Model-driven development for interactive surface prototypes

EVERETT M. MTHUNZI, Bauhaus-Universität Weimar, Germany

MATHIAS MÜLLER, Hochschule für Technik und Wirtschaft Dresden, Germany

DIETRICH KAMMER, Hochschule für Technik und Wirtschaft Dresden, Germany

FLORIAN ECHTLER, Aalborg University, Denmark

Fig. 1. Interactive surface prototypes.

Interactive surface prototypes in the research community are typically application-specific. However, common features in existing artifacts suggest recurring design considerations. Given the rapid accumulation of near-identical prototypes, there is a need to promote design reuse. Existing research prototypes motivate abstracting generic components and architectural views to inform future designs. In this paper, we propose a UML-based framework for designing, documenting, and sharing interactive surface prototypes. The framework composes hardware, middleware, and software meta-models to support developers with technology-agnostic guidelines for expressing prototypes unambiguously. The meta-models are based on the study of existing prototypes and are evidence-based. For validation, we use the proposed framework to express interactive surface artifacts from prevailing studies. We then present the resulting model representations to experts to learn their perceptions towards the model-driven approach. As a proof of concept, and to show tacit benefits, we conducted a case study using ReFlex: a software framework for implementing elastic displays. Our findings highlight three significant benefits: (i) an accessible graphical syntax with unambiguous model representation, (ii) a system for capturing arbitrary technical specifications, and (iii) flexible model representation with consistent notation.

CCS Concepts: • **Hardware** → **Design reuse and communication-based design.**

Additional Key Words and Phrases: Interactive surface environments, interactive surface prototypes, UML-based framework

ACM Reference Format:

Everett M. Mthunzi, Mathias Müller, Dietrich Kammer, and Florian Ehtler. 2022. Model-driven development for interactive surface prototypes. 1, 1 (April 2022), 33 pages. <https://doi.org/10.1145/1122445.1122456>

1 INTRODUCTION

In the research space of interactive surface environments, prototyping enables the creative exploration of emerging hardware, middleware, and software. In principle, once a prototype is shared with the research community, its underlying concepts can be abstracted and reused as building blocks for future prototypes. However, in practice, replicating concepts in studies is non-trivial because research prototypes generally evaluate heuristics and usage while overlooking concept reuse. Few studies have addressed the challenge of concept reuse directly. Moreover, there is a body of literature that

indicates the accumulation of near-identical prototypes with similar limitations [3, 27, 57, 58, 65, 66, 68], suggesting a connection to the “replication crisis” in HCI research [14, 62].

Our approach to mitigating this issue considers the varying perspectives of unique developers. Different actors, e.g., system architects, embedded-systems developers, network developers, and software developers, address domain-specific challenges. While system architects are concerned with meta aspects such as sub-system infrastructure and overarching system design, embedded-system developers focus on data acquisition and performance. Also, while network developers focus on adhering to communication protocols and managing data transmission, software engineers center on data processing and data-driven functionality. Besides the focus on separate challenges, developers still need to work cohesively across the separate domains to realize a coherent interactive surface prototype. From a research perspective, developers must first review existing prototypes to underpin known limitations and knowledge gaps, irrespective of competency. While the goal may be straightforward, studying existing interactive surface prototypes is complex as there is no shared view, widespread representation, or unified syntax for documenting, sharing, and contrasting prototypes.

In this paper, we therefore propose Artefact: a UML-based framework for model-driven development of interactive surface prototypes. We first condense existing research prototypes into a knowledge base. Then, we leverage the knowledge base to formulate a flexible framework that promotes designing and expressing interactive surface prototypes unambiguously. With an emphasis on “models as end products” [56], we establish logical and consistent model representations for hardware, middleware, and software towards streamlining concept reuse. For validation, we employ the proposed framework to model prototypes from recent studies. We then subject the captured models to the scrutiny of experts and learn their perceptions towards the proposed model-driven approach. Finally, we present a proof of concept by applying the model-driven approach to our new ReFlex framework, developed to create applications for elastic displays.

2 RELATED WORK

Several studies have surveyed employing model-driven approaches for reusing artifact concepts [4, 10, 24, 33, 73]. [Genero et al.](#) [24] conducted a literature review on the role of models in development cycles and highlighted the significance of models in the iterative development of prototypes. [Ho-Quang et al.](#) [33] conducted a study across 458 projects to identify the motivation for growing interests in model-driven methods. The study suggested a causal relationship between the surge in open-source solutions and a need to communicate implementations unambiguously. [Ho-Quang et al.](#) also identified that developers utilize models to communicate structurally complex concepts to broader audiences, effectively bridging the gap between designers (i.e., originators of concepts) and future developers. In [10], [Da Silva and Oliveira](#) suggested stereotype-based modeling for concept reuse and elaborated by formulating generic artifact abstractions based on existing prototypes. The study highlighted the benefits of models as end products, underlining iterative advancement for existing artifacts. In [59], [Torre et al.](#) pointed to the widespread adoption of UML-based models and conducted a case study to investigate developer perceptions toward UML. Besides learning perceptions similar to those identified by [Ho-Quang et al.](#) in [33], the study also highlighted open and fast-growing UML-model repositories [19, 30].

[Yoshino and Matsuura](#) [73] discussed utility in employing models to reverse engineer prototypes. In a similar research context, [Bowen](#) contributed lightweight tools that enable reverse-engineering models from existing artifacts [6]. Akin to [33, 59], [Yoshino and Matsuura](#) also emphasized the role of UML models for bridging the communication gap between designers and developers. [Dittmar et al.](#) discussed how high-abstraction models can contribute to end-user programming

by enabling users to manipulate the models [12]. The authors suggested that UML-based model-driven development, in specific cases, could promote formal verification [73].

Turner et al. focused on using models for generating test cases that cover the intersection between frontend and backend components, targeting structured test coverage [60]. Similarly, Bowen and Reeves also investigated how models-specific components could be grouped to create consistent test cases across separate domains [7]. Hinze et al. presented an emulator to enable testing of user-facing components prior to implementation [32]. In the same research direction, Besnard et al. [5] discussed abstraction of physical environments to promote verification processes. Here, the authors pointed out two major challenges: First, capturing associations of components across separate domains. And second, capturing arbitrary technical specifications in abstract modules. To address these challenges, Besnard et al. suggested adopting UML mechanisms to generalize system description and proposed a technology-agnostic model-driven approach.

In the context of user interface development, several works [1, 8, 18, 23] have discussed using formal models to provide a structure for "user interface patterns", with focus on auto-generating user interface variants for multiple platforms. Luyten et al. discussed the replication crisis with respect to interactive surfaces. This state seems to have remained unchanged in the decade that has passed since this workshop [44]. Since then, a related metamodel has been proposed for pen-and-paper input. However, it has not been adapted for the more general context of interactive surfaces [31].

The related work presented in this section highlights the benefits of focusing on models as end products. There is a need to lend these benefits to the space of interactive surface environments. Facilitating a method that enables contributing models as end products would (1) promote expressing and communicating prototypes unambiguously, (2) enable systematic assessment of prototypes, and (3) facilitate concept reuse, i.e., address the replication crisis for interactive surface prototypes. We, therefore, propose the Artefact framework.

3 THE ARTEFACT FRAMEWORK

The Artefact framework enables viewing an interactive surface prototype as a system comprising of the hardware, middleware, and software subsystems. Each subsystem promotes identifying, factoring, and reusing domain-specific concepts, properties, and associations. Here, the term concept is used to refer to a component or a set of components as applied to realize domain-specific functionality. The framework employs UML [52] to represent each subdomain as an unambiguous model. UML provides a formal syntax which promotes logical and consistent model representation of interactive surface prototypes. In addition to generalization hierarchies that facilitate flexibility, another significant benefit to UML is its extensive documentation [52].

In the following, we introduce the Artefact framework and how it leverages stereotype-based modeling. We also discuss our approach to establishing evidence-based modeling constructs for the proposed hardware, middleware, and software models.

3.1 Stereotype-based modeling

The Artefact framework employs metaclasses, stereotypes, profiles, and tag values; UML mechanisms that extend the normative UML [52]. In contrast to the conventional *Class*, i.e., from normative UML, Metaclasses are abstract *Class* representations. Profiles are package modules required to define stereotypes: concrete *Metaclass* extensions, that can only be defined by extending existing metaclasses. Profiles can extend other profiles and can also be factored for

Fig. 2. Overview of the Artefact framework. For an interactive surface prototype, we describe the hardware, middleware, and software as complementary models where, a *Model* is an abstract description of a domain's environment, i.e., the organization of its concepts and associations.

reuse [52]. Akin to a normative UML *Class*, a *Stereotype* also uses structural compartments for describing attributes, members, and tag values [52].

Fig. 3. Stereotype-based modeling. Metaclasses, profiles, and stereotypes are used to express a domain's environment.

3.2 Evidence-based modeling constructs

We conducted a targeted literature review to identify hardware, middleware, and software requirements essential for developing interactive surface prototypes. The literature study prompted learning evidence-based engineering requirements, i.e., requirements based on existing research prototypes. As part of the review process, we formulated verification questions (see Tab. 1) to promote determining relevant studies. The methodology of the targeted literature review as well as an overview of the selected studies are presented in Appendix A.

Interactive surface prototypes were examined to identify hardware, middleware, and software components. Subsequently, common underlying components were abstracted into modeling constructs (see Tab. 3). As evinced by the literature study: software-design considerations for interactive surfaces are typically omitted in publications. During

Table 1. Verification questions formulated to target studies with well-rounded artifacts.

Verification questions	Points
Q1 Does the study present a clear research goal?	2
Q2 Is the presented research prototype adequate for the stated research goal?	2
Q3 Does the study outline specifications for the hardware, middleware, and software utilized?	2
Q4 Does the study justify the choice of hardware employed?	2
Q5 Does the study utilize commodity hardware?	2
Q6 Does the study motivate the software implementation?	2
Q7 Does the study employ a repeatable validation procedure?	2
Q8 Are validation metrics outlined?	2
Q9 Are limitations of the research prototypes discussed?	2
Q10 Is the efficacy of the utilized concepts outlined?	2
Q11 Are the design and implementation strategies of the software discussed?	2
Q12 Is the research goal realized?	2

study verification, no studies answered questions **Q6** and **Q11** (see Tab. 1), which were formulated to identify studies with software development strategies. This observation is again consistent with the “replication crisis” [14, 62]. Therefore, we studied the ReFlex framework to learn evidence-based software engineering requirements.

ReFlex provides technical guidelines and design considerations for developing interactive surface software. Tab. 2 presents studies that have employed the ReFlex framework to implement research prototypes [20, 40, 41, 48, 54]. In section 5 we present a case study that elaborates further on ReFlex framework. In addition to studying the ReFlex for evidence-based modeling constructs (Tab. 2), modeling constructs were also inferred from prototypical applications (see Tab. 6 in Appendix A).

Table 2. Studies that have employed the ReFlex framework to implement research prototypes.

Study	Focus	Prototype	Validation
DepthTouch: An elastic surface for tangible computing [54]	Emulating physical surface interactions	Interactive surface system for back-projection on tabletop surfaces using a mirror	Demonstration
FlexiWall: Interaction in-between 2D and 3D interfaces [20]	Toolkit for exploring layered data	Interactive surface system for back-projection on walls using a mirror	Categorization of data types and interaction techniques
Data exploration on elastic displays using physical metaphors [46]	Simulation of physical interaction metaphors on deformable interactive surfaces	Interactive surface system for tabletop surfaces	Demonstration
Exploring big data landscapes [40]	Visualization and analysis of clustering algorithms for interactive surfaces	Interactive surface system for wall surfaces	Demonstration

Continued on next page

Table 2 – continued from previous page

Study	Focus	Prototype	Validation
A tangible concept for layered map visualizations [49]	Exploring layered data combined with semantic zoom visualizations	Client-server middleware	Demonstration
Designing interfaces for elastic displays using workshop and prototyping methods [28]	Shared description for design Process	Exploring input modalities	Demonstration

Table 3. Hardware, middleware, and software modeling constructs. Common underlying components were identified generalized to stereotypes. The stereotypes were then abstracted into high-level metaclasses. For the software modeling constructs, in addition to studying the Reflex framework, flexible stereotypes were inferred based on the reverse engineering of existing software solutions.

Metaclass	Stereotype	Component	Body of literature
Hardware modeling constructs			
Device	System Processor	Workstation, Server	[13, 16, 17, 27, 29, 42, 43, 51, 53, 55, 61, 63–68]
		Microprocessor	[29, 53]
Device	Sensor	Camera Sensor	[13, 16, 27, 29, 42, 51, 53, 55, 61, 63, 66–68, 71]
		Depth Sensor	[13, 16, 29, 51, 53, 64–66, 68, 71]
		Infrared Sensor	[43, 61]
Device	Display Device	Projector, Display	[13, 17, 27, 29, 43, 51, 53, 55, 61, 63, 65–68, 71]
Device	Actuator	Actuator	[27, 43, 55, 61, 66, 68]
Device	Human Interface Device	Mobile Device	[29, 42, 51, 53, 55, 63, 66, 68]
Device	Peripheral Device	Phicon	[61]
		Marker	[42, 43, 55]
		Mirror	[17, 43, 61]
Device	Audio Device	Transducer	[17]
Middleware modeling constructs			
Architecture	Transmission Service	Framework	[13, 37]
		Specification, Protocol	[35, 36, 38]
Software modeling constructs			
Manager	Display manager		[16, 29, 42, 43, 51, 53, 63, 65, 68, 71]
Manager	Tracking manager		[16, 43, 51, 53, 63, 66, 71]
Manager	Service manager		[16, 17, 27, 29, 42, 43, 51, 53, 63, 65, 67, 71]
Manager	Calibration manager		[20, 25, 28, 40, 41, 46–50, 54]

3.3 The hardware, middleware, and software metamodels

Metamodels for the hardware, middleware, and software (see Fig. 4) were developed using the constructs presented in Tab. 3.¹ Metamodel formulation was based on outlines specified in [52], which follow: For a stereotype to be well-formed, it must exist within a *Profile* and be linked to a *Metaclass* and there are no restrictions on the number of profiles, metaclasses, and stereotypes that can be introduced [52]. Note that the profiles do not add any functionality and only describe a unified view for the underlying system. Akin to normative UML, concrete stereotypes in the hardware, middleware, and software models utilize structural-class compartments. This feature facilitates documenting arbitrary technical specifications using descriptions of attributes, members, and values. Metaclasses reflect a domain's environment and, by design, are high-level abstractions that remain adaptable for different application scenarios. The role of meta-classes is to enable modeling specifications using meta attributes, viz, tagged values. Leveraging UML's extension association, stereotypes afford the description of arbitrary information to meta-classes, i.e., using tagged values. SoaML and DICOM are good examples of how meta-classes are extended to model specifications using meta attributes [11, 74]. For engineering software solutions, the software meta-model offers a generic way to design, document, and share application programming interfaces.

4 EMPIRICAL VALIDATION

Given that there is no standard method for determining the absolute correctness of a model [52], for validation, we adopted a two-fold approach. First, the Artefact framework was employed to describe ten research artifacts, i.e., those presented in [20–22, 26, 34, 46, 49, 54, 69, 72].

The artifacts selected were subject to the selection criteria outlined for identifying relevant studies and prototypes in Appendix A. Services, formats, transmission rates, vendors, models, modes, quantities, process speeds, sampling rates, and communication interfaces were captured and expressed using tag values, directed relations, cardinalities, stereotype compartments. Once captured (graphical-model representation presented in Appendix B), we used the resulting model descriptions for validation. In [56], Sargent suggested evaluating the correctness of a model using “face validation”, i.e., subjecting a model to scrutiny of experts. Following this guideline, we conducted an interview study to determine whether experts considered the proposed modeling framework reasonable and acceptable. The study was also used to gain insights about perceptions towards adopting a model-driven approach to support developing interactive surface research prototypes.

A call for participation was sent out to research centers at KTH Royal Institute of Technology, the University of Regensburg, and Bauhaus-Universität Weimar. We also invited industry experts from Extend3D GmbH, Munich. Of twelve experts, five met the requirement of at least two years working with interactive surface environments or components associated with interactive surface environments (e.g., projector-camera systems). The interviews were conducted over Skype, and an online form was used to collect expert perceptions. Each interview was structured as follows: First, the research aim was discussed. Then, we asked the experts to confirm the suitability of their experience given the context of the research. Afterward, experts were introduced to the proposed modeling framework. Thereafter, we introduced the hardware, middleware, and software metamodels. After discussing the rationale and objectives of each model, we presented captured model representations of the prototypes. We asked the experts to scrutinize and remark on the efficacy of using the framework to model prototypes. Lastly, we discussed whether a model-driven approach would benefit the research community for developing prototypes in the early design stages. Discussion with

¹A meta-model is an abstract model that provides ontology mapping [45]. It describes modeling constructs that can be utilized to define concrete models.

Fig. 4. The hardware, middleware, and software meta-models. Profiles and metaclasses are used to describe a domain's environment. Properties and tags are used to convey arbitrary technical specifications. Directed relationships and cardinality are utilized to describe associations. Each model is flexible and extensible to variable application scenarios.

Fig. 5. Prototypes selected for validation. In (a), an overhead projector-camera system for tabletop interaction [21]. In (b), a digital tabletop (Panasonic TH-50PH12) and motion-track system for gesture detection [22]. In (c), a digital tabletop with head-mount gaze and eye-tracking for determining target interaction points [72]. In (d), wearable devices for surface interaction on surfaces [26]. In (e), a projector-camera system with a 10-camera Vicon system for tracking touch input on tabletop edges [34]. In (f), an overhead projector-camera system for tabletop and tabletop-object interaction [69]. In (g), *DepthTouch*, an elastic tabletop display [54]. In (h), *FlexiWall*, an elastic display wall for image blending [20]. In (i), Zoomable UI combined with physical interaction metaphors, called *DeeP* [46]. In (j), an elastic display using semantic layers and magic lenses [49].

all experts was structured using questionnaires. All questions were open-ended towards deeper discussions, as led by the experts. This approach promoted collecting experts' perceptions and learning their opinions on setbacks, benefits, and implications of a model-driven approach using the proposed framework.

Expert #1 remarked on the rationality of the models, "*The models are rational and clear.*" Expert #2 gave merit to unambiguous technical outlines, "*...specifications simplify prototyping.*" Expert #3 pointed to capitalizing existing prototypes, "*...developers can obtain structured information about existing prototypes...*" Expert #4 remarked on learnability, "*...it took me ~5 minutes to grasp the framework.*" Expert #5 underlined inherent fostering of artifact replication, "*...full specifications simplify replicating prototypes.*"

Data collected during the interviews was coded. Our findings suggest that experts found the proposed framework to be beneficial with perceptions converging toward three main benefits: (i) a generic and simple syntax for prototyping, (ii) an approach to systematically compare different prototypes, and (iii) a convenient starting point for developing interactive surface prototypes.

5 CASE STUDY: THE REFLEX FRAMEWORK

For proof-of-concept, we conducted a case study with software developers who have been actively developing the ReFlex framework over the last ten years. As already mentioned, ReFlex is a software framework that provides technical guidelines and design considerations for developing interactive surface software. ReFlex builds on an initial concept proposed by Peschke et al. in [54] and targets enabling the development of application programming interfaces (APIs) for interactive elastic displays. It also supports interactive visualizations using different interaction techniques [41]. The ReFlex framework has been progressively developed and used to implement several prototypes over the last ten

years [20, 40, 41, 48, 54]. ReFlex’s development has been driven by the demand to adapt APIs to rapidly evolving technology.

Given transparent software-development trails, using ReFlex for the case study enabled assessing how the Artefact framework can support developers to design and communicate interactive surface software. With an emphasis on models as end products [56], the hypothesis for the case study was as follows. “The Artefact framework provides a flexible approach to modeling different software specifications. Its software metamodel promotes clear design and unambiguous documentation of concrete APIs.”

5.1 Employing the Artefact framework to model research artifacts developed using ReFlex

The software developers selected four prototypes developed using the ReFlex framework. Akin to the validation presented in section four, APIs were captured and expressed as models. Once captured, together with the software developers, we discussed the benefits of the model-driven approach. To evaluate the hypothesis, we learned whether software developers considered the resulting software models beneficial. The case study also aimed to learn general perceptions models as artifacts. As a starting point, the software developers identified the software interfaces of the first elastic display prototype [54]. Then, they modeled the components using Artefact’s software metamodel (see Fig. 7). For comparative assessment, a similar approach was employed for remaining prototypes. Together with the software developers, we discussed the resulting model representations. We also discussed the benefit of using metamodels as flexible guidelines to support capturing and expressing APIs unambiguously.

5.2 Findings

Artefact’s software metamodel provided a flexible guideline for instantiating concrete stereotypes, thereby supporting abstract modeling of different APIs unambiguously. Moreover, the resulting model representations facilitated direct comparison between different APIs. This finding supported the first supposition: “The Artefact framework provides a flexible approach to modeling different software specifications.” The software developers emphasized the utility in knowledge transfer, pointing out the potential role of diffusing API models to bridge the gap between originators of concepts and future developers. Direct comparison between different APIs also promoted identifying advances in API development. This finding also supported the second supposition: “The Artefact’s software metamodel promotes clear

Fig. 6. ReFlex is a modular software framework for elastic displays that has been developed over the last ten years. It started as a proof-of-concept [54] and evolved into a platform-independent framework that utilizes a modular architecture with abstraction layers for different application scenarios. The framework has been used for prototypes in several publications, e.g., surface deformation interaction (left) and Zoomable user interfaces (right). It has served as a foundation for a task taxonomy for elastic displays [41]. ReFlex, in its current state, employs web-technologies for visualization, provides interface to different client technologies (e.g., Angular/React, .NET Core/WPF or Unreal Engine 4), and provides several tools for developing applications for elastic displays.

(a) Case study: software model from [54].

(b) Case study: software model from [49].

Fig. 7. While the API presented in (a) centered on processing depth data into a vector field for physical force simulation based on surface deformation, the API presented in (b) explored identifying and differentiating between single and multi-touch interactions. The API in (b) also targeted interpreting interactions to navigate layered images while utilizing a magic lens for visualization.

design as well as unambiguous documentation of concrete API designs.” Additionally, the software developers pointed out that the resulting model representations were technology agnostic. Despite employing different technologies in the different software solutions, i.e., a OpenNI-based implementation for one prototype and a .NET and WPF-based implementation for the other, the designs of the APIs remained comparable. Furthermore, the software developers pointed out that focusing on modeling high-level abstractions promoted understanding of structurally complex software concepts. Another benefit identified by the software developers was that unambiguous API models simplified identifying entry points for possible API extensions, even without comprehension of the entire API. This benefit was underlined as essential for advancing existing interactive surface concept.

Going one step further, the software developers discussed how the resulting model artifacts could, in principle, be leveraged to generate software documentation using tools such as doxygen² and graphviz³. Doxygen grouping mechanism could be utilized to declare API models, generate an overview of classes, and specify the stereotypes as nested elements. For structurally complex APIs, custom doxygen tags could be extracted for creating specific API-model components using the graphviz dot tool.

6 DISCUSSION

Given the verification questions (see Tab. 1), body of literature suggests there are no shared approaches to analyzing and designing interactive surface prototypes. The absence of such approaches is made apparent by question Q11, for which no study accounts. Also, questions Q3, Q4, Q6, and Q10, indicate there is no shared view, widespread representation, or unified syntax for designing, communicating, or documenting prototypes. The literature review also highlighted the limited extent to which middleware has been explored for interactive surface applications.

Limitations. Our validation approach suffers from the absence of a standard methodology for evaluating the “absolute” validity of models [52, 56]. Although an argument can be made for the intuitive representation of structurally complex systems, our proposed modeling framework is not comprehensive by design. Absent an extensive set of all possible abstractions, the minimal set presented in this paper is insufficient for non-experts who may also seek to employ the modeling framework as a knowledge base. Another limitation is that our validation was limited to six experts.

Future. Our empirical validation demonstrates the role of experts in the iterative development of the framework. In this regard, there is promise in conducting validation on a larger scale and learning expert perceptions towards “models as end products” [2], i.e., in the space of interactive surface environments. Moreover, experts have pointed out the benefit of open discourse about design considerations for such a modeling framework. Conducting workshops with experts and developers would be one possible approach to promote such a discussion.

7 CONCLUSION

This paper identifies the accelerated accumulation of comparable prototypes for interactive surface environments and proposes a model-driven approach to promote design reuse. Existing prototypes have been leveraged to define a generic UML-based framework for modeling hardware, middleware, and software layers of interactive surface prototypes. The proposed framework has been applied to capture existing prototypes, and a study has been conducted to learn experts’ perceptions towards the captured model representations. By modeling our new ReFlex framework for elastic displays, we further show the validity and tacit benefits to the proposed model-driven approach. Our initial findings highlight

²<https://www.doxygen.nl/index.html>

³<https://graphviz.org/>

three significant benefits: (i) an accessible graphical syntax with unambiguous model representation, (ii) a system for capturing arbitrary technical specifications, and (iii) flexible model representation with consistent notation. While no absolute conclusions can be drawn, initial results suggest significant benefits in the Artefact framework.

ACKNOWLEDGMENTS

This work was supported by the German Federal Ministry for Education and Research (BMBF) through grant 16SV8288 as part of the VIGITIA project.

REFERENCES

- [1] Nathalie Aquino. 2009. Adding Flexibility in the Model-Driven Engineering of User Interfaces. In *Proceedings of the 1st ACM SIGCHI Symposium on Engineering Interactive Computing Systems* (Pittsburgh, PA, USA) (EICS '09). Association for Computing Machinery, New York, NY, USA, 329–332. <https://doi.org/10.1145/1570433.1570496>
- [2] C Atkinson and T Kuhne. 2003. Model-driven development: a metamodeling foundation. *IEEE Software* 20, 5 (2003), 36–41. <https://doi.org/10.1109/MS.2003.1231149>
- [3] Alexander Bazo and Florian Echtler. 2014. Phone Proxies: Effortless Content Sharing between Smartphones and Interactive Surfaces. In *Proceedings of the 2014 ACM SIGCHI Symposium on Engineering Interactive Computing Systems*. ACM, New York, NY, USA, 229–234. <https://doi.org/10.1145/2607023.2610276>
- [4] Martin Becker and Andreas Schäfer. 2020. Variability Realization in UML/SysML Models. In *Proceedings of the 24th ACM Conference on Systems and Software Product Line*. ACM, New York, NY, USA, 1. <https://doi.org/10.1145/3382025.3414975>
- [5] Valentin Besnard, Frédéric Jouault, Matthias Brun, Ciprian Teodorov, Philippe Dhaussy, and Jérôme Delatour. 2020. Modular Deployment of UML Models for V and V Activities and Embedded Execution. In *Proceedings of the 23rd ACM/IEEE International Conference on Model Driven Engineering Languages and Systems: Companion Proceedings*. ACM, New York, NY, USA, 10. <https://doi.org/10.1145/3417990.3419227>
- [6] Judy Bowen. 2015. Creating Models of Interactive Systems with the Support of Lightweight Reverse-Engineering Tools. In *Proceedings of the 7th ACM SIGCHI Symposium on Engineering Interactive Computing Systems* (Duisburg, Germany) (EICS '15). Association for Computing Machinery, New York, NY, USA, 110–119. <https://doi.org/10.1145/2774225.2774840>
- [7] Judy Bowen and Steve Reeves. 2017. Generating Obligations, Assertions and Tests from UI Models. *Proc. ACM Hum.-Comput. Interact.* 1, EICS, Article 5 (jun 2017), 18 pages. <https://doi.org/10.1145/3095807>
- [8] Kai Breiner, Marc Seissler, Gerrit Meixner, Peter Forbrig, Ahmed Seffah, and Kerstin Klöckner. 2010. Pattern-Driven Engineering of Interactive Computing Systems (PEICS). In *Proceedings of the 2nd ACM SIGCHI Symposium on Engineering Interactive Computing Systems* (Berlin, Germany) (EICS '10). Association for Computing Machinery, New York, NY, USA, 367–368. <https://doi.org/10.1145/1822018.1822085>
- [9] Christian Cherek, David Asselborn, Simon Voelker, and Jan Borchers. 2019. Off-surface tangibles: Exploring the design space of midair tangible interaction. In *Extended Abstracts of the 2019 CHI Conference on Human Factors in Computing Systems*. ACM, New York, NY, USA, 1–6. <https://doi.org/10.1145/3290607.3312966>
- [10] Leandro Flores Da Silva and Edson Oliveira. 2020. Evaluating Usefulness, Ease of Use and Usability of an UML-Based Software Product Line Tool. In *Proceedings of the 34th Brazilian Symposium on Software Engineering*. ACM, New York, NY, USA, 798–807. <https://doi.org/10.1145/3422392.3422402>
- [11] Andrea Delgado, Ignacio de Guzman, Francisco Ruiz, and Mario Piattini. 2010. From BPMN business process models to SoaML service models: A transformation-driven approach. In *2nd International Conference on Software Technology and Engineering*, Vol. 1. IEEE, San Juan, PR, USA, 314–319. <https://doi.org/10.1109/ICSTE.2010.5608855>
- [12] Anke Dittmar, Alfonso García Frey, and Sophie Dupuy-Chessa. 2012. What Can Model-Based UI Design Offer to End-User Software Engineering?. In *Proceedings of the 4th ACM SIGCHI Symposium on Engineering Interactive Computing Systems* (Copenhagen, Denmark) (EICS '12). Association for Computing Machinery, New York, NY, USA, 189–194. <https://doi.org/10.1145/2305484.2305515>
- [13] Florian Echtler. 2018. Surfacestreams: A content-agnostic streaming toolkit for interactive surfaces. In *Proceedings of the 31st Annual ACM Symposium on User Interface Software and Technology*. ACM, New York, NY, USA, 10–12. <https://doi.org/10.1145/3266037.3266085>
- [14] Florian Echtler and Maximilian Häußler. 2018. Open Source, Open Science, and the Replication Crisis in HCI. In *Extended Abstracts of the 2018 CHI Conference on Human Factors in Computing Systems*. ACM, New York, NY, USA, 1–8. <https://doi.org/10.1145/3170427.3188395>
- [15] Florian Echtler and Gudrun Klinker. 2008. A multitouch software architecture. In *Proceedings of the 5th Nordic Conference on Human-Computer Interaction*, Vol. 358. Citeseer, Lund, Sweden, 463–466. <https://doi.org/10.1145/1463160.1463220>
- [16] Florian Echtler and Raphael Wimmer. 2014. The interactive dining table, or pass the weather widget, please. In *Proceedings of the 9th ACM International Conference on Interactive Tabletops and Surfaces*. ACM, New York, NY, USA, 419–422. <https://doi.org/10.1145/2669485.2669525>
- [17] Senem Ezgi Emgin, Amirreza Aghakhani, T. Metin Sezgin, and Cagatay Basdogan. 2019. HapTable: An interactive tabletop providing online haptic feedback for touch gestures. *IEEE Transactions on Visualization and Computer Graphics* 25, 9 (2019), 2749–2762. <https://doi.org/10.1109/TVCG.2018.2855154>

- [18] Jürgen Engel. 2010. A Model- and Pattern-Based Approach for Development of User Interfaces of Interactive Systems. In *Proceedings of the 2nd ACM SIGCHI Symposium on Engineering Interactive Computing Systems* (Berlin, Germany) (EICS '10). Association for Computing Machinery, New York, NY, USA, 337–340. <https://doi.org/10.1145/1822018.1822075>
- [19] Robert B France, James M Bieman, Sai Pradeep Mandalaparty, Betty H C Cheng, and Adam Jensen. 2012. Repository for Model Driven Development (ReMoDD). In *2012 34th International Conference on Software Engineering (ICSE)*. IEEE, Zurich, Switzerland, 1471–1472. <https://doi.org/10.1109/ICSE.2012.6227059>
- [20] Ingmar S Franke, Mathias Müller, Thomas Gründer, and Rainer Groh. 2014. FlexiWall: Interaction in-between 2D and 3D Interfaces. In *HCI International 2014 - Posters' Extended Abstracts*, Constantine Stephanidis (Ed.). Springer International Publishing, Cham, 415–420.
- [21] Eisuke Fujinawa, Kenji Goto, Atsushi Irie, Songtao Wu, and Kuanhong Xu. 2019. Occlusion-aware hand posture based interaction on tabletop projector. In *Proceedings of the 2nd Annual ACM Symposium on User Interface Software and Technology*. ACM, New Orleans, LA, USA, 113–115. <https://doi.org/10.1145/3332167.3356890>
- [22] Shun Fujita, Naoki Yanagihara, Buntarou Shizuki, and Shin Takahashi. 2019. Ray-casting based interaction using an extended pull-out gesture for interactive tabletops. In *Proceedings of the 31st Australian Conference on Human-Computer-Interaction*. ACM, New York, NY, USA, 546–549. <https://doi.org/10.1145/3369457.3369528>
- [23] Alfonso García Frey, Eric Cérét, Sophie Dupuy-Chessa, and Gaëlle Calvary. 2011. QUIMERA: A Quality Metamodel to Improve Design Rationale. In *Proceedings of the 3rd ACM SIGCHI Symposium on Engineering Interactive Computing Systems* (Pisa, Italy) (EICS '11). Association for Computing Machinery, New York, NY, USA, 265–270. <https://doi.org/10.1145/1996461.1996534>
- [24] Marcela Genero, Ana Fernández-Sáez, H Nelson, Geert Poels, and Mario Piattini. 2011. A Systematic Literature Review on the Quality of UML Models. *J. Database Manag.* 22 (2011), 46–70. <https://doi.org/10.4018/jdm.2011070103>
- [25] Thomas Gründer, Dietrich Kammer, Marius Brade, and Rainer Groh. 2013. Towards a design space for elastic displays. In *CHI 2013 Workshop. Displays Take New Shape: An Agenda for Future Interactive Surfaces*. ACM, New York, NY, USA, 1–4.
- [26] Yizheng Gu, Chun Yu, Zhipeng Li, Weiqi Li, Shuchang Xu, Xiaoying Wei, and Yuanchun Shi. 2019. Accurate and low-latency sensing of touch contact on any surface with finger-worn IMU sensor. In *Proceedings of the 32nd Annual ACM Symposium on User Interface Software and Technology*. ACM, New Orleans, LA, USA, 1059–1070. <https://doi.org/10.1145/3332165.3347947>
- [27] Jan Gugenheimer, Enrico Rukzio, Pascal Knierim, and Julian Seifert. 2014. UbiBeam: An interactive projector-camera system for domestic deployment. In *Proceedings of the 9th ACM International Conference on Interactive Tabletops and Surfaces*. ACM, New York, NY, USA, 305–310. <https://doi.org/10.1145/2669485.2669537>
- [28] Franziska Hannß, Mathias Müller, and Dietrich Kammer. 2021. Designing Interfaces for Elastic Displays Using Workshop and Prototyping Methods. In *Mensch und Computer 2021 - Workshopband*, Carolin Wienrich, Philipp Wintersberger, and Benjamin Weyers (Eds.). Gesellschaft für Informatik e.V., Bonn. <https://doi.org/10.18420/muc2021-mci-ws09-391>
- [29] Chris Harrison, Hrvoje Benko, and Andrew Wilson. 2011. OmniTouch : Wearable multi-touch interaction everywhere. In *Proceedings of the 24th Annual ACM Symposium on User Interface Software and Technology*. ACM, New York, NY, USA, 441–450. <https://doi.org/10.1145/2047196.2047255>
- [30] Regina Hebig, Truong Ho Quang, Michel R V Chaudron, Gregorio Robles, and Miguel Angel Fernandez. 2016. The Quest for Open Source Projects That Use UML: Mining GitHub. In *Proceedings of the ACM/IEEE 19th International Conference on Model Driven Engineering Languages and Systems*. ACM, New York, NY, USA, 173–183. <https://doi.org/10.1145/2976767.2976778>
- [31] Felix Heinrichs, Daniel Schreiber, Jochen Huber, and Max Mühlhäuser. 2011. W5: A Meta-Model for Pen-and-Paper Interaction. In *Proceedings of the 3rd ACM SIGCHI Symposium on Engineering Interactive Computing Systems* (Pisa, Italy) (EICS '11). Association for Computing Machinery, New York, NY, USA, 47–52. <https://doi.org/10.1145/1996461.1996493>
- [32] Annika Hinze, Judy Bowen, Yuting Wang, and Robi Malik. 2010. Model-Driven GUI & Interaction Design Using Emulation. In *Proceedings of the 2nd ACM SIGCHI Symposium on Engineering Interactive Computing Systems* (Berlin, Germany) (EICS '10). Association for Computing Machinery, New York, NY, USA, 273–278. <https://doi.org/10.1145/1822018.1822061>
- [33] Truong Ho-Quang, Regina Hebig, Gregorio Robles, Michel R V Chaudron, and Miguel Angel Fernandez. 2017. Practices and Perceptions of UML Use in Open Source Projects. In *Proceedings of the 39th International Conference on Software Engineering: Software Engineering in Practice Track*. IEEE, Buenos Aires, Argentina, 203–212. <https://doi.org/10.1109/ICSE-SEIP.2017.28>
- [34] Nikhita Joshi and Daniel Vogel. 2019. An evaluation of touch input at the edge of a table. In *Proceedings of the 2019 CHI Conference on Human Factors in Computing Systems*. ACM, New York, NY, USA, 2865–2874. <https://doi.org/10.1145/3290605.3300476>
- [35] Martin Kaltenbrunner. 2009. ReacTIVision and TUIO: A Tangible Tabletop Toolkit. In *Proceedings of the ACM International Conference on Interactive Tabletops and Surfaces*. ACM, New York, NY, USA, 9–16. <https://doi.org/10.1145/1731903.1731906>
- [36] Martin Kaltenbrunner, Till Bovermann, Ross Bencina, and Enrico Costanza. 2005. TUIO: A protocol for table-top tangible user interfaces. In *Proceedings of the 6th international conference on Gesture in Human-Computer Interaction and Simulation*. ACM, Berder Island, France, 1–5.
- [37] Martin Kaltenbrunner and Florian Echtler. 2015. Shared Infrastructures for Tangible Tabletops and Interactive Surfaces. In *Proceedings of the 2015 International Conference on Interactive Tabletops and Surfaces*. ACM, New York, NY, USA, 499–500. <https://doi.org/10.1145/2817721.2835070>
- [38] Martin Kaltenbrunner and Florian Echtler. 2018. The TUIO 2.0 protocol: An abstraction framework for tangible interactive surfaces. *Proceedings of the ACM on Human-Computer Interaction* 2, EICS (jun 2018), 35. <https://doi.org/10.1145/3229090>

- [39] M Kaltenbrunner, S Jorda, G Geiger, and M Alonso. 2006. The reacTable*: A collaborative musical instrument. In *Proceedings of the 15th IEEE International Workshops on Enabling Technologies: Infrastructure for Collaborative Enterprises*. IEEE, Manchester, UK, 406–411. <https://doi.org/10.1109/WETICE.2006.68>
- [40] Dietrich Kammer, Mandy Keck, Mathias Müller, Thomas Gründer, and Rainer Groh. 2017. Exploring Big Data Landscapes with Elastic Displays. In *Mensch und Computer 2017 - Workshopband*, Manuel Burghardt, Raphael Wimmer, Christian Wolff, and Christa Womser-Hacker (Eds.). Gesellschaft für Informatik e.V., Regensburg. <https://doi.org/10.18420/muc2017-ws08-0342>
- [41] Dietrich Kammer, Mathias Müller, Jan Wojdziak, and Ingmar S Franke. 2018. New Impressions in Interaction Design: A Task Taxonomy for Elastic Displays. *i-com* 17, 3 (2018), 247–256. <https://doi.org/10.1515/icom-2018-0021>
- [42] Hirokazu Kato, Mark Billinghurst, I. Poupyrev, K. Imamoto, and K. Tachibana. 2000. Virtual object manipulation on a table-top AR environment. In *Proceedings of the IEEE and ACM International Symposium on Augmented Reality*. IEEE, New York, NY, USA, 111–119. <https://doi.org/10.1109/ISAR.2000.880934>
- [43] Hideki Koike, Yoshinori Kobayashi, and Yoichi Sato. 2001. Integrating paper and digital Information on EnhancedDesk: A method for realtime finger tracking on an augmented desk system. In *ACM Transactions on Computer-Human Interaction*, Vol. 8. ACM, New York, NY, USA, 307–322. <https://doi.org/10.1145/504704.504706>
- [44] Kris Luyten, Davy Vanacken, Malte Weiss, Jan Borchers, and Miguel Nacenta. 2011. Second Workshop on Engineering Patterns for Multi-Touch Interfaces. In *Proceedings of the 3rd ACM SIGCHI Symposium on Engineering Interactive Computing Systems (Pisa, Italy) (EICS '11)*. Association for Computing Machinery, New York, NY, USA, 335–336. <https://doi.org/10.1145/1996461.1996553>
- [45] Nicholas Ross Milton. 2007. *Knowledge acquisition in practice: a step-by-step guide*. Springer, London. 176 pages. <https://doi.org/10.1007/978-1-84628-861-6>
- [46] Mathias Müller, Thomas Gründer, and Rainer Groh. [n. d.]. Data Exploration on Elastic Displays using Physical Metaphors. In *xCoAx 2015: Proceedings of the Third Conference on Computation, Communication, Aesthetics and X*.
- [47] Mathias Müller, Mandy Keck, Thomas Gründer, Natalie Hube, and Rainer Groh. 2017. A zoomable product browser for elastic displays. In *xCoAx 2017: Proceedings of the Fifth Conference on Computation, Communication, Aesthetics and X*. 127–136.
- [48] Mathias Müller, Anja Knöfel, Thomas Gründer, Ingmar Franke, and Rainer Groh. 2014. FlexiWall: Exploring Layered Data with Elastic Displays. In *Proceedings of the Ninth ACM International Conference on Interactive Tabletops and Surfaces*. ACM, New York, NY, USA, 439–442. <https://doi.org/10.1145/2669485.2669529>
- [49] Mathias Müller, Erik Lier, Rainer Groh, and Franziska Hannß. 2020. A Tangible Concept for Layered Map Visualizations: Supporting on-site Civil Engineering Construction Consultations using Elastic Displays. In *Mensch und Computer 2020 - Workshopband, Magdeburg, Germany, September 6-9, 2020*, Christian Hansen, Andreas N'urnberger, and Bernhard Preim (Eds.). Gesellschaft für Informatik e.V., Bonn. <https://doi.org/10.18420/muc2020-ws121-368>
- [50] Mathias Müller, Erik Lier, and Thomas Gründer. 2018. Zoomable User Interfaces für Elastic Displays. In *Mensch und Computer 2018 - Workshopband*, Raimund Dachelt and Gerhard Weber (Eds.). Gesellschaft für Informatik e.V., Bonn. <https://doi.org/10.18420/muc2018-ws05-0502>
- [51] Sundar Murugappan, Vinayak, Niklas Elmqvist, and Karthik Ramani. 2012. Extended Multitouch: Recovering touch posture and differentiating users using a depth camera. In *Proceedings of the 25th Annual ACM Symposium on User Interface Software and Technology*. ACM, New York, NY, USA, 487–496. <https://doi.org/10.1145/2380116.2380177>
- [52] Object Management Group. 2017. A specification defining a graphical language for visualizing, specifying, constructing, and documenting the artifacts of distributed object systems. <https://www.omg.org/spec/UML/About-UML/>
- [53] Yoon Jung Park, Hyocheol Ro, Jung-Hyun Byun, and Tack-Don Han. 2019. Adaptive Projection Augmented Reality with Object Recognition Based on Deep Learning. In *Proceedings of the 24th International Conference on Intelligent User Interfaces: Companion*. ACM, New York, NY, USA, 51–52. <https://doi.org/10.1145/3308557.3308678>
- [54] Joshua Peschke, Fabian Göbel, Thomas Gründer, Mandy Keck, Dietrich Kammer, and Rainer Groh. 2012. DepthTouch: An Elastic Surface for Tangible Computing. In *Proceedings of the International Working Conference on Advanced Visual Interfaces*. ACM, New York, NY, USA, 770–771. <https://doi.org/10.1145/2254556.2254706>
- [55] Jun Rekimoto and Masanori Saitoh. 1999. Augmented surfaces: A spatially continuous work space for hybrid computing environments. *Proceedings of the SIGCHI conference on Human Factors in Computing Systems* (1999), 378–385. <https://doi.org/10.1145/302979.303113>
- [56] R G Sargent. 2010. Verification and validation of simulation models. In *Proceedings of the 2010 Winter Simulation Conference*. IEEE, Baltimore, MD, USA, 166–183. <https://doi.org/10.1109/WSC.2010.5679166>
- [57] Dominik Schmidt, Fadi Chehimi, Enrico Rukzio, and Hans Gellersen. 2010. PhoneTouch: A Technique for Direct Phone Interaction on Surfaces. In *Proceedings of the 23rd Annual ACM Symposium on User Interface Software and Technology*. ACM, New York, NY, USA, 13–16. <https://doi.org/10.1145/1866029.1866034>
- [58] Dominik Schmidt, Julian Seifert, Enrico Rukzio, and Hans Gellersen. 2012. A Cross-Device Interaction Style for Mobiles and Surfaces. In *Proceedings of the Designing Interactive Systems Conference*. ACM, New York, NY, USA, 318–327. <https://doi.org/10.1145/2317956.2318005>
- [59] Damiano Torre, Yvan Labiche, Marcela Genero, Maged Elaasar, and Claudio Menghi. 2020. UML Consistency Rules: A Case Study with Open-Source UML Models. In *Proceedings of the 8th International Conference on Formal Methods in Software Engineering*. ACM, New York, NY, USA, 130–140. <https://doi.org/10.1145/3372020.3391554>

- [60] Jessica Turner, Judy Bowen, and Steve Reeves. 2020. Model-Based Testing of Interactive Systems Using Interaction Sequences. *Proc. ACM Hum.-Comput. Interact.* 4, EICS, Article 85 (jun 2020), 37 pages. <https://doi.org/10.1145/3397873>
- [61] Brygg Ullmer and Hiroshi Ishii. 1998. The metaDESK: Models and prototypes for tangible user interfaces. *Proceedings of the 10th annual ACM Symposium on User Interface Software and Technology* 8 (1998), 223–232. <https://doi.org/10.1145/263407.263551>
- [62] Chat Wacharamanotham, Lukas Eisenring, Steve Haroz, and Florian Echtler. 2020. Transparency of CHI Research Artifacts: Results of a Self-Reported Survey. In *Proceedings of the 2020 CHI Conference on Human Factors in Computing Systems*. ACM, New York, NY, USA, 1–14. <https://doi.org/10.1145/3313831.3376448>
- [63] Andrew D. Wilson. 2005. PlayAnywhere: A Compact Interactive Tabletop Projection-Vision System. In *Proceedings of the Annual ACM Symposium on User Interface Software and Technology*. ACM, New York, NY, USA, 83–92. <https://doi.org/10.1145/1095034.1095047>
- [64] Andrew D. Wilson. 2010. Using a Depth Camera as a Touch Sensor. In *ACM International Conference on Interactive Tabletops and Surfaces, ITS 2010*. ACM, New York, NY, USA, 69–72. <https://doi.org/10.1145/1936652.1936665>
- [65] Andrew D. Wilson and Hrvoje Benko. 2010. Combining multiple depth cameras and projectors for interactions on, above, and between surfaces. In *23rd ACM Symposium on User Interface Software and Technology*. ACM, New York, NY, USA, 273–282. <https://doi.org/10.1145/1866029.1866073>
- [66] Andrew D. Wilson, Hrvoje Benko, Shahram Izadi, and Otmar Hilliges. 2012. Steerable augmented reality with the Beamatron. In *Proceedings of the 25th Annual ACM Symposium on User Interface Software and Technology*. ACM, New York, NY, USA, 413–422. <https://doi.org/10.1145/2380116.2380169>
- [67] Benjamin Wingert, Isabel Schöllhorn, and Matthias Bues. 2017. Prodesk: An interactive ubiquitous desktop surface. In *Proceedings of the 2017 ACM International Conference on Interactive Surfaces and Spaces*. ACM, New York, NY, USA, 366–371. <https://doi.org/10.1145/3132272.3135078>
- [68] Christian Winkler, Julian Seifert, David Dobbstein, and Enrico Rukzio. 2014. Pervasive information through constant personal projection: The ambient mobile pervasive display (AMP-D). In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*. ACM, New York, NY, USA, 4117–4126. <https://doi.org/10.1145/2556288.2557365>
- [69] Qin Wu, Jiayuan Wang, Sirui Wang, Tong Su, and Chenmei Yu. 2019. MagicPAPER: Tabletop interactive projection device based on tangible interaction. In *Proceedings of the 2018 ACM International Conference on Interactive Surfaces and Spaces*. ACM, Los Angeles, CA, USA, 2. <https://doi.org/10.1145/3306214.3338575>
- [70] Robert Xiao, Chris Harrison, and Scott E. Hudson. 2013. WorldKit: Rapid and easy creation of Ad-hoc interactive applications on everyday surfaces. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*. ACM, New York, NY, USA, 879–888. <https://doi.org/10.1145/2470654.2466113>
- [71] Robert Xiao, Scott Hudson, and Chris Harrison. 2017. Supporting Responsive Cohabitation Between Virtual Interfaces and Physical Objects on Everyday Surfaces. *Proceedings of the ACM on Human-Computer Interaction* 1, EICS (jun 2017), 17. <https://doi.org/10.1145/3095814>
- [72] Xuhai Xu, Chun Yu, Yuntao Wang, and Yuanchun Shi. 2020. Recognizing unintentional touch on interactive tabletop. *Proceedings of the ACM on Interactive, Mobile, Wearable and Ubiquitous Technologies* 4, 1 (mar 2020), 24. <https://doi.org/10.1145/3381011>
- [73] Kaito Yoshino and Saeko Matsuura. 2020. Requirements Traceability Management Support Tool for UML Models. In *Proceedings of the 2020 9th International Conference on Software and Computer Applications*. ACM, New York, NY, USA, 163–166. <https://doi.org/10.1145/3384544.3384586>
- [74] Jiehan Zhou, M Rautiainen, M Ylianttila, M Foulonneau, and P Blandin. 2008. Metamodeling for Community Coordinated Multimedia and Experience on Metamodel-Driven Content Annotation Service Prototype. In *IEEE Congress on Services Part II*. IEEE, Beijing, China, 88–95. <https://doi.org/10.1109/SERVICES-2.2008.31>

A LITERATURE REVIEW

Fig. 8 provides an overview of the methodology employed for developing the Artefact framework. A targeted literature review was used to answer the question: What hardware, middleware, and software engineering requirements are necessary to develop an interactive surface prototype? This central question promoted identifying interactive surface prototypes and learning engineering requirements that have been considered for existing implementations.

In the following, we elaborate on how the targeted literature review was conducted, discussing the search strategy, selection criteria, and verification of selected literature. We also present an overview of the final studies used to develop the proposed framework.

A.1 Search strategy

The ACM Digital Library, CiteSeerX, and the IEEE Xplore Digital Library were databases of choice. Where possible, advanced searches were used to exclude non-refereed publications and books.

We inferred a list of search strings from keywords such as interactive surfaces, tabletop interaction, projector-camera systems, and spatial augmented reality. We then used the search strings to find publications in aforementioned databases.

Fig. 8. Methodology for developing the Artefact framework: First, a relative knowledge base was systematically aggregated. Then, common underlying concepts were identified and modeling constructs were derived systematically to formulate flexible hardware, middleware, and software models. In the final phase, which is perpetual by definition, the proposed framework is evaluated and developed iteratively.

Fig. 9. Targeted search. Example of how the keyword 'tabletop interaction' was used to infer a list of database search strings (e.g., "interactive tabletop", "interactive surface", and "tangible interactive surface") to identify publications.

A.2 Initial selection of related work

Besides appraising titles, we reviewed each paper's abstract, introduction, discussion, and conclusion subject to non-strict criteria that considered;

- publication—the study must be published,
- an artifact—the study must contribute a research prototype,
- peer review—the study must be subjected to the scrutiny of field experts,
- proof of concept—the study's prototype must demonstrate validity, and
- prototype description—the study must outline development in a manner that enables artifact reproducibility.

Fig. 10. Overview of excluded studies.

While rudimentary, this approach simplified excluding non-relevant studies and building up an initial body of literature to be verified.

A.3 Verification

In addition to the verification questions (see Table 1), an analysis scheme was used to assess each study. For study assessment, mark allocation was as follows: If a verification question was not answered, no points were allocated. If an answer to a question was inferred but not stated explicitly in the study, 1 point was allocated, and if a question was answered directly by the study, 2 points were allocated. Given a total number of 12 questions, the maximum possible score a study could achieve was 24. Each study's score was then expressed as a percentage, and the corresponding percentage equivalency (Table 4) assigned.

Table 4. Percentage equivalency and grade descriptors. A grade more than 3 was interpreted as an indication that the study in question would provide insufficient information towards answering the outlined research question.

Percentage Grade Descriptor		
91 - 100%	1	Highly relevant
81 - 90%	2	Relevant
66 - 80%	3	Moderately relevant
50 - 65%	4	Marginally relevant
0 - 49%	5	Irrelevant
Accept studies ≤ 3 Exclude studies > 3		

A scale of 1 – 5 was introduced, 1 being the highest possible score and 5 being the lowest possible score. Studies were selected in a three-stage process. In the first stage, initial scores were allocated. In the second stage, excluded studies were double-checked. Last, accepted studies were also double-checked. We note, no scores were changed during the double-checking.

A.4 Overview of selected studies

Our literature search identified 136 studies. In evaluating the identified studies against the selection criteria, 47 studies were excluded. During verification, 63 studies were excluded. In summary, 26 published and peer-reviewed papers were selected. We organized the selected studies using Mendeley, which simplified ordered caching of bibliographies. Table 5 lists the publication sources and Table 6 provides a summary of studies selected during the systematic literature review.

Table 5. Literature sources.

Publication	Type	Year
Proceedings of the ACM on Interactive, Mobile, Wearable and Ubiquitous Technologies	Journal	2020
CHI Conference on Human Factors in Computing Systems	Conference	2019
Proceedings of the 31st Australian Conference on Human-Computer-Interaction	Conference	2019
Proceedings of the 32nd Annual ACM Symposium on User Interface Software and Technology	Conference	2019
Proceedings of the CHI Conference on Human Factors in Computing Systems	Conference	2019
Proceedings of the 31st Annual ACM Symposium on User Interface Software and Technology	Conference	2018
Proceedings of the ACM International Conference on Interactive Surfaces and Spaces	Conference	2018
Proceedings of the ACM on Human-Computer Interaction	Journal	2018
Proceedings of the ACM International Conference on Interactive Surfaces and Spaces	Conference	2017
Conference on Human Factors in Computing Systems	Conference	2014
Proceedings of the 9th ACM International Conference on Interactive Tabletops and Surfaces	Conference	2014
Proceedings of the SIGCHI Conference on Human Factors in Computing Systems	Conference	2013
Proceedings of the 25th Annual ACM Symposium on User Interface Software and Technology	Conference	2012
Proceedings of the 24th Annual ACM Symposium on User Interface Software and Technology	Conference	2011
Proceedings of the 23rd ACM Symposium on User Interface Software and Technology	Conference	2010
Proceedings of the SIGCHI Conference on Human Factors in Computing Systems	Conference	2012
Proceedings of the 5th Nordic Conference on Human-Computer Interaction	Conference	2008
Proceedings of the 15th IEEE International Workshops on Enabling Technologies	Conference	2006
Proceedings of the Annual ACM Symposium on User Interface Software and Technology	Conference	2005
Proceedings of the 6th International Workshop on Gesture in Human Computer Interaction and Simulation	Conference	2005
ACM Transactions on Computer-Human Interaction	Conference	2001
IEEE and ACM International Symposium on Augmented Reality	Conference	2000
Conference on Human Factors in Computing Systems	Conference	1999
Proceedings of the 10th annual ACM Symposium on User Interface Software and Technology	Conference	1998

Table 6. Reviewed literature.

Study	Score	Focus	Approach	Validation
Off-surface tangibles: Exploring the design space of midair tangible interaction Cherek et al.	2	Midair interaction above interactive surfaces	Exploring input modalities	Demonstration
HapTable: An interactive tabletop providing online haptic feedback for touch gestures Emgin et al.	2	Haptic feedback for touch interaction on display surfaces	Design and implementation of a multi-modal digital interactive display	Usability study
Occlusion-aware hand posture based interaction on tabletop projector Fujinawa et al.	3	Finger occlusion on interacting with projected images	Depth sensor-based evaluation of hand posture	Demonstration
Ray-casting based interaction using an extended pull-out gesture for interactive tabletops Fujita et al.	2	Exploration for a pull-out gesture	Ray-casting based interaction	Demonstration
Accurate and low-latency sensing of touch contact on any surface with finger-worn IMU sensor Gu et al.	1	Optimization of wearable finger touch sensing on surfaces	Evaluation of response rates based on different finger modalities	Evaluation of recall rates of contact sensors
An evaluation of touch input at the edge of a table Joshi and Vogel	1	Surface-edge interaction for occluded tabletops	Exploration of edge tailored gestures	Usability study
MagicPAPER: Tabletop interactive projection device based on tangible interaction Wu et al.	2	Interactive tabletop projection	Development of a multipurpose sensor	Demonstration
Recognizing unintentional touch on interactive tabletop Xu et al.	2	Robust touch detection for surface interaction	Study of user behavioral patterns on tabletop surfaces	Comparative
Surfacestreams: A content-agnostic streaming toolkit for interactive surfaces Echtler	1	Communicating interactive surface information	Open source development of a information transmission pipeline	Demonstration
The TUIO 2.0 protocol: An abstraction framework for tangible interactive surfaces Kaltenbrunner and Echtler	1	Formal description for tabletop semantics	Definition of properties for common controller objects	—
Prodesk: An interactive ubiquitous desktop surface Wingert et al.	1	Tabletop surface interaction	Extending WIMP functionality and surface user interface concepts	Usability study
The interactive dining table, or pass the weather widget, please Echtler and Wimmer	2	Augmenting tabletops with projection images	Integrating low-cost projector-camera systems into everyday environments	Living lab report
UbiBeam: An interactive projector-camera system for domestic deployment Gugenheimer et al.	1	Qualitative study on the practicality of projector-camera systems	Study of interaction modalities	Usability study
Pervasive information through constant personal projection: The ambient mobile pervasive display (AMP-D) Winkler et al.	1	Evaluation of wearable projector system	Study of continuous interaction space controlled using hand gestures	Demonstration of interaction metaphors and use cases

Continued on next page

Table 6 – continued from previous page

Study	Score	Focus	Approach	Validation
WorldKit: Rapid and easy creation of Ad-hoc interactive applications on everyday surfaces Xiao et al.	3	On-the-fly instantiation of interactive surfaces in everyday environments	Abstracting and extending existing projector-camera system implementations	Demonstration of use case scenarios
Extended multitouch: Recovering touch posture and differentiating users using a depth camera Murugappan et al.	1	Multi-user tabletop touch interactions	Study of multi-user input modalities	Usability study
Steerable augmented reality with the Beamatron Wilson et al.	1	Steerable surface projections	Depth camera-based interactions	Demonstration of multi-view projection
OmniTouch: Wearable multi-touch interaction everywhere Harrison et al.	1	Pervasive interaction on surfaces in everyday environments	Wearable depth sensing	Usability study
Combining multiple depth cameras and projectors for interactions on, above, and between surfaces Wilson and Benko	1	Omni-directional interaction with surfaces	Processing multiple depth sensor view perspectives	In the wild usability study
A multitouch software architecture Echtler and Klinker	2	Layered software architecture for surface interaction interfaces	Systematic review of existing interaction interfaces	Case study
The reacTable*: A Collaborative Musical Instrument Kaltenbrunner et al.	2	Tabletop tangible user interface	Fiducial marker vision engine	Case study
TUIO: A protocol for table-top tangible user interfaces Kaltenbrunner et al.	1	Formal description for tabletop semantics	Definition of properties for common controller objects	—
PlayAnywhere: A Compact Interactive Tabletop Projection-Vision System Wilson	1	Mobile front-projection for surfaces in everyday environments	study Surface projection and user interaction sensing in small form factor, mobile devices	Demonstration of interactive surface applications in different environments
Virtual object manipulation on a table-top AR environment Kato et al.	3	Detecting user input for virtual object manipulation	Augmenting interactive surface tangibles with imagery	Usability study
Integrating paper and digital Information on EnhancedDesk: A method for realtime finger tracking on an augmented desk system Koike et al.	2	Design and implementation for tabletop surface	Exploring design strategies for interactive surface implementations	Case study
Augmented surfaces: A spatially continuous work space for hybrid computing environments Rekimoto and Saitoh	1	Design and implementation for spatially continuous interactive surface environments	Exploring continuous input modalities	Case study
The metaDESK: Models and prototypes for tangible user interfaces Ullmer and Ishii	2	Platform for developing interaction techniques	Describing interface abstractions for implementing interactive tabletop surfaces	Case study

B FRAMEWORK EVALUATION: MODELING EXISTING RESEARCH PROTOTYPES

As there is no standard methodology for evaluating model correctness [52, 56], for validation, we employed the proposed framework to describe existing research artifacts. For each modeled prototype, we assessed completeness, i.e., whether or not the framework

- captured all concepts in the hardware, middleware, and software layers unambiguously,
- captured all associations between identified concepts across the separate layers, and
- conveyed all arbitrary specifications.

We formulated model representations for ten prototypes [20–22, 26, 34, 46, 49, 54, 69, 72]. For each prototype, possible logical views for the hardware, middleware, and software layers were modeled based on technical descriptions provided in corresponding publications. As discussed in section 3, the Artefact framework facilitates modeling specifications, i.e., meta-modeling. The framework also enables instantiating concrete software interfaces for the software layer, i.e., description of APIs. For [21, 22, 26, 34, 69, 72], all high-level meta-models are captured adequately. However, as APIs are not disclosed, the captured software models only describe possible logical views derived from application descriptions in publications. The Artefact framework also demonstrates utility in underlining non-disclosed hardware specifications, i.e., technical descriptions omitted in corresponding publications. Four of the ten models were cross-verified by the original developers of the prototypes [20, 46, 49, 54].

B.1 Digital tabletop with head-mount gaze and eye-tracking

(a) Hardware model.

(b) Software model.

Fig. 11. Modeling the digital tabletop prototype. The software model shows a possible logical view derived from the application described by Xu et al. in [72].

B.2 Digital tabletop (Panasonic TH-50PH12) and motion-track system

(a) Hardware model.

(b) Software model.

Fig. 12. Modeling the digital tabletop prototype. The software model shows a possible logical view derived from the application described by Fujita et al. in [22].

B.3 Surface interaction wearables

(a) Hardware model.

(b) Software model.

Fig. 13. Modeling the surface interaction prototype. The software model shows a possible logical view derived from the application described by Gu et al. in [26].

B.4 Overhead projector-camera system for tabletops

(a) Hardware model.

(b) Software model.

Fig. 14. Modeling the projector-camera system prototype. The software model shows a possible logical view derived from the application described by Fujinawa et al. in [21].

B.5 Projector-camera system with a 10-camera Vicon system for tabletop edges

(a) Hardware model.

(b) Software model.

Fig. 15. Modeling the projector-camera system prototype. The software model shows a possible logical view derived from the application described by Joshi and Vogel in [34].

B.6 Overhead projector-camera system for tabletops

(a) Hardware model.

(b) Software model.

Fig. 16. Modeling the projector-camera system prototype. The software model shows a possible logical view derived from the application described by Wu et al. in [69].

B.7 Back-projected interactive tabletop surface

(a) Hardware model.

(b) Software model.

Fig. 17. Modeling the projector-camera system prototype [54].

B.8 Back-projected interactive wall

Fig. 18. Modeling the projecter-camera system prototype [20].

B.9 Back-projected interactive wall

(a) Hardware model.

(b) Software model.

Fig. 19. Modeling the projector-camera system prototype [46].

B.10 Back-projected interactive wall

(a) Hardware model.

(b) Software model.

Fig. 20. Modeling the projector-camera system prototype [49].

B.11 Back-projected interactive wall

(a) Middleware model.

Fig. 21. Modeling the projector-camera system prototype [49].