D-Macs: Building Multi-Device User Interfaces by Demonstrating, Sharing and Replaying Design Actions

Jan Meskens  Kris Luyten  Karin Coninx
Hasselt University – tUL – IBBT
Expertise centre for Digital Media
Wetenschapspark 2, B-3590 Diepenbeek, Belgium
{jan.meskens,kris.luyten,karin.coninx}@uhasselt.be

ABSTRACT

Multi-device user interface design mostly implies creating a suitable interface for each targeted device, using a diverse set of design tools and toolkits. This is a time consuming activity, concerning a lot of repetitive design actions without support for reusing this effort in later designs. In this paper, we propose D-Macs: a design tool that allows designers to record their design actions across devices, to share these actions with other designers and to replay their own design actions and those of others. D-Macs lowers the burden in multi-device user interface design and can reduce the necessity for manually repeating design actions.

ACM Classification: H5.2 [Information interfaces and presentation]: User Interfaces. - Graphical user interfaces.

General terms: Design

Keywords: Design Tools, Multi-Device UI Design

Introduction

Today, there is an increasing need for application deployment and accessibility on multiple computing devices. People tend to use the same applications and services, regardless of the device at hand. A notable example for this trend is Youtube, which provides Youtube clients for a wide variety of computing platforms, including interactive TVs, different types of mobile phones, tablet PCs and traditional desktop PCs.

Manually designing a suitable User Interface (UI) for every targeted device is time consuming, complex and hard to manage when the number of target platforms increases. Indeed, during the design of a multi-device video analysis application for three target platforms (i.e. a regular desktop PC, a web application and a mobile device), we noticed that design actions often have to be repeated across the different device-specific UIs. For example, when adding a play button to the desktop UI, it would be necessary to add a similar widget to the other device-specific UIs as well. Duplicated design actions also occur when updating the properties of a widget [19]. For instance, changing the contents of a widget displaying video meta-data in one device-specific UI required making similar changes in the other device-specific UIs as well.

Current design tools do not provide support for automating repetitive design actions across device-specific UIs. Instead, multi-device design tools are mostly focused around automatically transforming (parts of) UIs from one platform to another [24, 23, 3, 4, 19, 29]. Those automatic transformations are mostly achieved by advanced algorithms or heuristics, which might be hard to predict for UI designers and can result in UIs that are not aesthetically pleasing [26].

In this paper, we explore how designers can automate repetitive design actions when creating interactive prototypes with limited functionality using a multi-device design tool. This approach is inspired by a large scale study on future design tool needs [9], which suggested that designers would benefit from tool support for (1) recording and replaying redundant design actions and (2) sharing a design’s look, feel, code, etc. between designers as sources of inspiration. Fulfilling these two needs in one design environment would result in better
tool support to realize interactive GUI prototypes on multiple devices. This can help designers to uncover design problems and to generate suggestions for new, and better, designs [14].

We propose Design tool Macros (D-Macs), a multi-device GUI builder allowing designers to automate repetitive design actions. The three major steps in the D-Macs approach are outlined in Figure 1:

- Designers demonstrate the design action sequences that have to be automated;
- Designers can share the recorded steps through a central repository with other designers;
- Designers can replay and edit their own design actions and those of others.

The main contribution in this paper is a new strategy of automating multi-device user interface design by capturing and reusing design actions across designers and user interfaces. The paper explores previous research results to find the necessary features for such a system, it contributes a tool supporting these features and a case study illustrating the effectiveness of our strategy.

RELATED WORK
The work presented in this paper is related to multi-device UI design, programming by demonstration and community shared expertise systems.

Multi-Device User Interface Design
The type of tool that is probably most explored for creating multi-device graphical user interfaces are automatic user interface generation systems, with Supple [6], XWeb [32] and PUC [28] being three well-known approaches in this domain. These systems are often referred to as model-based design tools, since designers mostly have to specify a UI by means of one or more abstract models [26]. The platform-specific UIs are then generated automatically from this abstract description. Even though model-based UI design can simplify the development of UIs for multiple devices in a few specific application domains [28, 7], this technique can be daunting for designers, since they have to master a new language to specify the high-level models and cannot directly control the look and feel of the resulting UI [26]. D-Macs does not employ abstract models or model based UI generation. Instead, it allows UI designers to work on the concrete visual design of every platform specific UI.

Recently, researchers have been exploring several approaches for building multi-device design environments. Damask [19] is one such design environment that allows designers to sketch low fidelity multi-device UI prototypes and to maintain the consistency of these UIs across devices. It employs a fixed set of UI pattern transformations and a widget mapping model to convert parts of a UI design from one device to another. D-Macs builds upon this idea by using a continuously growing set of pattern transformations. Designers can demonstrate how a pattern on one device has to be transformed into a suitable pattern for another device. These transformations can then be shared and reused in many other designs.

While Damask supports designers to sketch low fidelity prototypes, there exist several other design environments that can produce running Graphical User Interfaces (GUIs) for multiple devices and toolkits. Notable examples of such systems are SketchiXML [4, 3], Jelly [23] and Gummy [24]. These approaches include a typical GUI builder workspace to shape UIs for multiple devices and define a set of UI component transformations between device specific UIs. Instead of defining component transformations, D-Macs empowers GUI designers to demonstrate themselves how components in one device have to be transformed into suitable components on another device.

Programming by Demonstration
D-Macs records design action sequences in order to automate these actions later on. This approach is related to the ideas behind Programming by Demonstration (PBD) [18] systems, where users create programming scripts or macros by demonstrating sample input and output values. Notable PBD systems for demonstrating UI behavior are Monet [17], DEMO [33], Jade [34] and Gamut [21]. The major difficulty in PBD systems is to define how the inferred programming scripts should generalize so that they can work with new input values. This is mostly done by means of machine learning or pattern recognition algorithms, which might not produce the results as intended by the user. In contrast to most PBD systems, D-Macs does not rely on artificial intelligence algorithms that might be hard to predict for designers. Instead, it empowers the UI designers so that they can decide how a previously recorded action sequence should generalize across multiple different designs.

An important consideration in PBD systems is how the inferred behavior should be represented. Providing good feedback about what has been learned from the sample values can facilitate users to understand and eventually correct undesired system behavior [27]. Appropriate feedback also lowers the barrier for end-users (i.e. non programmers) to create scripts by demonstration. Recently, CoScripter [16] showed that scripts in human-readable text are easy to use for automating end-user interactions with web applications. Due to their comprehensibility, it is also fairly easy to discuss and share these human readable scripts across users.

D-Macs also seeks for a comprehensive feedback mechanism and accompanying notation that helps designers to easily understand their own and each other’s action sequences. For this purpose, D-Macs will rely on graphic history notations which are visual depictions of user operation histories. Notable graphic history visualizations that inspired the action visualizations in D-Macs are the action history of Adobe Photoshop5 and comic strip style notations (e.g. [8, 22, 25]), which are using sequences of screenshots together with textual descriptions.

Community Shared Expertise
In D-Macs, designers can share, discuss and reuse design actions from other UI designers. These features are inspired by systems that allow sharing knowledge across a community of system-users. Such systems have become an important approach to help users to deal with information overload and to learn from each other’s knowledge. Community shared

5http://www.adobe.com/products/photoshop/
expertise has been introduced in various application domains such as application development (e.g. HelpMeOut [12]), web browsers (e.g. CoScripter [16]), information visualization (e.g. ManyEyes [5]) and recommender systems (e.g. CommunityCommands [20]).

**DESIGN TOOL FEATURES**

In this section we present the key features of D-Macs. These features were explored in previous research and were classified as important challenges for design tool support.

**Automate Repetitive Design Actions**

When prototyping multi-device GUIs, designers often have to repeat design actions across the different designs [23, 19]. This increases the need for automation tools that can record one or more design actions, and automatically repeat those actions on many different parts of the design [9]. Automating and repeating design actions is a key feature of D-Macs and is closely related to the control generalization feature.

**Feedback**

Applying automation might result in unexpected and unpredictable output [26]. Therefore it is important that there is a properly designed feedback mechanism available so that designers can understand what the system is doing [27]. D-Macs contributes by providing multiple visualizations of the design action sequences that are part of the automation process.

**Control Generalization**

Recorded actions can be considered generalizable if they can be applied to other design contexts than the one in which these actions were captured. This is a challenging goal: many PBD systems force users to manually describe how generalization should occur, a few other systems try to infer this generalization automatically by employing heuristics or additional training data [1, 15]. D-Macs uses a mixed initiative [13] approach, that involves designers in the generalization process without forcing them to learn a complex UI. This generalization strategy is similar to the way CoScripter generalizes web interactions [16].

**Contextual Assistance**

Every action that is replayed, has certain assumptions about the UI it is applied to. For example, an action that updates a button widget can only be applied to a UI that contains a button. In order to guide designers in meeting the requirements of a loaded action sequence, D-Macs employs animated visualizations for providing contextual assistance. This contextual assistance was preferred over providing written instructions because users rarely pay attention to written instructions [10, 11].

**Design Community**

Designers often use other designer’s realizations as sources of inspiration and they need better tool support for reusing and sharing each other’s work [9]. In various application domains, it has already been shown that connecting tools with a community of tool users empowers these users to discuss and learn from each other’s experiences and expertise [16, 5, 12, 20]. D-Macs provides a platform for sharing design actions as well as to access and reuse already shared design actions. Therefore, it has to be taken into account that design actions are described in such a way that they are comprehensible by other designers as well.

**D-MACS**

In this section we discuss the most important interaction techniques and UI elements that D-Macs offers to support the desired features.

**The Multi-Device GUI Design Workspace**

When creating a multi-device UI with D-Macs, the required target platforms first have to be specified. These platforms are later shown as platform tabs on top of the D-Macs design workspace (Figure 2-A). Selecting a platform tab updates the D-Macs workspace to design UIs for the corresponding platform: it will contain a toolbox showing the available widgets for this platform (Figure 2-B); a canvas supporting direct manipulation to build a UI for this platform (Figure 2-C); and a properties panel to change the style properties of the user interface elements on the canvas (Figure 2-D).

The D-Macs usage modal is very similar to existing GUI builders. Designers can create UIs in D-Macs by placing widgets from the toolbox on the canvas and dragging them around until the resulting layout is visually appealing. This similarity with existing GUI builders allows designers to reuse their knowledge of single-platform user interface design tools in a multi-device design environment.

**Action Recording**

GUI designers can record the actions they perform in the D-Macs design environment by pressing the record button at the bottom of the action recording panel (Figure 2-E). In record mode, this panel will visualize the action sequence that is being recorded. An action sequence ends when the action recorder’s stop button is pressed. This way, designers can record multiple distinct action sequences. Figure 2 shows three recorded action sequences.

The action recorder allows designers to remove actions from a sequence or add newly recorded actions to this sequence. Actions are deleted by selecting an action and pressing the delete button. In record mode, all actions a designer performs are added to the end of the selected sequence. If a designer selects the first action of this sequence, new actions will be added at the beginning of the sequence. To insert recorded actions in the middle of two consecutive actions, the designer first has to select these two actions.

The D-Macs design environment supports a wide variety of design operations that can be recorded. Example operations include direct manipulation actions for resizing, moving or rotating widgets on the design tool’s canvas, actions for adding or removing widgets, property update actions for changing the property values in the properties panel, clipboard actions for copy and paste operations, and tab-switch actions for switching between device-specific designs.

**Action Sequence Visualizations**

D-Macs employs two visualizations to provide rich feedback about how every recorded action affects the GUI design. In one visualization, every action is shown as a combination of a human-readable textual description and a representative icon.
Figure 2: D-Macs augments a multi-device GUI builder with a design action recorder. The example user interface design shown here was inspired by the classroom interface example of Gajos et al. [6].

Figure 3: Action recordings are visualized as icons and human readable text. Every action can be visualized as a key frame animation that shows how this action affects the GUI design.

(Figure 3-top). This visualization is similar to how Adobe Photoshop\(^6\) represents its image manipulation action history.

The second visualization concerns a key frame animation that appears when pressing the filmstrip icon \(\text{play} \) on the right of an action. Key frame animations show how a recorded action changed the GUI design. The animation’s first frame shows the GUI design before the action was executed, while the last frame depicts the design after executing the action. Animations are also produced for complete action sequences and show step-by-step how all individual actions in an action sequence affect the design.

Publishing Design Action Sequences

Recorded action sequences can be published to a central repository by pressing the web icon \(\text{web} \) next to every action sequence. When publishing an action sequence, designers can add several annotations such as a title, a textual description and a number of keywords. This metadata might be useful for other designers to understand the rationale behind the sequence.

The central repository can be accessed through a web interface. This website serves as a social platform that gives an overview of all submitted action sequences and stimulates discussion among designers about these recordings. Discussing action recordings can help designers to gain a deeper understanding in each other’s actions, to improve existing action sequences and to learn from more experienced designers. The action sequences listed on this central website can be imported in D-Macs and reused by other designers.

An action sequence that appears on the central website is visualized through a comic strip notation (Figure 4). A comic strip presents how a design action sequence affects a GUI design. Every panel in the strip shows a snapshot of the design canvas and a list of the actions that were performed on this snapshot. The comic strip will only start a new panel if there is a visual difference with the canvas snapshot of the previous panel. For example, in Figure 4-2 there is no new panel created for the copy action because this does not change the

\(^6\)http://www.adobe.com/products/photoshop/
visual layout. Clicking on a strip panel allows designers to add comments about the actions in this panel. An overview of these discussions is available through the discussion tab.

The strip panel and according discussions can be accessed in D-Macs as well. Clicking on the comments icon next to every action shows the discussion about this action. The comic strip also corresponds to the previously introduced key frame animations that appear after clicking on the filmstrip icon next to every action sequence. Inside D-Macs, key frame animations were favoured over comic strips because they are more compact and do not occlude the design workspace.

Reusing Actions in Different Designs
A sequence is replayed in D-Macs by loading the action sequence from the repository and then pressing the play button in the action recorder. During action replay, D-Macs employs its own mixed-initiative generalization mechanism that allows combining automated actions with manual design actions.

The cornerstone of D-Macs’s generalization approach is the state icon (Figure 5) that indicates for every action if it has to be replayed automatically by the D-Macs system or manually by the UI designer. Designers can easily toggle between the manual and automatic state icon by clicking on it. Manual actions are typically used for the design steps that are difficult to replay in other designs than the one in which they were recorded. When loading an action sequence, D-Macs will automatically put the state of these difficult-to-automate actions to manual.

An action that is typically hard to replay automatically is a selection action. Under the hood, this action selects a component of a certain type at a specific X,Y coordinate. This action will be hard to replay because there is no guarantee about the location of the desired component in other designs. Figure 5 shows an example of how we employ the D-Macs mixed initiative approach to solve this automation problem. In this example, we first set the state of the panel selection action to manual. This allows designers to select manually the panel of the current design that has to be affected by the action sequence. After selecting this panel, the remainder of this sequence will take place automatically.

Handling Action Errors
When reusing previously recorded action sequences in other GUI designs, action errors might occur. One example is the selection action that was discussed in the previous section. Another example concerns an action sequence that starts with changing the layout of the selected panel. To execute correctly, this action requires that there is a panel selected in the current design. Because we do not want to expose action errors to the designers, D-Macs contains a smooth error assistance mechanism. Assistance is provided by means of two visualizations: suggestions and highlighted regions. Each time an action cannot be executed, suggestions for resolving this error appear below the action. When clicking on a suggestion, all parts of the D-Macs workspace that are needed to follow this suggestion are highlighted using a fade-in/fade-out animation. For example, the action in Figure 6-1 cannot be executed and the suggestion to “Please select a javax.swing.JPanel” is shown below this action. When clicking on this suggestion, all possible panels in the current GUI design are highlighted in or-
Figure 6: D-Macs provides suggestions (1) for resolving errors that might occur during action replay. When clicking on a suggestion, all parts of the D-Macs design workspace that contribute to follow this suggestion are highlighted (2).

ARCHITECTURE AND IMPLEMENTATION
This section describes D-Macs’ underlying architecture and discusses the details of its implementation.

Design Environment
The D-Macs design environment is implemented using the Jelly multi-device design framework that is written in .NET [23]. This framework provides a graphical design environment that communicates with different platform-specific UI renderers through a networked protocol. These renderers can be implemented in any programming language and are responsible for showing the visual design that can be perceived in the design environment’s canvas.

By default, Jelly includes renderers that support the design of Windows Mobile, Adobe Flex and Java Swing user interfaces. We created an additional renderer to target the .NET Windows Presentation Foundation (WPF).

Design Action Recording
Action recording is supported through the “Command Object Model” [2]. Every action supported by our GUI builder is implemented as a command object. Each time a designer interacts with the design environment (Figure 7-1) to perform a design action, there is a command object executed (Figure 7-2). As commands are executed, they are stored in a command list which serves as a history of all actions that have been taken (Figure 7-4). This command list can then be published on the central D-Macs repository (Figure 7-5).

Every command object contains a textual and visual description of the interaction that triggers this command. This textual description is supplied as human readable text during action recording. The visual description on the other hand is constructed by a screen grabber (Figure 7-3) which takes snapshots of the design tool’s canvas before and after command execution. This combination of visual and textual information is employed by D-Macs to visualize the actions in the action recorder panel (Figure 3), to show the key frame animation of every action (Figure 3) and to build up the comic strip notations on the central repository (see Figure 4).

Replaying Design Actions
In order to replay design actions, a previously recorded command list is first loaded into D-Macs. Each command object in this list contains a set of fields (Figure 8-1) describing the content that is needed to execute this command. For example, a property update command contains two fields: one for the property name that has to be updated, and one for the value that has to be assigned to this property.

Command objects contain a list of state constraints (Figure 8-2). These constraints will check several aspects of the current UI design state in order to determine whether it is possible to execute the command object. For example, a property update action contains a state constraint that checks if the currently selected component has a property that is named like its property name field.

Only if all state constraints are satisfied, a command object can be executed. If one or more state constraints cannot be satisfied, they will produce one or more suggestions (Figure 8-6). These suggestions are human readable descriptions that describe how the UI design state should be changed in order to satisfy the state constraint. Every suggestion also involves a number of highlights (Figure 8-7), which describe design workspace regions that have to be highlighted in order to track the designer’s attention. Example regions that can be highlighted are the design tabs, property fields, UI components on the canvas, UI components in the toolbox, etc.

Once all constraints are satisfied, the actual command object execution can take place. This can be done either automatically or manually. When automatic execution is chosen (Fig-
The Desktop News Video Explorer can be reused in another design. In a second example, we show how a previously recorded action sequence can be reused in another design.

EXAMPLE USER INTERFACES
To exemplify the use of D-Macs, this section describes several case studies of multi-device user interfaces we created with D-Macs. The first example shows how a part of a UI can be transformed from one platform to another. In a second example, we show how a previously recorded action sequence can be reused in another design.

News Video Explorer

Desktop The news video explorer is a desktop application that provides an overview of a news video. It is built around an annotated timeline which outlines the most important key frames of the video in chronological order (Figure 9-1). This timeline can be used to scroll through the video in order to discover a desired fragment of interest. The news video explorer is implemented in .NET using the Windows Presentation Foundation (WPF) toolkit.

D-Macs has been used to transform this annotated timeline into a web-based fish eye view visualization (Figure 9-2) and into a mobile key frame list (Figure 9-3). While the former is realized using the Adobe Flex toolkit, the latter is developed for a Windows Mobile device with the compact .NET framework.

Web For creating the web-based fish eye view, we first outlined an empty fish eye view component on the Flex design. This fish eye view component visualizes an array of images in a grid. We then started to record a design action sequence, which is shown in Figure 9-arrow A. This action sequence describes the selection of a key frame image (which is a

7 System.Windows.Controls.Image component) in the desktop .NET design. The value of the source property, which contains the file path of the image, is then copied to the clipboard. Next, this value is pasted in the fish eye view’s image list on the Adobe Flex design. The image will then appear in the fish eye view’s grid.

The recorded action sequence now has to be repeated for all images in the timeline. To automate the sequence, the state icon of the key frame selection action is toggled to manual (Figure 9-arrow A). The action can now be replayed multiple times. Each time D-Macs reaches the key frame selection action, the designer just has to select the desired key frame that has to be added to the fish eye view.

Mobile The mobile version of the news explorer is created by first outlining a scrollable, vertically oriented, custom flow layout panel on the mobile UI. The recorded action sequence for adding the timeline key frames into this list is outlined in Figure 9-arrow B. The beginning of this action sequence is similar to the previously discussed action sequence for creating the web application. In the last two actions, a System.Windows.Forms.PictureBox widget is added to the flow layout panel and the file path is copied into this widget’s image property.

Action sequence B is automated in a similar way as action sequence A. The state icon of the second action is toggled to manual. Each time this action sequence is executed, designers have to select manually the key frame that has to appear in the mobile UI.

Image Gallery

To demonstrate how action sequence A (Figure 9-arrow A) can be reused in another UI, we introduce a second example application: the image gallery (Figure 10-top). This image gallery is a desktop application created in .NET using the Windows Presentation Foundation (WPF) toolkit. It organizes image thumbnails as a card deck and allows users to view a full sized image by clicking on its thumbnail.

Using D-Macs, the image gallery UI is transformed into an Adobe Flex fisheye view gallery (Figure 10-bottom). For this transformation, the previously recorded action sequence A (Figure 9-arrow A) is reused. On loading this sequence, D-Macs automatically toggles the second step of this action sequence to manual. This gives designers the opportunity to select manually every image they want to use in the Adobe Flex UI. After selecting an image, the remainder of the action sequence is replayed automatically. The result of this transformation is shown at the bottom of Figure 10.

D-MACS EFFECTIVENESS ATTRIBUTES
This section discusses the effectiveness of our approach by showing that D-Macs reduces solution viscosity and supports power in combination. These are two important attributes that contribute to the effectiveness of UI design tools, as stated by Olsen [30].

Reducing Solution Viscosity

An important characteristic of a good UI design tool is that it reduces solution viscosity, which means that it minimizes the effort required to iterate on many possible design solutions.
Three ways in which a tool can reduce solution viscosity are flexibility, expressive leverage and expressive match.

**Flexibility** Because D-Macs is integrated in an environment to design UIs for many different computing platforms, it can be considered as a flexible tool. D-Macs lowers the necessity for switching between multiple incompatible design environments, which is experienced as a major burden by UI designers [9].

**Expressive Leverage** Expressive leverage [31] is where designers can accomplish more by expressing less. D-Macs achieves leverage by helping designers to avoid repetitive design steps. Designers can record an action sequence and automatically repeat these sequences multiple times. By sharing and discussing action sequences via a central repository, designers are stimulated to reuse each other’s design action sequences.

A common challenge in achieving leverage is to find a good “generalize and reuse” strategy. D-Macs contains a powerful, yet easy to control, mixed-initiative generalization mechanism. This mechanism allows easily intertwining manual and automatic design steps by toggling the state icon of each action. During these manual steps, designers complete a design step the same way as they usually work with the design environment. This does not require them to learn a new, often complicated, user interface in order to express how an action should generalize. D-Macs assists designers in two ways during this automation process. Firstly, it will state all actions that are difficult to replay as manual when loading an action sequence. Secondly, a smooth error recovering mechanism is applied to help designers in resolving problems with actions that are difficult to replay.
Expressive Match Olson [30] defines expressive match as an estimate of how close the means for expressing design choices are to the problem being solved. D-Macs adheres to this principle by employing a comic strip visualization in the central repository’s web interface. These comic strips tell the story of how a design action sequence can affect a GUI design. Every panel in the strip uses a snapshot of the UI design canvas, which is visually very close to the look and feel of the final user interface. In D-Macs, comic strips can be accessed as key frame animations that provide a compact overview of an action sequence without occluding the design workspace.

Power In Combination

Power in combination refers to the way UI tools can support new components to create new solutions. This can be supported by minimizing the cost of adding new components and by making these components easy to use and combine.

D-Macs allows designers to add custom widgets in a similar way as in traditional GUI builders. A custom widget can be used in D-Macs just by loading the library that describes this widget. This is certainly a lower cost than existing multi-device UI design tools and UI generation systems which rely on abstract UI models. Adapting these abstract models to support new components is mostly complex and does not suit UI designers, because they often do not master the model formalisms [26]. When the number of supported components grows and becomes more diverse, the effort needed to build a good abstraction model increases as well.

Multi-device design environments usually allow designers to transform UI parts from one target device to another. These transformations are often based on for designers unpredictable artificial intelligence algorithms that may produce undesired results [26]. D-Macs follows a different approach by describing and visualizing UI element transformations explicitly through understandable and adaptable action sequences. This contributes to a better mechanism for using UI elements in a multi-device design environment.

DISCUSSION AND FUTURE WORK

We introduced D-Macs, a design environment that can automate repetitive design actions. D-Macs records UI design action sequences and provides a generic reuse strategy for applying one action sequence in multiple different UI designs. This is potentially very useful when prototyping multi-device GUIs, since this type of design projects contains a high amount of repetitive design actions. In addition, D-Macs can potentially be applied to prototype single-device UIs as well. As future work, we will further investigate the use of D-Macs in general-purpose GUI design projects.

D-Macs provides a central repository where designers can share their own recordings and reuse other designers’ action sequences. In the current set-up, designers have to search the central repository and have to decide themselves which sequences are useful for their current design. We believe this effort can be lowered by automatically recommending relevant design sequences in the D-Macs workspace. Automatically recommended actions would further stimulate reuse because designers are continuously informed about the action sequences that are relevant for their current work. An interesting approach that explores this recommendation idea is CommunityCommands [20].

An important aspect of D-Macs is to guide designers in resolving errors that might occur when actions are being replayed. We currently integrated a basic visualization to demonstrate this idea and to validate the suggestions produced by D-Macs’s internal command object architecture. However, we do not aim to contribute to the field of software assistance visualizations. In a future iteration, we might consider integrating more advanced assistance visualizations (e.g., contextual video tooltips [10]).

The transformations on the image gallery and news video explorer example UIs showed the capabilities of the D-Macs approach. In a next step, we are planning to conduct a formal evaluation of D-Macs. We will continue the development of D-Macs and release it as open-source software so that other researchers and designers can benefit from its architecture and implementation.

CONCLUSION

We have explored a new multi-device user interface design approach through recording and replaying design actions. An important innovation of D-Macs is that it reuses designer’s expertise instead of complex, often unpredictable, automatic UI transformation algorithms. This is accomplished by a mixed initiative approach that divides the effort between manual design actions and automatic design decisions. Reuse is stimulated by a central web-based repository that allows designers to share and discuss recorded actions sequences. D-Macs acknowledges the importance of automation feedback and therefore provides several visualizations that help designers to understand how every automated design step works. The combination of all these functionalities makes D-Macs potentially useful in a wide variety of multi-device UI design projects.

ACKNOWLEDGEMENTS

This research was funded by the AMASS++ (Advanced Multimedia Alignment and Structured Summarization) project IWT 060051, which is directly funded by the IWT (Flemish subsidy organization). We also thank Jeffrey Nichols (IBM Research - Almaden) for his useful insights and comments about our design approach.

REFERENCES

4. Adrien Coyette, Suzanne Kieffer, and Jean Vanderdon-