Abstract. Fast production of a solution is a necessity in the world of competitive IT consulting business today. In engagements where early user interface design mock-ups are needed to visualize proposed business processes, the need to quickly create UI becomes prominent very early in the process. Our work aims to speed up the UI design process, enabling rapid creation of low-fidelity UI design with traditional user-centered design thinking but different tooling concepts. This paper explains the approach and the rationale behind our model and tools. One key focal point is in leveraging business process models as a starting point of the UI design process. The other focal point is on using a model-driven approach with designer-centered tools to eliminate some design overheads, to help manage a large design space, and to cope with changes in requirements. We used examples from a real business engagement to derive and strengthen this work.

Keywords: User-centered Design Process, UI Design, Model-driven User Interface, Low-fidelity UI Tools

1 Problem Statements and Context

In the world where competition for business is severe, we are in need for better tooling to support the solution design process. One such design activity is the user-centered design process. Whilst user observations and interviews are essential part of the process, these activities are undoubtedly irreplaceable by IT tools. However, some other parts of the user-centered design process – in particular coming up with low-fidelity UI mock-ups in the early design phase – is still lacking by way of appropriate tools. Many popular tools designers use today such as Adobe Illustrator, Adobe Photoshop, or Microsoft PowerPoint, were developed for other purposes such as drawings and presentations. They support high-fidelity drawings in nature and require high-fidelity efforts to produce low-fidelity mock-ups. This class of tools also lacks support for information structure and design management, hence dealing with requirement changes throughout a project is rather painstaking. Another class of
tools such as Macromedia Dreamweaver popular for creating web-based interfaces has more support on data connections and back end services, but the tooling itself is yet another high-fidelity tool. Some research tools were explored such as SILK [1] and DENIM [2] for capturing hand sketches low-fidelity UI and navigation structure. However, the outputs of these tools are not sufficiently organized visually, hence are not practical for business practices. None of these tools mentioned allow a quick way to come up with low-fidelity mock-ups with support for proper alignment and visual organization.

Many people often associate low-fidelity mock-ups with low-cost, hand-drawn interfaces. Our preliminary interviews with several user-centered designers revealed that hand-drawn sketches were not used in their UI mock-ups. They may start out with hand sketching but that typically lasted shortly. Some mentioned that their hand-drawn sketches did not look organized enough. But most agreed that it would take an even larger effort to use hand-drawn sketches including re-drawing repeatedly and scanning paper sketches. Needless to say that maintaining hand-drawn sketch changes even in the early design phase for a large project is far from viable. There’s a level of visual organization and alignments, especially in information intensive applications, which is implicitly required in presenting UI design ideas in the early design phase. Without it, the review process will suffer. All of these designers we interviewed commonly use high-fidelity tools mentioned above to create mock-ups that look low in fidelity.

In IT consulting practices, many business engagements nowadays often start with business process modeling – be it for an entirely new business process or transforming an old process to a new and improved one. Within our company, while Microsoft Visio is still the most popular tool for business process modeling, many consultants now have switched to use an IBM software product called Websphere Business Modeler (WBM.) In this kind of engagements, business analysts use WBM to draw business process diagrams, which are then used as a way to propose and have the customers approve the work. Prompt demand for low-fidelity UI mock-ups is three-fold in such engagements. First is an earlier need even before the UCD process begins. Low-fidelity UI designs and storyboards are commonly used as a means to visualize how a business process might work. Visualization of connected boxes as often diagrammed in a business model has limitations; they do not give people a good grasp of a more tangible understanding of the process. Secondly, once the work is approved, low-fidelity user interface mock-ups are needed as an instrument for discussing detailed system design and requirements. Throughout the early design phase, business analysts, subject matter experts, architects, and UCD designers gradually and repeatedly discuss over different parts of the UI design to comb out detailed requirements, conflicts of understanding, and realization of the need for further investigations. The UI design repeatedly changes over the course of this activity. The essence of using low-fidelity mock-ups as a way to share the understanding of a solution is monumental in our experience. Lastly, once the design starts to firm up, user-centered designers review low-fidelity UI mock-ups with the users for early design feedback. This has to be done as quickly as possible as the development team is often ready and waiting for inputs at this stage.

The needs for UI mock-ups mentioned above do not require a high fidelity design really. That is, not all buttons have to be in the right place, not all UI components
have to be in their final locations. The mock-ups do not need styles and the finished look. The mock-ups are used for the purpose of understanding the process in different lights. For business executives who view the UI with a business process, the low-fidelity mock-ups merely gives them another, yet very powerful, perspective to the process flow and interactions between people in the process. For business analysts and subject matter experts (SMEs), low-fidelity UIs are just a matter of making sure the required business functions are supported and information is presented with the proper semantics. For the end users, designers solicit their ability to comprehend the design through low-fidelity UI mock-ups. Designers focus on whether the users can understand the information presented, whether they can perceive what needs to be done, whether some information is missing, and whether certain information is not necessary. We do not want to present high-fidelity UI design to these various reviewers as they may be caught in unnecessary cosmetic discussions [3].

We hope to have convinced the readers at this point the increasing demand for quick low-fidelity UI mock-ups and how important these mock-ups are to the early design process. Lack of proper tooling support for the early UI design, as evidently from our experience, makes UI mocking up often appears to hold up the process. Our current focus is on enabling a fast UI mocking up tool leveraging business process models while maintaining the user-centered design perspectives. We at the same time focus on helping designers manage a large design space through our tools. Our other goal not emphasized in this paper is to support the end-to-end UI design process to gradually change from low fidelity UI designs towards high fidelity UI for production environments. This latter endeavor requires a clean separation of concerns from people with different skill sets, i.e. UCD designers and developers, as moving from design to development increasingly adds technical complexity. Our ultimate goal is to encompass within our model and tools sufficient support for low fidelity design thinking and management, as well as the high fidelity UI production process while maintaining the overall usability of the tools.

2 Our Approach

In this section, we will walk the readers through our end-to-end approach. We start from a business model (Section 2.1) and transform it into various views in our user-centered Human Interactions (HI) perspectives (Section 2.2). These views are information tools designed to provide user-centric information harvested from a business process. The views are also designed as places upon which a design such as pages and grouping of information. Section 2.3 describes how design pages are roughly laid out and filled with some automation help. Section 2.4 describes the user interface generation and illustrates the outputs.

2.1 Business Model as a Starting Point

Our starting point is, but not necessarily always, a business process model. Though there are several process notations out there [4,5], we find the basic information in business processes having to do with “who” will do which “task” “when” in the
al process, the sequence of business tasks, and what are the inputs and outputs of a task, to be the most useful set of information for starting a user interface design. This set of information interestingly shared some characteristics with the user task model commonly used as a starting point for user-centered design process [6]. Let’s put aside for later, if it occurs in your mind, whether business modeling and user task modeling, if done in the same tools, would yield the same “task model.” In our approach, we gladly take the business process as is and turn it into perspectives that are more in tune with the user-centered design thinking. Before we embark into further detail, let’s look at an example of an invoice-to-cash business process in Figure 1. Please note here that the complete invoice-to-cash process is rather large consisting of several process diagrams. We apologize for the intentional incompleteness of the model shown here. For confidentiality reasons, we will not reveal much further information on the entire model. Neither is there enough space in the paper to illustrate much more with legibility. However, a version of it is used in the following examples in this paper.

Fig. 1. A partial example of a business process shown in the Process Editor in IBM Websphere Business Modeler 6.0. In this figure we show a small portion of a large invoice-to-cash process in which the cash collection task on the left is the beginning of all other activities that follow.

From Figure 1, we want to point out information that is essential for understanding the following steps in our approach. Notice the “Cash Collection” task at the leftmost of the diagram. A user role “Cash Collector” is identified below the task. The cash collection task, as we learned from SMEs and user observations, typically involves the user calling customers based on “transactions,” which is one type of input to the task. Another type of input is “collection task” – a group of transactions recommended to Cash Collector (by the system) for collection. Documents that flow between tasks are referred to in this paper as “information artifacts.” Attributes of each artifact in this model and their data types can be defined through other editors in WBM. The directions of arrows indicate the directions of the process flow.
2.2 Transformation into Human Interaction Perspectives

The next step in our approach is converting the business process model into the HI perspectives to give designers a schematic summary of the process. There are 4 views in the HI perspectives:

- “User role – Artifacts” – shows artifacts needed per user role
- “Artifacts – User role” – shows per each artifact user roles with access to it
- “User role – Task” – shows business tasks required for each user role
- “User role – Design” – shows UI design pages that have been defined for each user roles. This view only has user roles and empty content initially.

Figure 2 illustrates the “User role – Artifact” view as initially populated from the business model partially illustrated in Figure 1. The view centers on user roles and shows all the artifacts that each role requires for performing their tasks. This information is derived from inputs to each business task. From the design point of view, this view gives an initial glimpse of the design scope – how many user roles am I designing for? It also gives a sub-scope for each user – what type of information is of concern for each user role. Since our tool is implemented on an Eclipse-based platform, we use a tree-like structure to present the hierarchical relationships of user roles and artifacts in the Eclipse style. The underlying information model that altogether supports the 4 views is not quite so hierarchical but rather with many interconnections. We made efforts to conceal the actual model behind the scene and at the same time conform to standard views in the Eclipse environment.

In Figure 2, notice the division of the two types of artifacts. The “Business Artifact” denotes information artifacts that flow from one task to another. The “Contextual Artifact” denotes artifacts that are inputs or outputs to but do not flow across business tasks. We use this information to guesstimate the importance of an artifact. By assuming that if an artifact flows from one task to the next, and often from one user role to another, it must be more important. Any artifact that does not flow must be contextual and is used for a particular task context. The model for the
two categories of artifacts are actually similar, the separation is purely to provide
designers with a slight cue on which artifacts might be more important.

When selecting an artifact from the tree on the left, the attributes of the artifact is
shown in an attribute editor panel. We will discuss the use of the attribute editor with
the next view.

Complimentary to the “User role – Artifacts” view is the “Artifact – User Role”
view illustrated in Figure 3. In this view, each artifact is listed with user roles that
require access to it. The view is designed to help designers with the understanding of
how artifacts are shared. By seeing how an artifact is used by more than one user
role, designers may and often opt to design similar interface for the same artifact
unless dictated otherwise by requirements or by their knowledge of the user tasks.
These two views always reflect the same information.

![Fig. 3. The “Artifact – User Role” view](image)

The latter view is also designed to reveal insights into business requirements on
read/write access to artifacts. Notice in Figure 3 that the Cash Collector has
“read/write” access to the “Transaction” and “Collection Tasks” artifacts. Under the
“Transaction” and the “Cash Collector” user role, there is a “master” copy of
“Transaction.” Any information refined for the UI in this master copy will govern all
partitions of this artifact when they apply to UI designs. This statement will be made
clearer in Section 2.3 with examples of “data groups,” subsets of this master copy
gearing for various parts of the UI design.

A user role may have “read/write” and/or “read only” master copies of an artifact.
When one or the other copy appears under a user role, it is straightforward access
permission for the role as stated. When both copies appear under the same user role,
it means the read/write permission does not prevail in all tasks and some tasks are
restricted to read only. A common use case is, for example, a contract may not be
modified after it is signed. Being explicit about read/write access brings attention to
the designer on how artifacts may be displayed in various stages of the process.
When a master copy is viewed in the attribute editor, the tasks that are applicable to
the access rights are shown as illustrated in Figure 4.
Fig. 4. Attributes of the Transaction artifact shown in the Attribute Editor after modifications.

Through the attribute editor, the designer can enrich the information by being specific whether a user role needs to see all attributes in the whole artifact or only parts of it, and which attributes are read-only thought the overall access for the artifact is read/write. The indication of access rights plays an important part of the tool’s automation of how a UI component may be associated with an attribute in a design context.

The attribute editor also provides the designer an opportunity to place a user-friendly label for each attribute, which will then be consistently used throughout the UI design. In this editor, we also introduce a notion of “UI semantic types” which entail more semantics for basic types (string, integer) of attributes. For example, a string may in deed be a “social security number.” We use UI semantic types to figure out appropriate display format, i.e., a social security number should be displayed as 222-33-4444. We also use semantic types to help pick appropriate sample data for the
UI mock-ups. Figure 4 illustrates what the designer may have done in the attribute editor to enrich the “Transaction” artifact.

The “User role – Task” view, illustrated in Figure 5, summarizes all business tasks for each user role. Each business task shows inputs, information needed for the task, and outputs, the outcomes of the task. Using this view, the designers are informed of the governing business tasks for each user and should be able to design appropriately.

We will defer the description of the “User role – Design” view to the next section.

2.3 Building Up the Design with Pages and Data Groups

The designer can now start building up the UI design. Imagine this is when a designer would start sketching out design ideas. Let’s assume after some intensive thoughts, the designer will try out a couple of pages. Figure 6 shows the skeletons of 2 pages for the Cash Collector user role. While our visual skeletal sketching tool has not been implemented at the time of the submission, here is what the designer might visualize in mind and what they might do had the tool existed.

The 2 pages would be associated to the user role Cash Collector and to the “Cash Collection” task. Figure 7 illustrates the page structure above in the “User role – Design” view. In this view, pages associated to a particular user role are specific to that user role. Pages associated with “users” at the top of the user tree are designed to be reused among multiple user roles. A page consists of “UI fragments,” which represent visual blocks as shown in Figure 6. A UI fragment is a portion of a page that displays related data attributes, UI elements, etc. Partitions of UI fragments is purely the designer’s discretion.

Next the designer focuses on the information to be placed in various UI fragments on the pages. Here we introduce a “data group” notion denoting a group of selected attributes from an artifact’s master copy for the purposes of singling them for a design context, i.e., a visual block. For example, the collection task list in the first page may contain 5 columns; each column corresponds to an attribute in the Collection Task artifact. Figure 8 illustrated the data group for such a list. Notice that in a data group, designers have limited opportunities to localize attribute enrichment, for example, the user-friendly label may change to fit a small design space. (In this example, the customer “Location” is re-labeled “Loc” so it occupies a narrower column in the
table. The resulting UI is shown in Figure 9. Read-write access can be reduced to read-only as it is more appropriate for a tabular display in this particular example. Notice that changing from read only to read/write is not allowed as doing so would violate the restrictions from the business model. For the data group in Figure 8, the sequence of the selected attributes is also rearranged as this sequence will dictate the display of information in the way the designer wishes to control the design output.

Fig. 7. Page structure defined in the “User role – Design” view. (Need new picture)

Note that designers may add attributes had they found through UCD investigations that some additional information is significantly important to the users. Attributes can be added to the “Master” copy of an artifact hence the addition can be used across the board and can be included in data groups. Contextual artifacts can also be added had the designer found artifacts undocumented in the business model that the user needs to support a task. When attributes or artifacts are added, discussions need to happen with business analysts and architects of how appropriate and plausible to support such information. The result is often a compromise.

We do not allow business artifacts to be added for obvious reasons.

Fig. 8. A data group showing selected attributes for listing Collection Task artifacts.
A page is not completed unless the content is filled. In our tool, the designer may choose to use high-level components such as a tabular list or a block display components and associate data groups to them. The block display is one of the most useful components and can be specified to spread out attribute display in 1, 2 or more columns. Low level UI components such as a button, a label, etc., can also be singly added to a page in UI fragments. Due to space limitation, we will not provide detail on this aspect of the tool.

2.4 Generating the User Interface from the Model

Fig. 9. The generated UI for the collection task list page.

Fig. 10. The generated UI for the collection detail page.

Once pages are sketched out with data groups associated to the appropriate UI components, designers can generate a user interface to see the results. The model captured through various views contains enough information to generate the basic
page layout and page transition. (The latter is not discussed in this paper.) The user-centric design model in the tool is stripped down to the more execution oriented XML model which is further used to generate the actual UI pages and connections to a simulated backend system. Figures 9 and 10 illustrate the resulting interfaces of the 2 pages mentioned in the previous section.

For each page, the XML schema contains 4 main types of information:
- Page content: describes the page layout and placements of UI elements.
- Page bindings: lists the bindings from UI elements to the data elements.
- Page interactions: defines actions and page transitions.
- Styles: low fidelity design assumes default styles. More elaborate styles can be used when extending the model to generate high fidelity UI.

We follow the consumer-provider pattern to link the UI to the backend system that provides data. The page content is used to generate the consumer code and page bindings and interactions define the provider. The generated provider mainly consists of stubs to connect to the backend which can provide actual data and other functions through any of the following: web services, simple java classes, RMI etc. In absence of real connection to the backend system, the provider gives dummy data for different UI elements. The dummy data is generated based on the semantic types associated with the attributes in the artifact model. We use a test database which contains sample data for different semantic types.

The HI model is captured in both UML2 and XML – UML mostly for supporting the design tool and XML for generating executable UIs. To generate a UI, we transform our model into a platform-specific XML for IBM Websphere Portlet Factory. The result UI runs on any application server for JSP pages, or as portlets in Websphere Process Server.

The same model can be extended to generate the high fidelity UI and backend connections to the actual web services in a production setting. This would require inputs from the designers for the style information and input from the developers on what information is passed from one page to another or from UI to backend for different events. We are currently looking further into supporting this part of the design process.

3 The Model

The HI model consists of modeling elements that captures information around user roles. Our modeling concerns cover both the support for the UI design thinking for designers as much as capturing the actual content of the visual design, navigation, and user interactions. Many modeling elements are introduced mainly to provide design analysis perspectives, flexibility in the design, and design management. A number of elements, many of which are well hidden, provide technical information to produce support for execution. Following is the list of key concepts and relations in the HI model:
• User Role: a category of targeted end users
• Business Artifact: semantically related group of information that flows from between business tasks
• Contextual Artifact: semantically related group of information that is an input or output of a task
• Attribute: properties of an artifact
• Data Group: selected attributes of one or more artifacts semantically grouped together for the purpose of displaying related information
• Page: a set of design equivalent to a full screen
• UI Fragment: a portion of a page that is used for display related information and functions
• UI Component: type of elements rendered on the screen. Basic types include text, text input, text area, link, drop down list, radio group, check box group, two way select list, calendar, etc. Complex types include tabbed blocks, list blocks, newspaper blocks and navigation blocks
• User role – Artifact: a user role that needs access to an artifact, an artifact that is used by a user role
• Artifact – attribute: has a property
• Data Group – attribute: inclusion of an attribute
• User Role – Page: A page that is designed for a user role
• Page – UI Fragment: A page is made up of UI fragments. Each UI fragment in turn contains information on it’s height, width on screen and also where it lies on the page. The position of the fragment can be defined in absolute or relative terms.
• UI Component – Data Group: A UI component is assigned to a data group, to define how the group will be laid out on screen. For each individual item in the data group the simple UI component to use is determined on the basis of its semantic type and access i.e. read or write.

4 Related Work

Recent related research with a similar goal to our work of providing UCD support is in by Campos and Nunes in [7]. In this work, they built a user-centered tool for rapid prototyping using the notations for abstract prototypes developed by Constantine [8]. Their focus is on using interaction patterns to fill in design blocks while our work fills in design blocks with the knowledge of business data elements, their read/write access as restricted by the business process, and the UI semantic types defined by the designer.

Automatic generation of user interfaces dated back to the early 90’s. In [9], a comprehensive notation for expressing the quality of data attributes was used. The automatic generation effort in this work attempted to lay out page detail as well as a page’s grouping and semantic structure; the latter task is proved to be quite easy for an expert designer to do but harder for a machine. While the work in [9] eliminated the tediousness of repeatedly laying out component in detail, the entangled
complexity of trying to solve semantic layout problem outweighed the benefits. A smaller piece of work in [10] is much more practical and use a similar technique as deployed by our work here of laying out a small number of components piecemeal.

There has been related research on generating user interfaces from task-based models. The work by Paterno in a series of publications [11, 12, 13] focused on a tool for modeling tasks and automatically generating user interfaces from high-level specification and from analyzing task structures. The task model is device independent therefore can automatically generate to multiple platforms [12]. Venderdonkt in [14] does semi-automatic generations through Automatic Interaction Objects structure. Our work uses the business task model as the input into the design process. We are using business task as requirements but currently are not using it to the full extent in helping generating the UI. We find that large enterprises have rather complex workflow and work context, and only small portion of this complexity is captured in business process model. User-centered designers still depend a great deal on closing the knowledge gap through business requirements and user observations to be able to properly complete their work.

Last but not least, a company called iRise [15] of which products center around simulating UI design and visualizing IT solutions early in the design phase. Their products are capable of generating rather attractive UI. Our work shared a common goal of the ability to visualize IT solution early. However, our work aims also at transporting the low-fidelity mock-ups into high-fidelity as well. Our low-fidelity mock-ups are executable user interface and can be beyond presentation materials.

5 Experience and Analysis

Using business models as a starting point for the UI design process is an interesting endeavor. One observation we had was that some business processes provide a rather concrete set of business tasks, to the point that designers can almost visualize what happens on the screen just from viewing a business process model. Some business processes are rather abstract. In this case, designers need to do a great deal of research to understand the nature of these abstract business tasks. The example given in this paper is an example of the latter. As business processes are not and will most likely never be user task models, we take business models merely as requirements. Composing user tasks for these business tasks are upon the discretion of the designers.

Our HI perspectives and tool are designed such that the designers can start the UI design process with or without a business process. They are also designed such that information can be added, specifically user roles, artifacts, and attributes. There are many reasons why this is a viable design. One, business models may not be available at the time designers need to start. Two, there may be yet ambiguous knowledge at the start of the design but user-centered designer may have higher confidence in the accuracy of information they have collected through user observations and could indeed start mocking up design ideas. Third, we observed that there are often additional user roles not called upon in the business model. These roles are often in managerial positions and are not really related to the operations of the model, but more so exist for reporting and monitoring capabilities. Currently we allow adding
user roles but supporting monitoring types of interaction designs has not been addressed in the current tooling.

Our current approach is top down at the moment. We understand the UI design process is bottom up, top down, and often middle out. We are looking to further support the design process by allowing designers to flexibly approach a design. Our work is still on-going. One obvious extension is the visual skeletal sketching tool and an ability to drop data groups into UI fragments. There are many other features that we have not addressed in the current tool, for example, shared templates in the design hierarchy; appropriate style sheets for low fidelity mock up, to name a few. We plan to take better advantage of business models even deeper such as using the flow to help with UI flow, using the business task tree as a place to create user subtasks, etc. Most importantly, we have not validated the tool with the general user target at this point. Our near future work includes user validation and more business cases to drive further developments.

References