Opportunities for Generative AI in UX Modernization

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

[1]IBM Research AI, Cambridge, MA, USA
[2]IBM Research AI, Yorktown Heights, NY, USA
[3]IBM Research AI, Seattle, WA, USA
[4]IBM Argentina, La Plata, Buenos Aires, AR
[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[1], and UX code implementations need to be transformed to utilize a modern UI framework[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.

[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]

[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]

Joint Proceedings of the ACM IUI Workshops 2022, March 2022, Helsinki, Finland
Stephanie.Houde@ibm.com (S. Houde); steven_ross@us.ibm.com (S. I. Ross); michael_muller@us.ibm.com (M. Muller); Mayank.Agarwal@ibm.com (M. Agarwal); martferc@ar.ibm.com (F. Martinez); ajtr@us.ibm.com (J. Richards); krtalamad@us.ibm.com (K. Talamadupula); jweisz@us.ibm.com (J. D. Weisz)
0000-0002-0246-2183 (S. Houde); 0000-0002-2533-9946 (S. I. Ross); 0000-0001-7860-163X (M. Muller); 0000-0002-8442-2651 (M. Agarwal); 0000-0001-7172-4805 (F. Martinez); 0000-0001-8489-2170 (J. Richards); 0000-0002-4628-3785 (K. Talamadupula); 0000-0003-2228-2398 (J. D. Weisz)

We also provide detail on the many unmet needs we have learned about in Table 1.

Next, we begin crafting a vision for how generative AI technologies can address *some* of the unmet needs in Section 5.1, and map those onto specific kinds of generative models that could be trained to support these needs in Section 5.2.

We begin to identify opportunities to develop new kinds of generative models, based on feasibility (e.g. the accessing specific data) and potential for impact (e.g. meeting critical needs). Our philosophy is to decompose the large, generally-specified tasks within UX modernization into a suite of purpose-built generative models that can be used by designers and software engineers to accomplish specific key tasks.

2. Background

UX modernization involves both the redesign of user facing elements of a legacy application and the underlying architecture and technology that supports it. The goal is to provide users with an improved experience in terms of efficiency, productivity, responsiveness, ease of use, and enjoyment [15]. The task is complicated and constrained by the user habits and expectations that result from their experience with earlier versions of the application.

Recent work has applied a variety of generative AI and machine learning techniques to a number of software development tasks, such as code completion [16, 17, 18, 19], documentation and test generation [20, 21, 22, 23], and computer language translation [24]. Allamanis et al. [25] provide a comprehensive review of the use of AI and machine learning within software engineering. In the realm of application modernization, a variety of tools have been developed that help in the analysis and re-architecture of legacy systems [26, 27, 3]. Tools to address the modernization of the user experience, however, are notably lacking.

Since application user interfaces typically have a visual component to them, recent work in image analysis, generation, and style transfer [28, 29, 30, 31, 32] establish a baseline of capabilities that may be adapted to the UX modernization domain.

Finally, since UX modernization activities will typically involve much iteration, work on Human-AI Collaboration with generative models [33, 34, 35, 36] and the types of interaction made possible by the emergence of generative AI techniques for natural language, such as GPT-2 [37] and GPT-3 [38] may also prove to be relevant.

3. Method

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[3]; a lack of product documentation, and a lack of functional specifications (e.g. validation logic of input fields). There are also 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 shared 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[4] 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.

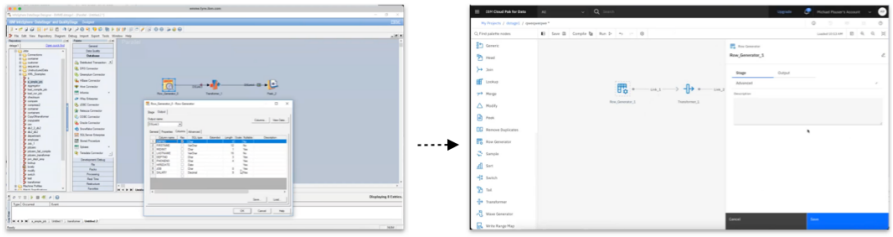
5. Discussion

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

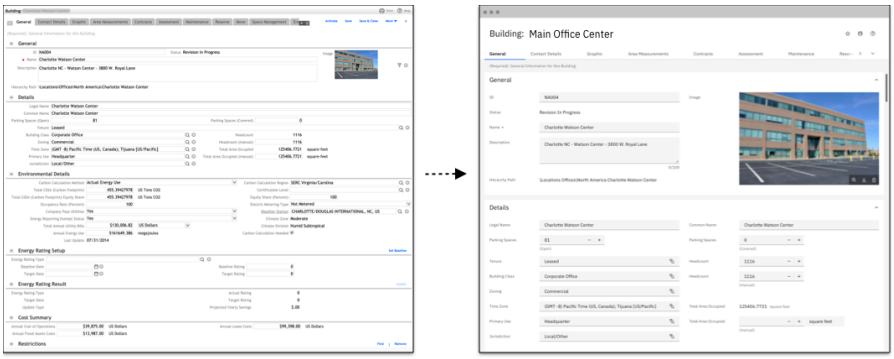
[3]For some of the products we examined, their original design dates back 20+ years.

[4]Visual specification mockups are often created in applications such as Sketch [39] and Figma [40].



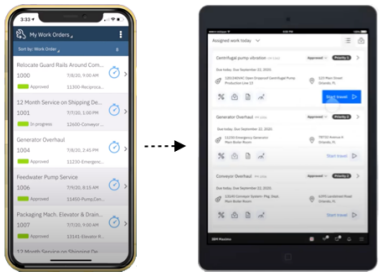
Legacy Next Generation

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 o□ers 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.



Legacy Next Generation

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.



Legacy Next Generation

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 e□orts 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.

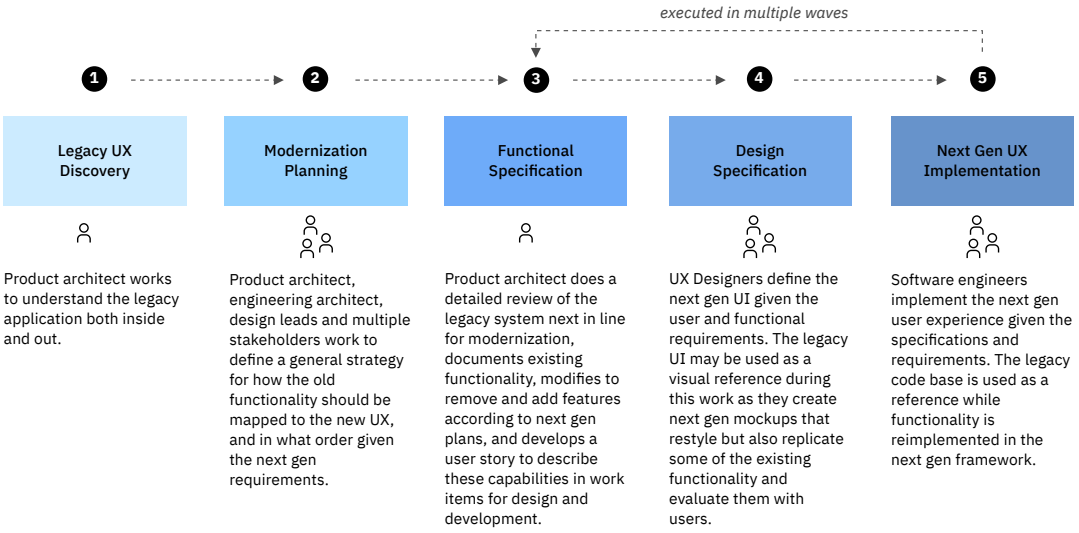


Figure 4: A high level view of the UX modernization process. All three projects first required discovering and inventorying the user-facing capabilities presented in the legacy product (**Step 1**). Next, strategic plans are made (**Step 2**) to choose the highest impact feature(s) to modernize first, 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 work 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[5]. 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.

[5]For example, if a UI catalog exists for the legacy product, then cataloging the UI is not necessary.

Activity	Unmet Need	Solution Idea	Model Idea
Modernization Planning	It takes a long time and special design and implementation skills to visualize the modernized UX for stakeholder evaluation.	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.	*UI-Style-Transfer, UI-Framework-Translation*
Design Specification	All controls in the legacy application have to be redrawn in a design tool to produce an editable version of the UX.	Provide tools that transform legacy screens into editable representations in UX (e.g. as Sketch or Figma design files).	*Screen-to-Mockup*
Design Specification	Di□erent products have di□erent UI style and layout conventions. It requires a lot of manual e□ort to transform legacy styles & layouts to product-specific modernized styles & layouts.	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.	*UI-Style-Transfer*
Next Gen UX Implementation	Even though design specifications are pixel-perfect, so□ware engineers need to implement them from scratch in code.	Provide tools that generate UI code that match the intended design to save time by o□ering a starting point for further editing.	*Mockup-to-Implementation*
Next Gen UX Implementation	It's di□icult to re-purpose or re-use UI code written in a legacy UI framework.	Provide tools that can translate design systems from one framework to another as a starting point for further editing.	*UI-Framework-Translation*
Next Gen UX Implementation	It's di□icult 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.	Provide tools that can disentangle code that manages the UI from code that manages business logic.	*UI-Code-Disentanglement*

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.

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).

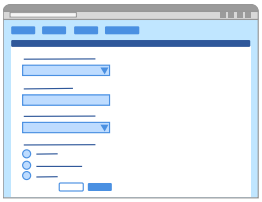
6. Conclusion

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.

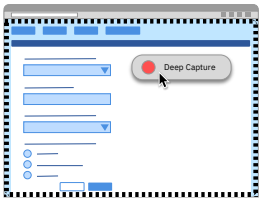
Our contributions are as follows:

- 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.

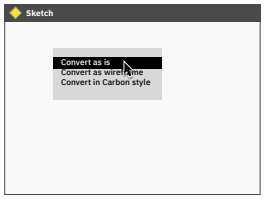
In future work we will evaluate the desirability of the models and applications depicted with prospective users. At the same time we will explore feasibility from a data collection and training perspective. We share these ideas with this workshop in the hope of inviting critique, discussion, and generative AI experimentation in the area of assisting the UX modernization process.



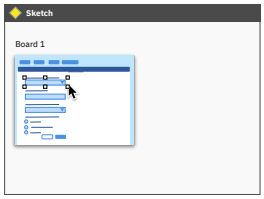
1. Joe is a UX designer on a team that is modernizing a legacy web application. Today he is going to design and specify an updated Preferences page. He begins by opening the legacy preferences page and reading the functional user story as well as links to legacy documentation provided by the product manager.



2. He will use 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 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.

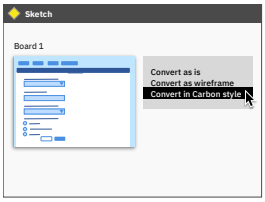


3. Next, he opens Sketch (a vector-based drawing tool that he uses to prototype UI designs). There he right clicks to access a Generative AI powered plugin utility that allows him to paste an editable version of the legacy page he just copied onto the Sketch canvas. Several different conversion style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

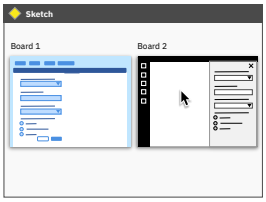


4. This converts the image into editable text and UI graphics where text and images can be examined, copied, pasted, and subsequently edited. The conversion process enabled by this paste was facilitated by the multiple image and code inputs captured earlier in the deep copy.

Screen-to-Mockup

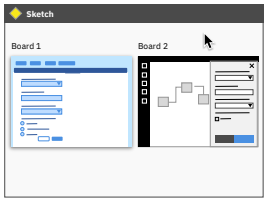


5. In this case, Joe is just going to use the legacy screen for reference while he converts the functionality they plan to preserve into a new design. He right clicks to access a menu of style transfer capabilities offered by the plugin. He chooses to convert the style to Carbon, which is the design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

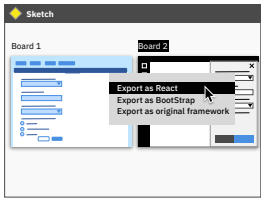


6. This allows him to generate a second screen mock-up next to the legacy mock-up with the same UX features mapped to their Carbon design system equivalents. The conversion is not perfect but gives Joe a head-start on different changes he needs to make to comply with style patterns that his group is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



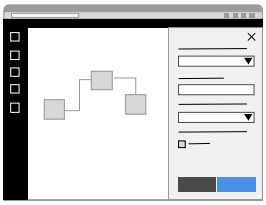
8. Eva, a software engineer on the development team gets the implementation task assignment. She starts by opening Joes's Sketch file that Joe prepared using the Generative AI plugin to convert the new page mockup-in sketch to a coded version using the React framework of the next gen application. The 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.

UI-Code-Disentanglement



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!

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-o work to so ware 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

[1] M. Yost, A brief history of software development (2018). URL: https://https://medium.com/@micahyost/a-brief-history-of-software-development-f67a6e6ddae0.

[2] S. Jain, I. Chana, Modernization of legacy systems: A generalised roadmap, in: Proceedings of the Sixth International Conference on Computer and Communication Technology 2015, 2015, pp. 62–67.

[3] A. K. Kalia, J. Xiao, C. Lin, S. Sinha, J. Rofrano, M. Vukovic, D. Banerjee, Mono2micro: an ai-based toolchain for evolving monolithic enterprise applications to a microservice architecture, in: Proceedings of the 28th ACM Joint Meeting on European Software Engineering Conference and Symposium on the Foundations of Software Engineering, 2020, pp. 1606–1610.

[4] M. Mishra, S. Kunde, M. Nambiar, Cracking the monolith: Challenges in data transitioning to cloud native architectures, in: Proceedings of the 12th European Conference on Software Architecture: Companion Proceedings, 2018, pp. 1–4.

[5] M. Fauscette, R. Perry, Simplifying it to drive better business outcomes and improved roi: Introducing the it complexity index, International Data Corporation (2014).

[6] M. Kangasaho, et al., Legacy application modernization with rest wrapping (2016).

[7] J. D. Weisz, M. Muller, S. Houde, J. Richards, S. I. Ross, F. Martinez, M. Agarwal, K. Talamadupula, Perfection not required? human-ai partnerships in code translation, in: 26th International Conference on Intelligent User Interfaces, 2021, pp. 402–412.

[8] J. Johnson, H. Mulder, Endless modernization (2020). URL: https://www.researchgate.net/profile/Hans-Mulder-2/publication/348849361_Endless_Modernization_How_Infinite_Flow_Keeps_Software_Fresh/links/60132878299bf1b33e30c29e/Endless-Modernization-How-Infinite-Flow-Keeps-Software-Fresh.pdf.

[9] R. Settu, P. Raj, Cloud application modernization and migration methodology, in: Cloud Computing, Springer, 2013, pp. 243–271.

[10] L. Tahlawi, Combining legacy modernization approaches for OO and SOA, Ph.D. thesis, University of New Brunswick., 2012.

[11] Google, Material design, n.d. URL: https://material.io/, n.d.

[12] IBM, Carbon design system, 2021. URL: https://www.carbondesignsystem.com/, accessed: 2021-12-20.

[13] O. source community, Get bootstrap, n.d. URL: https://getbootstrap.com/, n.d.

[14] Anonymous, Awesome react design systems, 2021. URL: https://github.com/jbranchaud/awesome-react-design-systems, accessed: 2021-12-20.

[15] J. Lasarte, Principles to guide your ux modernization (2018). URL: https://medium.com/headspring-ux-team/principles-to-guide-your-ux-modernization-d1d7ee56270d, [Online; accessed 16-Dec-2021].

[16] A. Hindle, E. T. Barr, Z. Su, M. Gabel, P. Devanbu, On the naturalness of software, in: 2012 34th International Conference on Software Engineering (ICSE), IEEE, 2012, pp. 837–847.

[17] V. Raychev, M. Vechev, A. Krause, Predicting program properties from" big code", in: Proceedings of the 42nd Annual ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages, 2015, pp. 111–124.

[18] M. Bruch, M. Monperrus, M. Mezini, Learning from examples to improve code completion systems, in: Proceedings of the 7th joint meeting of the European software engineering conference and the ACM SIGSOFT symposium on the foundations of software engineering, 2009, pp. 213–222.

[19] A. Svyatkovskiy, S. K. Deng, S. Fu, N. Sundaresan, Intellicode compose: Code generation using transformer, arXiv preprint arXiv:2005.08025 (2020).

[20] M. Tufano, D. Drain, A. Svyatkovskiy, S. K. Deng, N. Sundaresan, Unit test case generation with transformers, arXiv preprint arXiv:2009.05617 (2020).

[21] X. Guo, Towards automated software testing with generative adversarial networks, in: 2021 51st Annual IEEE/IFIP International Conference on Dependable Systems and Networks - Supplemental Volume (DSN-S), IEEE Computer Society, Los Alamitos, CA, USA, 2021, pp. 21–22. URL: https://doi.ieeecomputersociety.org/10.1109/DSN-S52858.2021.00021. doi:10.1109/DSN-S52858.2021.00021.

[22] L. Moreno, J. Aponte, G. Sridhara, A. Marcus, L. Pollock, K. Vijay-Shanker, Automatic generation of natural language summaries for java classes, in: 2013 21st International Conference on Program Comprehension (ICPC), IEEE, 2013, pp. 23–32.

[23] A. Y. Wang, D. Wang, J. Drozdal, M. Muller, S. Park, J. D. Weisz, X. Liu, L. Wu, C. Dugan, Themisto: Towards automated documentation generation in computational notebooks, arXiv preprint arXiv:2102.12592 (2021).

[24] B. Roziere, M.-A. Lachaux, L. Chanussot, G. Lample, Unsupervised translation of programming languages., in: NeurIPS, 2020.

[25] M. Allamanis, E. T. Barr, P. Devanbu, C. Sutton, A survey of machine learning for big code and naturalness, ACM Computing Surveys (CSUR) 51 (2018) 1–37.

[26] T. Tritchew, A multi-step incremental approach to modernization (2020). URL: https://medium.com/@tedt_39153/a-multi-step-incremental-approach-to-modernization-e6264318b462.

[27] IBM, Ibm cloud transformation advisor, 2021. URL: https://www.ibm.com/garage/method/practices/learn/ibm-transformation-advisor, accessed: 2021-12-20.

[28] Z. Zhao, X. Ma, A compensation method of two-stage image generation for human-ai collaborated in-situ fashion design in augmented reality environment, in: 2018 IEEE International Conference on Artificial Intelligence and Virtual Reality (AIVR), IEEE, 2018, pp. 76–83.

[29] K. Gregor, I. Danihelka, A. Graves, D. Rezende, D. Wierstra, Draw: A recurrent neural network for image generation, in: F. Bach, D. Blei (Eds.), Proceedings of the 32nd International Conference on Machine Learning, volume 37 of *Proceedings of Machine Learning Research*, PMLR, Lille, France, 2015, pp. 1462–1471. URL: https://proceedings.mlr.press/v37/gregor15.html.

[30] L. A. Gatys, A. S. Ecker, M. Bethge, Image style transfer using convolutional neural networks, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2016.

[31] Y. Aslam, S. N, A review of deep learning approaches for image analysis, in: 2019 International Conference on Smart Systems and Inventive Technology (ICSSIT), 2019, pp. 709–714. doi:10.1109/ICSSIT46314.2019.8987922.

[32] T. J.-J. Li, L. Popowski, T. Mitchell, B. A. Myers, Screen2vec: Semantic embedding of gui screens and gui components, in: Proceedings of the 2021 CHI Conference on Human Factors in Computing Systems, 2021, pp. 1–15.

[33] T. Weber, H. Hußmann, Z. Han, S. Matthes, Y. Liu, Draw with me: Human-in-the-loop for image restoration, in: Proceedings of the 25th International Conference on Intelligent User Interfaces, 2020, pp. 243–253.

[34] B. Shneiderman, Human-centered artificial intelligence: Reliable, safe & trustworthy, International Journal of Human–Computer Interaction 36 (2020) 495–504.

[35] J. M. Bradshaw, P. J. Feltovich, M. Johnson, Human–agent interaction, in: The handbook of human-machine interaction, CRC Press, 2017, pp. 283–300.

[36] E. Horvitz, Principles of mixed-initiative user interfaces, in: Proceedings of the SIGCHI Conference on Human Factors in Computing Systems, CHI '99, Association for Computing Machinery, New York, NY, USA, 1999, p. 159–166. URL: https://doi.org/10.1145/302979.303030. doi:10.1145/302979.303030.

[37] A. Radford, J. Wu, R. Child, D. Luan, D. Amodei, I. Sutskever, Language models are unsupervised multitask learners, OpenAI blog 1 (2019) 9.

[38] T. B. Brown, B. Mann, N. Ryder, M. Subbiah, J. Kaplan, P. Dhariwal, A. Neelakantan, P. Shyam, G. Sastry, A. Askell, et al., Language models are few-shot learners, arXiv preprint arXiv:2005.14165 (2020).

[39] Anonymous, Sketch, 2021. URL: https://www.sketch.com/, accessed: 2021-12-20.

[40] Anonymous, Figma, n.d. URL: https://www.figma.com/, accessed: 2022-12-20.

[41] D. Friedmana, D. Pollaka, Image co-creation by non-programmers and generative adversarial networks (2021). URL: http://ceur-ws.org/Vol-2903/IUI21WS-HAIGEN-4.pdf.

[42] S. Nilsson, Application modernization: Approaches, problems and evaluation, Master's thesis, Umeå University, 2015. URL: https://www.diva-portal.org/smash/get/diva2:875375/FULLTEXT01.pdf.

[43] IBM, Ibm mono2micro, 2021. URL: https://www.ibm.com/cloud/mono2micro, accessed: 2021-08-05.

[44] M. Laine, Y. Zhang, S. Santala, J. P. Jokinen, A. Oulasvirta, Responsive and personalized web layouts with integer programming, Proceedings of the ACM on Human-Computer Interaction 5 (2021) 1–23.

[45] S. Rahman, V. P. Sermuga Pandian, M. Jarke, Ruite: Refining ui layout aesthetics using transformer encoder, in: 26th International Conference on Intelligent User Interfaces - Companion, IUI '21 Companion, Association for Computing Machinery, New York, NY, USA, 2021, p. 81–83. URL: https://doi.org/10.1145/3397482.3450716. doi:10.1145/3397482.3450716.

[46] D. K. Palani, Statistical Machine Translation of English Text to API Code Usages: A comparison of Word Map, Contextual Graph Ordering, Phrase-based, and Neural Network Translations, Ph.D. thesis, Concordia University, 2018.

[47] T. Beltramelli, pix2code: Generating code from a graphical user interface screenshot, in: Proceedings of the ACM SIGCHI Symposium on Engineering Interactive Computing Systems, 2018, pp. 1–6.

[48] Wikipedia contributors, Unix philosophy — Wikipedia, the free encyclopedia, https://en.wikipedia.org/w/index.php?title=Unix_philosophy&oldid=1036396134, 2021. [Online; accessed 6-August-2021].

[49] N. Gajjar, V. P. Sermuga Pandian, S. Suleri, M. Jarke, Akin: Generating ui wireframes from ui design patterns using deep learning, in: 26th International Conference on Intelligent User Interfaces - Companion, IUI '21 Companion, Association for Computing Machinery, New York, NY, USA, 2021, p. 40–42. URL: https://doi.org/10.1145/3397482.3450727. doi:10.1145/3397482.3450727.

[50] T. Karras, S. Laine, T. Aila, A style-based generator architecture for generative adversarial networks, in: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, 2019, pp. 4401–4410.

[51] T. Zhao, C. Chen, Y. Liu, X. Zhu, Guigan: Learning to generate gui designs using generative adversarial networks, in: Proceedings of the 43rd International Conference on Software Engineering, ICSE '21, IEEE Press, 2021, p. 748–760. URL: https://doi.org/10.1109/ICSE43902.2021.00074. doi:10.1109/ICSE43902.2021.00074.

[52] Sketch2React, Build react & html prototypes in sketch, 2021. URL: http://sketch2react.io.

[53] V. P. Sermuga Pandian, S. Suleri, M. Jarke, Synz: Enhanced synthetic dataset for training ui element detectors, in: 26th International Conference on Intelligent User Interfaces - Companion, IUI '21 Companion, Association for Computing Machinery, New York, NY, USA, 2021, p. 67–69. URL: https://doi.org/10.1145/3397482.3450725. doi:10.1145/3397482.3450725.

[54] Z. Feng, D. Guo, D. Tang, N. Duan, X. Feng, M. Gong, L. Shou, B. Qin, T. Liu, D. Jiang, et al., Codebert: A pre-trained model for programming and natural languages, arXiv preprint arXiv:2002.08155 (2020).

[55] D. Guo, S. Ren, S. Lu, Z. Feng, D. Tang, S. Liu, L. Zhou, N. Duan, A. Svyatkovskiy, S. Fu, et al., Graphcodebert: Pre-training code representations with data flow, arXiv preprint arXiv:2009.08366 (2020).

[56] H. Knoche, W. Hasselbring, Using microservices for legacy software modernization, IEEE Software 35 (2018) 44–49.

[57] H. Knoche, W. Hasselbring, Experience with microservices for legacy software modernization, Software Engineering and Software Management 2019 (2019).

[58] T. Arachchi, Process of Conversion Monolithic Application to Microservices Based Architecture, Ph.D. thesis, 2021.