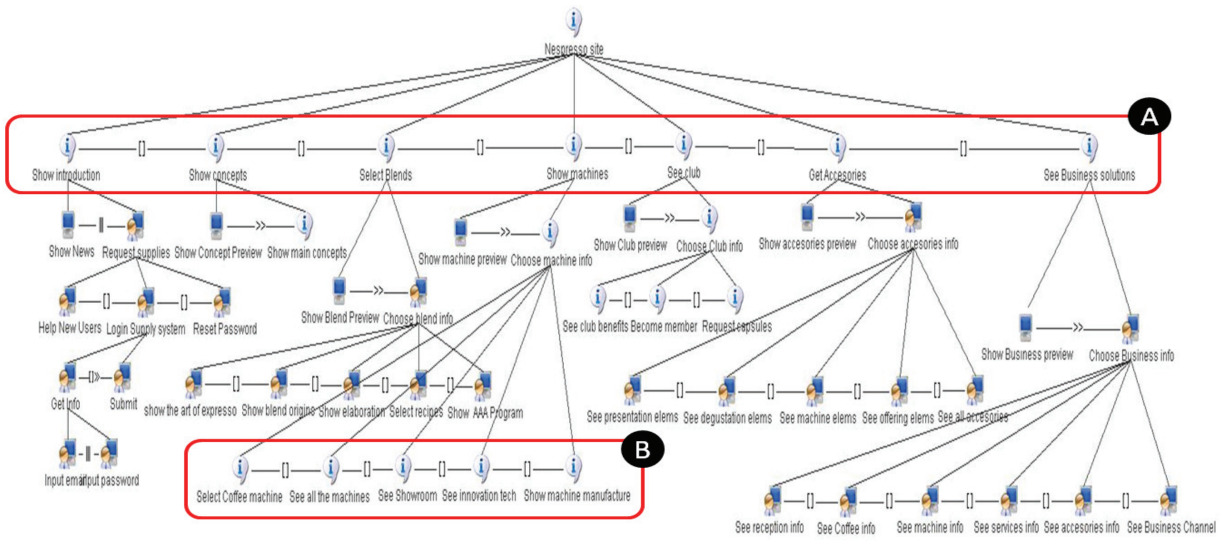


Figure 16. Partial TTD of case study.

VI. Conclusions and Directions for Future Research

In this paper we introduced ZUIT, a novel approach to support model driven development of RIAs. The Task Tree Modeling step is treated with a Zoomable User Interface based widget in order to help developers to browse and refine the different tasks that constitute their webapps. Various levels of the task hierarchy are permanently available and can be reached promptly. The navigation is one of the strong attributes of this approach, we choose Piccolo [19] as the development platform. Because this API provides pan-and-zoom navigation and ease interaction with the different task hierarchies. Finally, a general algorithm for delivering Zoomable UIs is proposed.

It is assumed a better performance of this approach in comparison to scrolling or simple decorative zooming ones but further user studies are needed. Also, we are working in the creation of a prototype in order to test this approach with running projects. Current examples are using [22] as primary development software (see Fig. 15). Also, we are preparing a full definition of the equation for calculate the weight. Finally, the process of updating is manual (see section 4.2.4) and we are working in automatic version

Acknowledgment

This work is supported by the Programme AlBan, the European Union Programme of High Level Scholarships for Latin America, scholarship No. (E06D101371MX) and the Belgian Computer Human Interaction Lab. We gratefully acknowledge the support of the ITEA2 Call 3 UsiXML Project under Contract #2008026

References

[1] Dachselt, R. and Frisch, M. 2007. Mambo: a facet-based zoomable music browser. In Proceedings of the 6th international Conference on Mobile and Ubiquitous Multimedia (Oulu, Dec. 12-14, 2007). MUM '07, vol. 284. ACM, New York, NY, 110-117.

[2] Dachselt, R., Frisch, M., and Weiland, M. 2008. FacetZoom: a continuous multi-scale widget for navigating hierarchical metadata. In Proceeding of the Twenty-Sixth Annual SIGCHI Conference on Human Factors in Computing Systems (Florence, Italy, April 05 - 10, 2008). CHI '08. ACM, New York, NY, 1353-1356.

[3] Plumlee, M. D. and Ware, C. 2006. Zooming versus multiple window interfaces: Cognitive costs of visual comparisons. ACM Trans. Comput.-Hum. Interact. 13, 2 (Jun. 2006), 179-209.

[4] UsiXML. http://www.usixml.org/ (January 15th, 2007)

[5] Martínez-Ruiz, F.J., Muñoz Arteaga, J., Vanderdonckt, J., González-Calleros, J.M. (2006), A first draft of a Model-driven Method for Designing Graphical User Interfaces of Rich Internet Applications, Proc. of 4th Latin American Web Congress LA-Web'2006 (Puebla, October 25-27, 2006), IEEE Computer Society Press, 2006.

[6] Bederson, B. B., Meyer, J., and Good, L. 2000. Jazz: an extensible zoomable user interface graphics toolkit in Java. In Proceedings of the 13th Annual ACM Symposium on User interface Software and Technology (San Diego, Cal., United States, Nov. 06 - 08, 2000). UIST '00. ACM, New York.

[7] Combs, T. T. and Bederson, B. B. 1999. Does zooming improve image browsing?. In Proceedings of the Fourth ACM Conference on Digital Libraries (Berkeley, California, United States, August 11 - 14, 1999). DL '99. ACM, New York, NY, 130-137.

[8] Furnas, G. W. and Zhang, X. 1998. MuSE: a multiscale editor. In Proc. of the 11th Annual ACM Symposium on User interface Software and Technology (San Francisco,, November 01 - 04, 1998). UIST '98. ACM, New York, 107-116.

[9] Johnson, B. and Shneiderman, B. 1991. Tree-Maps: a space-filling approach to the visualization of hierarchical information structures. In Proceedings of the 2nd Conference on Visualization '91 (San Diego, California, October 22 - 25, 1991). G. M. Nielson and L. Rosenblum, Eds. IEEE Visualization. IEEE Computer Society Press, Los Alamitos, CA, 284-291.

[10] Perlin, K. and Meyer, J. 1999. Nested user interface components. In Proceedings of the 12th Annual ACM Symposium on User interface Software and Technology (Asheville, North Carolina, United States, November 07 - 10, 1999). UIST '99. ACM, New York, NY, 11-18.

[11] Appert, C. and Fekete, J. OrthoZoom scroller: 1D multi-scale navigation. In Proc. CHI 2006, ACM Press, 21-30.

[12] Bederson, B.B., Clamage, A., Czerwinski, M.P., and Robertson, G.G. DateLens: A fisheye calendar interface for PDAs. Transactions on Computer-Human Interaction 11, 1 (2004), 90-119.

[13] Dachselt, R. and Weiland, M. TimeZoom: a flexible detail and context timeline. In CHI 2006 Extended Abstracts, ACM Press (2006), 682-687.

[14] Bederson, B., Boltman, A., "Does Animation Help Users Build Mental Maps of Spatial Information", submitted to CHI '99.

[15] Paternò, Fabio. Towards a UML for Interactive Systems Engineering for Human-Computer Interaction: 8th IFIP International Conf., EHCI 2001, Canada.

[16] OMG, http://www.omg.org , (May 10th, 2009)

[17] Francisco J. Martinez-Ruiz, Jean Vanderdonckt, Jaime Muñoz Arteaga Context-aware Generation of User Interface Containers for a Mobil Device, Mexican International Conferences on Computer Science, IEEE track in Human-Computer Interaction, Mexicali, Mexico, pp 63-72, October 2008.

[18] http://www.nespresso.com (May 10th, 2009).

[19] http://www.piccolo2d.org (May 10th, 2009).

[20] Montero, F., López-Jaquero, V., Lozano, M., González, P., IdealXML: un entorno para la gestión de experiencia relacionada con el desarrollo hipermedial, in ADACO: Ingeniería de la usabilidad en nuevos paradigmas aplicados a entornos web colaborativos y adaptativos, Proyecto Cicyt TEN2004-08000-C03-03, Taller celebrado en Granada, September 2005.

[21] Martinez-Ruiz, F., A Development Method for User

Interfaces of Rich Internet Applications, DEA thesis, UCL, Louvain-la-Neuve, 31 August 2007.

[22] http://jquery.com (May 10th, 2009).

[23] Paternò, F. and Zini, E. 2004. Applying information visualization techniques to visual representations of task models. In Proc. of the 3rd Annual Conference on Task Models and Diagrams (Prague,Nov. 15 - 16, 2004). TAMODIA '04, vol. 86. ACM, New York.

[24] Vanderdonckt, J., A MDA-Compliant Environment for Developing User Interfaces of Information Systems, Proc. of 17th Conf. on Advanced Information Systems Engineering CAiSE'05 (Porto, 13-17 June 2005), Lecture Notes in Computer Science, Vol. 3520, Springer-Verlag, Berlin, 2005, pp. 16–31.

[25] Stanciulescu, A., Limbourg, Q., Vanderdonckt, J., Michotte, B., Montero, F., A Transformational Approach for Multimodal Web User Interfaces based on UsiXML, Proc. of 7th Int. Conf. on Multi-modal Interfaces ICMI'2005 (Trento, 4-6 October 2005), ACM Press, New York, 2005, pp. 259-266.