Opportunities for Generative AI in UX Modernization

Stephanie Houde\textsuperscript{1,5}, Steven I. Ross\textsuperscript{1,5}, Michael Muller\textsuperscript{1,5}, Mayank Agarwal\textsuperscript{1}, Fernando Martinez\textsuperscript{4}, John Richards\textsuperscript{2}, Kartik Talamadupula\textsuperscript{3} and Justin D. Weisz\textsuperscript{2}

\textsuperscript{1}IBM Research AI, Cambridge, MA, USA
\textsuperscript{2}IBM Research AI, Yorktown Heights, NY, USA
\textsuperscript{3}IBM Research AI, Seattle, WA, USA
\textsuperscript{4}IBM Argentina, La Plata, Buenos Aires, AR
\textsuperscript{5}Authors contributed equally to the work

Abstract

The process of application modernization consumes the effort of software development teams charged with upgrading legacy applications to modern technology, architecture, and design. While some tools exist to aid in these efforts, the modernization of an application’s user experience is an arduous and primarily manual undertaking. Through a process of user research and design exploration, we investigate how generative AI models might be used to assist software development teams in modernizing the user experience of legacy applications. Our goal is to identify opportunities for further research to aid teams involved in user experience modernization efforts.

Keywords

application modernization, user experience, generative AI models, software development, design

1. Introduction

Software development has been an ongoing human activity for over 70 years [1]. While new applications continue to be created, software development teams are now often faced with the task of modernizing existing or legacy applications [2, 3, 4]. These applications may continue to have large and dedicated user bases, but the technology that they were based upon and the languages and tools that were used to construct them may now be obsolete, and the user interfaces that they present may be old-fashioned and out-dated [5]. Attempting to bring these applications up to modern standards is further hampered by inadequate or missing documentation and the loss of institutional memory that results from employee attrition and retirement [6, 7]. While all aspects of the application modernization effort present challenges, the task of modernizing the user experience (UX) is particularly daunting [7] – but also particularly necessary for customer satisfaction [8, 9] – and requires a great deal of effort involving multiple roles [10].

As part of a broader evaluation of potential uses of generative AI, in this paper we report on user research and design inspiration that explores how generative AI models might assist software engineers, product managers, and user experience designers engaged in modernizing the UX of legacy products. In this kind of modernization effort, the design and implementation of a user interface must undergo a simultaneous transformation: UX designs need to be transformed to fit a modern design system\textsuperscript{1}, and UX code implementations need to be transformed to utilize a modern UI framework\textsuperscript{2}. Our aim is to inspire development of new machine learning techniques that enable software engineers and designers to work across a multitude of UX representations, including screenshots, design mock-ups, and user interface implementations.

We have identified UX modernization as a unique opportunity area through our deliberate effort to identify technological challenges through user research. We describe the process we used to identify specific pain points and unmet needs in UX modernization in Section 4.1 and our overall observation is that it has a broad scope with many different constituent tasks and sub-tasks. We describe the as-is state of UX modernization in Section 4.2 and show a high-level overview of the process in Figure 4.

\textsuperscript{1}A "design system" is a set of visual and interactive layout, UI component, color and other standards designed to create consistent UX often across multiple displays such as web pages and mobile application screens. Examples of popular design systems include Material [11] and Carbon [12].

\textsuperscript{2}A "UI framework" is a library of code that facilitates front-end web and mobile application development. Popular UI frameworks include Bootstrap [13] and React [14].
We interviewed 12 product managers, architects, and design leads who are currently working on legacy application modernization projects. For topical completeness and consistency, we guided each informant through a series of open-ended questions. We balanced these intentional topics with ample opportunity for informants to introduce their own concerns, and to educate us about the complexities of their work.

Each interview took place online, and lasted approximately 60 minutes. Informants were invited to show as well as tell, by sharing their screen to illustrate the materials of their practices (e.g., screenshots from applications) and the ways that they worked with those materials (e.g., their own notes and proposals for further work). Each interview was led by the same researcher, and at least one other researcher attended as note-taker. In addition to the manual notes, we recorded all interviews (with informants’ permission), and we used the e-meeting software to generate a speech-to-text transcript. We used the transcript as a guide to what the informants had said, and we referred back to the recordings as necessary when the transcripts may have contained speech-recognition errors.

The majority of those we spoke to were members of a Data integration application, a Facility management application, and Asset management application product teams who are currently engaged in large modernization projects. These informants gave us a high degree of access to other project stakeholders and the content of their modernization projects, which helped us gain a more concrete understanding of the people, processes, tools, and artifacts involved in modernization. By forming a deep understanding of their work practices, we were able to identify pain points that AI technologies could help alleviate, as well as new opportunities for which generative AI technologies could be a useful technological approach.

4. Results

4.1. Identifying UX Modernization as a particular challenge

One area in which all three of these product groups are currently making massive, multi-year investments is in the modernization of the UX. In some of our first interviews we learned that the UX transformation part of the modernization process was particularly difficult and time consuming, and it did not seem to provide many opportunities for salvaging or re-use of legacy UI designs or code. One informant who is an architect called their effort “a complete re-write,” indicating that nearly all of the legacy code driving the UI had to be re-written to match a completely re-designed UX. A Program Director for the Data Integration tool volunteered that modernizing the UI is “the biggest pain point” in building their next generation application because it requires a full UX design team and
UI re-architecture, which would take as much time as designing a whole new application from scratch. This observation piqued our interest because it seemed that a legacy product’s UX ought to provide a starting point for creating a modern re-design, rather than designing and building a new UX from scratch.

In subsequent interviews, we confirmed that these large-scale UX modernization projects require high effort, large teams, and multiple years to complete. Some of the major challenges are similar to those faced when modernizing back-end systems: a lack of access to people who created the original product; a lack of product documentation, and a lack of functional specifications (e.g. validation logic of input fields). There are additional challenges due to the visual and interactive nature of the front-end: a lack of malleable design assets makes it difficult to conduct re-design work; a lack of user stories makes it difficult to understand core functionality; a lack of flow diagrams makes it difficult to understand the interaction design; a lack of a UI catalog makes it difficult to understand how UI components are used across the entirety of the UX; and the requirements to use modern, web-based, front-end UI frameworks makes it difficult to understand how to re-write legacy code (e.g. porting a native Windows application to a Web application).

4.2. As-Is: Current Process for UX Modernization

The Data Integration, Facility Management, and Asset Management application modernization projects we studied are each briefly introduced in Figures 1, 2, and 3. Although each project has its own unique goals and technical and design challenges, we found that the phases of work required across all of them share some common traits. We show a high-level overview of this process in Figure 4.

The start of each modernization project is characterized by an initial period of discovery in which product managers, designers and software architects learn about the functionality of the legacy product (Step 1). This process can be difficult due to a lack of product knowledge on the team and a lack of comprehensive product documentation. Much of this learning happens through trial-and-error usage of the product, as well as conducting Internet searches and watching online video tutorials (e.g. YouTube video demonstrations of the product).

The next phase involves planning which existing functionality will be moved into the next generation product (Step 2). Next, small functional units are specified (Step 3), designed (Step 4), and implemented (Step 5) in an iterative fashion with integrated functional and user testing as needed, until the modernization process is complete.

Overall, this process is highly manual and incurs tremendous coordination costs across designers, architects, software engineers, and product managers. All tasks – large or small – require human attention and manual work. Very little legacy code or design artifacts are reused, if they are even present. In some cases, design artifacts for the legacy application need to be created before artifacts for the transformed application can be created, due to the need to catalog and preserve the functionality of the legacy application. Then, once modernized design specification mockups have been created the process of implementing them in code is also completely manual, even though certain aspects of the UX, such as layout and UI component usage, have been precisely specified in the design files. Hence, the overall UX modernization process for an enterprise application is time consuming: each of the 3 UX modernization projects we studied has a 1-3 year roadmap for completion.

In speaking with our informants about their as-is process for conducing UX modernization, we identified a large number of opportunities where AI technologies (not just generative) could provide support. After compiling this list, we filtered it through the lens of generative techniques (as consistent with our broader research question of potential uses of generative AI) to focus on new areas and directions for generative machine learning research. At a high level, we find UX modernization to be interesting from a machine learning perspective because of the need to simultaneously co-transform multiple media types to conform to a desired state: the legacy UX design (as represented by images or design files) and the legacy UX implementation (as represented by code) need to be transformed into modernized UX design and modernized UX implementation.

We enumerate the subset of unmet needs we learned about in our user research that might be supported by generative AI innovations in Table 1, organized by the phases of the modernization process. We identify how those unmet needs may be addressed by motivating new generative AI model development.

5. Discussion

5.1. AI-Supported UX Modernization

After gaining insight into the characteristics and unmet needs of the existing UX modernization process, we sought to re-envision a new, AI-supported UX modernization process. Our aim is to apply generative technologies in an assistive or augmentative fashion, turning UX modernization into a co-creative process in which human
Figure 1: Data integration application UX Modernization project. The legacy version of the data integration application we studied was originally written 25 years ago. It offers a direct manipulation solution for defining Extract/Transfer/Load (ETL) data processing jobs. A team is currently working to build a next generation version of the product while, in parallel, continuing to support the still widely used legacy product. UX modernization plans (now partially implemented) include changing from a desktop Windows application framework to the web-based React framework [14] with the Carbon design system [12]. The team also seeks to identify pre-built platform components that can replace existing functionality in the Windows application. This project is estimated to take 1-2 years.

Figure 2: Facility management application UX modernization project. The legacy version of the facility management product we studied was originally written 20 years ago. A team is currently working to modernize its web UI while making as few functional changes as possible. The legacy web UI was implemented as Java Server Pages (JSP) with a legacy web design system. The team is working to re-implement it using the React framework and the Carbon design system. This work will be done incrementally on the live product, transforming the existing product while it is still in use. Incremental implementation has already begun and this project is estimated to be completed in 2-3 years.

Figure 3: Asset management application UX modernization project. The legacy version of the enterprise facility management product we studied is used heavily as a mobile application for equipment maintenance in the field. The focus of their UX modernization efforts are on re-designing and re-implementing their legacy mobile applications as a new web applications using the mobile version of the Carbon design system. New features and improved navigation will also be added. The modernization process is being executed on one major feature at a time. In the UI re-implementation, the architecture of front-end code is being dramatically changed to adopt a new XML-based declarative system.
and out.
application both inside
to understand the legacy
Product architect works

Legacy UX
Discovery

Modernization
Planning

Functional
Specification

Design
Specification

Next Gen UX
Implementation

Product architect works to understand the legacy application both inside and out.

Product architect, engineering architect, design leads and multiple stakeholders work to define a general strategy for how the old functionality should be mapped to the new UX, and in what order given the next gen requirements.

Product architect does a detailed review of the legacy system next in line for modernization, documents existing functionality, modifies to remove and add features according to next gen plans, and develops a user story to describe these capabilities in work items for design and development.

UX Designers define the next gen UI given the user and functional requirements. The legacy UI may be used as a visual reference during this work as they create next gen mockups that restyle but also replicate some of the existing functionality and evaluate them with users.

Software engineers implement the next gen user experience given the specifications and requirements. The legacy code base is used as a reference while functionality is reimplemented in the next gen framework.

Figure 4: A high level view of the UX modernization process. All three projects first required discovering and inventorying the user-facing capabilities presented in the legacy product (Step 1). Next, strategic plans are made (Step 2) to choose the highest impact feature(s) to modernize first, choose the functional group to modernize first, or figure out how old functionality can be replaced by a new application suite context. Once the priorities have been established, functional specifications (Step 3), design specifications (Step 4), and UX implementations (Step 5) occur in an iterative fashion, akin to traditional product development. We learned the common steps of this process from our informants. It is also documented similarly and in more technical detail in Tritchew [26].

attention is focused on work requiring creativity and judgement, and generative models are used to automate tedious and manual work (e.g., [41]). Settu and Raj describe the challenges in automating modernization, and propose guidelines for practical cases [9]. In their thesis, Nilsson explored rationales for modernization and factors to consider when planning a modernization project [42]. Examples of modernization and automation approaches appear in [4, 43, 27].

We also desire to increase the reuse of legacy assets. Already, designers and architects need to capture and preserve details of the legacy applications in order to generate functional specifications, but aspects of this work are highly automatable, such as crawling UI screens and cataloging the use of UI elements, and separating business logic from presentation logic. An important aspect of this is to transform legacy assets into the kinds of malleable intermediate representations used by designers and architects in the redesign process. For example, rather than having designers re-create representations of the legacy UI in a drawing tool such as Sketch to prepare it for transformation, generative models might transform screenshots or legacy UI code into a “first cut” representation a designer can then work with [44, 45]. Or, generative models might perform transformative operations between representations, such as converting design mockups or textual descriptions into implemented code (e.g., [25, 46]), or vice versa (e.g., [47]).

5.2. Generative Models to explore for UX Modernization

In Table 1, we identified the subset of unmet needs that we felt could be addressed by new generative models. We expand on these ideas below by describing how we might implement such models to support those unmet needs. Our observation is that UX modernization involves a large number of diverse “micro-tasks” that may or may not be needed, depending on the availability of information or documentation of the legacy product. Thus, we assert that a single UX modernization assistive tool will not be sufficient for addressing the myriad of challenges faced by practitioners. Rather, we take an approach motivated by Unix philosophy [48]: build small tools that do one task well, and enable users to chain these tools together to achieve greater behaviors. We demonstrate how an AI-supported designer might work in this new fashion in Figure 5.

---

1 For example, if a UI catalog exists for the legacy product, then cataloging the UI is not necessary.
Table 1
Unmet needs discovered through our interviews with UX modernization practitioners. Each unmet need is mapped to a potential solution, which may (or may not) involve an application of AI. For some unmet needs, there are clear mappings to potential generative models (shown in bold italic); these are extracted and discussed in further detail in section 5.2.

<table>
<thead>
<tr>
<th>Activity</th>
<th>Unmet Need</th>
<th>Solution Idea</th>
<th>Model Idea</th>
</tr>
</thead>
<tbody>
<tr>
<td>Modernization Planning</td>
<td>It takes a long time and special design and implementation skills to visualize the modernized UX for stakeholder evaluation.</td>
<td>Provide tools that convert legacy UX representations into a modern design system (e.g., transform UX screenshots to look like Carbon). This functionality may be especially useful in cases in which a minimal amount of layout and functional changes are planned, or where a quick what-if view would help the planning process.</td>
<td><strong>UI-Style-Transfer, UI-Framework-Translation</strong></td>
</tr>
<tr>
<td>Design Specification</td>
<td>All controls in the legacy application have to be redrawn in a design tool to produce an editable version of the UX.</td>
<td>Provide tools that transform legacy screens into editable representations in UX (e.g., as Sketch or Figma design files).</td>
<td><strong>Screen-to-Mockup</strong></td>
</tr>
<tr>
<td>Design Specification</td>
<td>Different products have different UI style and layout conventions. It requires a lot of manual effort to transform legacy styles &amp; layouts to product-specific modernized styles &amp; layouts.</td>
<td>Provide tools that can be used to restyle UI elements and layouts to fit the style of a specified design system and usage pattern. Note that design systems are generic, but layout and usage conventions are specific to product standards, and these should be a part of the transformation target.</td>
<td><strong>UI-Style-Transfer</strong></td>
</tr>
<tr>
<td>Next Gen UX Implementation</td>
<td>Even though design specifications are pixel-perfect, software engineers need to implement them from scratch in code.</td>
<td>Provide tools that generate UI code that match the intended design to save time by offering a starting point for further editing.</td>
<td><strong>Mockup-to-Implementation</strong></td>
</tr>
<tr>
<td>Next Gen UX Implementation</td>
<td>It’s difficult to re-purpose or re-use UI code written in a legacy UI framework.</td>
<td>Provide tools that can translate design systems from one framework to another as a starting point for further editing.</td>
<td><strong>UI-Framework-Translation</strong></td>
</tr>
<tr>
<td>Next Gen UX Implementation</td>
<td>It’s difficult to re-purpose or re-use UI code written in a legacy UI framework, when code for the UI is mixed with code that implements business logic.</td>
<td>Provide tools that can disentangle code that manages the UI from code that manages business logic.</td>
<td><strong>UI-Code-Disentanglement</strong></td>
</tr>
</tbody>
</table>

5.2.1. Screen-to-Mockup
A Screen-to-Mockup model would use a screenshot of a UI as input and generate an editable UX mock-up representation as output (e.g., Sketch file). Where available, it would be helpful to include legacy source code to improve the quality of the transformation. The Screen2Vec research by Li et al. [32] as well as the wireframe generation research by Gajjar et al. [49] suggests this goal is feasible.

5.2.2. UI-Style-Transfer
A UI-Style-Transfer model would be able to take a screenshot of a visual UI mockup as input and transform the styling of UI components to match a particular design system. For example, renderings using a legacy design system could be transformed to use a next generation design system. Additionally, any detailed mockup could be rendered as low fidelity style wireframes to facilitate discussions of functionality when UI detail is not a useful focus. Existing style transfer mechanisms (e.g., StyleGAN [50]) may be able to accomplish this task through...
fine tuning. Layout optimization capabilities such as those described by Rahman et al. [45] would be required for style transforms that include layout practices. Generative GUI design creation capabilities are also explored in research by Zhao et al. [51].

5.2.3. Mockup-to-Implementation

Given a UI design representation (e.g. Sketch file) as input, a Mockup-to-Implementation model would be able to generate UI code as output (e.g. React). Existing rule-based systems have shown this is desirable functionality [52]; we hypothesize that generative models may be able to improve the transformation process by incorporating transformed UI logic (e.g. form validation code) from legacy code. The pix2code research by Beltramelli [47] as well as the SynZ research by Sermuga Pandian et al. [53] show that deep learning models can be leveraged for this purpose as well.

5.2.4. UI-Framework-Translation

A UI-Framework-Translation model would take legacy UI code as input (e.g. Java Swing, JSP) and transform it to a new UI framework as output (e.g. React, SwiftUI). Existing state-of-the-art generative code models (e.g. [54, 55, 24]) demonstrate how to translate source code between languages, but provide very limited support for translation at the level of packages and libraries.

5.2.5. UI-Code-Disentanglement

A UI-Code-Disentanglement model would take legacy UI code as input (e.g. PHP) and transform it into two modules: code that drives display & presentation logic (e.g. the Controller in a Model-View-Controller architecture) and code that drives business logic (e.g. the Model in a Model-View-Controller architecture).

6. Conclusion

We conducted user research that enabled us to understand goals, challenges, and work practices of teams engaged in application modernization work exemplified by the three projects described in section 4.2. Within the scope of these projects, we identified UX modernization as a challenging and time consuming part of the overall modernization process that requires a lot of manual steps by multiple modernization team members including project managers, UX designers, and software engineers as illustrated in Figure 4. Current approaches to modernization [3, 2, 3, 4, 8, 56, 57, 58] revealed that UX modernization is not as well addressed as other modernization tasks such as transforming core code into microservices [43, 27, 3], and potentially holds promise as a good area to consider generative AI technology help. We identified a number of pain points within the UX modernization process observed that could potentially be addressed by new generative AI technology in the form of a set of generative AI models (described in Table 1 and in Section 5.2). Finally, in Figure 5, we used a scenario to visualize what the future of UX modernization work might be like if we were able to create and deploy those models.

Our contributions are as follows:

- Discovery of a previously undescribed pain-point and opportunity for application modernization, in the difficulty of UX modernization. This aspect of our work extends previous analyses of legacy application modernization [3, 2, 3, 4, 56, 57, 58], supplementing previous generative AI work in other aspects of software engineering [16, 19, 20, 21, 22, 23, 24].
- Documentation of three UX modernization use-cases, which require significant amounts of time and human effort.
- Proposals for new tooling that can apply contemporary AI approaches [17, 18, 19, 21, 22, 24] to UX modernization.

In future work we will evaluate the desirability of the models and applications depicted with prospective users. At the same time we will explore feasibility from a data collection and training perspective. We share these ideas with this workshop in the hope of inviting critique, discussion, and generative AI experimentation in the area of assisting the UX modernization process.
Joe is a UX designer on a team that is modernizing a legacy web application. Today he is going to design and specify an updated Preferences page. He begins by opening the legacy preferences page and reading the functional user story as well as links to legacy documentation provided by the product manager.

1. Joe uses a number of new generative AI modernization utilities as he works today. One of these is a browser plug-in that allows him to do a "deep capture" of a UX screen. He uses it to capture an image of the legacy page as well as the HTML representation of the screen, hierarchical Domain Object Mode model (DOM), text content, and any associated front-end code that can be located.

2. Next, he opens Sketch (a vector-based drawing tool that he uses to prototype UI designs). There he right clicks to access a Generative AI powered plugin utility that allows him to paste an editable version of the legacy page he just copied onto the Sketch canvas. Several different conversion style options are available. He chooses to convert the screen "as is".

3. The conversion is not perfect but it’s a good head start. She has to chase down some legacy business logic that was not picked up in the conversion because it resided in a back-end part of the code. She finds that code using another AI Utility that helps her to find and extract it.

5. In this case, Joe is just going to use the legacy screen for reference while he converts the functionality they plan to preserve into a new design. He right clicks to access a menu of style transfer capabilities offered by the plugin. He chooses to convert the style to Carbon, which is the design system of the next gen version of the application.

6. This allows him to generate a second screen mock-up next to the legacy mock-up with the same UX features mapped to their Carbon design system equivalents. The conversion is not perfect but gives Joe a head-start on different changes he needs to make to comply with style patterns that his group is using as well as functional changes specified by the product manager.

*UI-Style-Transfer

7. He replaces some radio button choices with check boxes, puts the slide-out panel in context of a new drag and drop workflow area of the tool that was not present in the legacy version, and he adds accept/cancel buttons that got lost in the transfer. After Joe has evaluated and refined the design with users and other stakeholders, he attaches the completed Sketch file to a GitHub work item already prepared by the product manager.

8. Eva, a software engineer on the development team gets the implementation task assignment. She starts by opening Joe’s Sketch file that Joe prepared using the Generative AI plugin to convert the new page mock-up—in sketch to a coded version using the React framework of the next gen application. The export capability uses the new design as well as details such as field names and validation in the legacy framework to produce the resulting code file.

*Mockup-to-Implementation

*UI-Framework-Translation

5. Eva opens the resulting code file in her IDE where she reviews what is there and tries displaying and testing the page. It’s not perfect but it’s a good head start. She has to fix some problems and also needs to chase down some legacy business logic that was not picked up in the conversion because it resided in a back-end part of the code. She finds that code using another AI Utility that helps her to find and extract it.

10. When her work is done the first version of the modernized page is ready for testing. The AI tools used along the way accelerated the time it took to get to this point compared with the way the team used to do all steps manually. Eva estimates it might make the process faster by 20% or even more!

*UI-Code-Disentanglement

**Figure 5: Future UX modernization "to-be" scenario.** In this scenario, a UX designer’s work is augmented via a suite of generative models that make it easier to move between UI representations (e.g. Screen-to-Mockup and UI Style transfer in Steps 4 and 6) as well as hand-off work to software engineers (e.g. Mockup-to-Implementation, UI Framework Translation, and UI Code Disentanglement in steps 8 and 9). *Names of future generative models, shown in bold italic text, described in more detail in Section 5.2.
References


