Towards Rapid Prototyping of Foldable Graphical User Interfaces with Flecto

IYAD KHADDAM*, Université catholique de Louvain, Belgium
JEAN VANDERDONCKT*, Université catholique de Louvain, Belgium
SALAH DOWAJI, Damascus University, Syria
DONATIEN GROLAUX, ICHEC Brussels Management School, Belgium

Whilst new patents and announcements advertise the technical availability of foldable displays, which are capable to be folded to some extent, there is still a lack of fundamental and applied understanding of how to model, to design, and to prototype graphical user interfaces for these devices before actually implementing them. Without waiting for their off-the-shelf availability and without being tied to any physical foldable mechanism, Flecto defines a model, an associated notation, and a supporting software for prototyping graphical user interfaces running on foldable displays, such as foldable smartphone or assemblies of foldable surfaces. For this purpose, we use an extended notation of the Yoshizawa-Randlett diagramming system, used to describe the folds of origami models, to characterize a foldable display and define possible interactive actions based on its folding operations. A guiding method for rapidly prototyping foldable user interfaces is devised and supported by Flecto, a design environment where foldable user interfaces are simulated in 3D environment instead of in physical reality. We report on a case study to demonstrate Flecto in action and we gather the feedback from users on Flecto, using Microsoft Product Reaction Cards.

CCS Concepts: • **Human-centered computing → Displays and imagers**; **Interactive systems and tools**; *Graphical user interfaces*; *Virtual reality*; Ubiquitous and mobile computing design and evaluation methods; • **Software and its engineering → Integrated and visual development environments**; **Rapid application development**.

Additional Key Words and Phrases: Foldable Displays, Graphical User Interface, Origami, Rapid prototyping, Simulator, Flecto

ACM Reference Format:
Iyad Khaddam, Jean Vanderdonckt, Salah Dowaji, and Donatien Grolaux. 2020. Towards Rapid Prototyping of Foldable Graphical User Interfaces with Flecto. *Proc. ACM Hum.-Comput. Interact.* 4, ISS, Article 194 (November 2020), 33 pages. https://doi.org/10.1145/3427322

1 INTRODUCTION

Foldable Active Matrix Organic-Light-Emitting-Diode (AMOLED) displays have been built and proved mechanically viable [21]: no visible crease exists at the junction line or surface for folding, extensive folding-unfolding cycles do not significantly affect the physical structure of the display, the interaction surface is preserved, active display matrices are of high-quality.

Authors' addresses: Iyad Khaddam, Université catholique de Louvain, Place des Doyens, 1, Louvain-la-Neuve, 1348, Belgium, iyad.khaddam@uclouvain.be; Jean Vanderdonckt, Université catholique de Louvain, Place des Doyens, 1, Louvain-la-Neuve, 1348, Belgium, jean.vanderdonckt@uclouvain.be; Salah Dowaji, Damascus University, Damascus, , Syria, sdowaji@gmail.com; Donatien Grolaux, ICHEC Brussels Management School, Boulevard Brand Whitlock, 4, Brussels, 1150, Belgium, donatien.grolaux@ichec.be.

2573-0142/2020/11-ART194 $15.00
https://doi.org/10.1145/3427322

The AMOLED technology opened new horizons to screens and mobile device vendors. One interesting feature is their ability to increase the size of the display, while reducing the size of the device. This technology is gaining more and more attention from manufacturers, who have already reached the market with foldable devices, introduced concept foldable devices, filed patents on foldable devices, or are on their way to ride this new technology wave. Royole FlexPai[1], Samsung Galaxy Fold[2], Huawei Mate X[3] and Escobar Fold 1 are examples of existing foldable devices in the market. Motorola Razr V4[4], Microsoft 'Surface Duo' [5], TCL concept foldable smartphone[6], Xiaomi foldable concept device[7] are examples of concept devices. Other big manufacturers already filed patents like Apple [41], LG [24], Samsung [19] and Sony [11].

Foldable screens are more than just dual screens and high resolution. There are foldable device prototypes in research with advanced and innovative interaction capabilities that are not implemented in any foldable products yet. For instance, some research prototypes define multiple folding hinges (compared to one hinge in foldable products). Others define innovative interaction techniques such as leafing [38], bending to zoom in and out [2] among others. With over 20 foldable display prototypes [34], a new era of foldable displays is introducing early prototypes and exploring creative interaction techniques. Some day, these prototypes will find their way into the industry. We will be able to buy a foldable device to read books while using the leafing gesture in a natural way, another for a classy restaurant menu, a third for playing games. Each foldable widget will serve its function with a dedicated software, at least the User Interface (UI) to grasp the device interaction capabilities. Manufacturers of such widgets will be challenged by the tight coupling between the device and the function it is designed for. They need to design a device that fits the function, and at the same time they need to develop the software that adequately serves that function while exploiting the device capabilities. This will couple the design decisions on the hardware device and the software. By then, approaching the development of software applications for foldable devices as dual screens and high resolution will not be enough to address such a challenge.

Manufacturers of foldable widgets may address the coupling between the device hardware and the software by following a try and error approach, passing through several iterations of prototyping the device and the software. The challenge introduced by the mutual dependence between the hardware and the software decisions manifests in several ways. It may introduce further costs for the fabrication of several prototypes over several iterations, especially in the early phases of conception of the product, not to mention the complexity of the process while dealing with high technologies [30, 31]. Besides, being a hardware prototype, it lacks flexibility in the functionality to provide (ability to add or modify functionalities), may suffer from reduced mobility out of the laboratory and complexity in the setup that may involve multiple specialized equipment. Furthermore, on the software part, there is a lack in software development methods, models and tools, specifically tailored for the UI, to address this challenge.

In order to address the above challenges and gaps, we introduce Flecto, a model-based approach to develop software for foldable devices. It allows the widget designer to prototype the foldable device using a software tool, to prototype a software for that device and then run the software on the device in a simulated environment, while enabling interaction with the foldable device. It consists of a set of models for the device and the software, a method that provides systematic guidance to

[1]See https://www.cnet.com/reviews/royole-flexpai-preview/
[2]See https://www.samsung.com/global/galaxy/galaxy-fold/
[3]See https://consumer.huawei.com/en/phones/mate-x/
[4]See https://www.mobielkopen.net/motorola-patent-opvouwbare-telefoon-scharnier
[5]See Seehttps://www.techradar.com/au/news/microsoft-surface-duo-is-the-surface-phone-weve-been-waiting-for
[6]See https://robbreport.com/gear/phones/tcl-tri-fold-phone-2876493/
[7]See https://www.techradar.com/au/news/the-xiaomi-foldable-phone-concept-looks-incredible-in-new-teaser-video

the designer to use these models, and a tool to support the models and the method. Flecto uses the notion of Origami in modeling the foldable device, because it is a powerful notation and it has been largely explored for foldable displays in the literature, such as in [7, 20, 39, 42]. Flecto addresses the challenge of developing UIs for foldable devices by providing a methodological guidance and tools that enhance the communication with stakeholders and provide an early feedback on the widget design and on the requirements for the final product. It enables conducting preliminary tests on the foldable device capabilities and the software running on it. However, Flecto is not concerned with the fabrication of foldable devices, and therefore their fabrication is out of the scope of our research. Besides, as a simulator, it is very hard to use Flecto to conduct usability testings on the foldable widget, where the physical contact is quite important to the user. Therefore usability testing is out of the scope as well. The contributions on this paper are the following:

(1) A model to define foldable devices, their physical surfaces and interaction capabilities.
(2) A model to develop UI prototypes for foldable screens.
(3) A method to use the above models together to evaluate foldable devices prototypes.
(4) A software to support the method and the models, which includes design tools for the above models and a simulator to run the UI on the foldable device in a 3D environment.

Flecto enables researchers and practitioners interested in investigating foldable devices to early prototype their devices and applications, to conduct a preliminary assessment of defined interaction and to produce the specifications of both the foldable device and the software running on it.

1.1 Definitions and Terms

Display surfaces can be classified as observation or action surfaces [5]. In the case of foldable screens: users can interact with the content displayed on the screen either by using direct input or physical folding gestures. Foldable devices are part of the larger domain of Organic User Interfaces OUIs [13], which are based on three design principles: (1) Input equals output (shape deformation is a key form of input) (2) function equals form (the physical shape of the device is chosen to better support the main function of the device) and (3) the form follows the flow, such as in actuated displays that change their shape according to the state of the device (incoming call, reading, typing..). Our definition for foldable devices does not cover actuated displays.

In order to maximise the precision, we clarify the terms used in the text in regard to foldables. A *Foldable Device* is a device with surfaces that can be folded or bended according to defined hinges. We call the surfaces of a foldable device *Foldable Surfaces*. A *Foldable Surface* may have up to two displays, which we call *Foldable Displays*. We call the set of displays of a foldable device a *Foldable Screen*. The application that runs on a foldable device is a *Foldable Application* and its UI is a *Foldable User Interface* (FoUI). We can define a FoUI as a user interface that can be displayed on a non-planar surface and has the ability to adapt itself to physical deformations of the surface. Finally, the *Projection of a UI* is displaying that UI on one or on several displays, independently of the means used. FoUIs have the ability to project themselves on several displays.

This paper is structured as follows: Section 2 introduces related work, existing research prototypes and concludes with a list of requirements for Flecto to support the widest range possible of existing prototypes. Section 3 presents a formal representation of foldable devices, models, and a method to use them, Section 4 describes Flecto, the software supporting the method and the models. Section 5 presents a step by step case study on developing a complete application following the method and using the tool. Section 6 presents the feedback gathered from participants, using Microsoft Product Reaction Cards. The final section 7 is dedicated to the conclusion and future works.

2 RELATED WORK AND BACKGROUND

Folding has been evaluated by researchers in different domains. Carpendale et al. [4] investigated using folding techniques for comparing regions of graph layouts. Dragicevic [6] investigated a *Fold & Drop* as a folding interaction technique, in which flipping between windows is achieved through folding and dropping gestures instead of keyboard keys. However, the first foldable prototype we encountered is PaperWindows [14] which simulates a graphical window projected on a paper, that is tracked by a Vicon Motion Capturing System. the graphical window follows the movements and deformation of the paper. Although innovative, its cost is very high due to the use of specialized equipment. The display size is A4 and is used on a table in a laboratory.

Several foldable prototypes followed. We identify four families based on the way they integrate the FoUI: (1) the projection prototype family: a projector projects the FoUI on the foldable device, (2) the foldable input devices family: the device controls a virtual scene or an application on a computer, but it does not display content on its surfaces, (3) the hardware prototype family: a fabricated device actually runs the FoUI on it, and finally (4) the simulation environment prototypes: the device and the FoUI are running inside a simulator. To identify modeling requirements for Flecto, we employed the following criteria to analyse prototypes.

- *Folding capabilities*: folding capabilities are unique to foldable devices. We need to understand the folding capabilities of prototypes in order to model them.
- *Direct input to the FoUI*: the ability to interact directly with the FoUI contents projected on the displays, such as clicking on a button, scrolling down on a foldable screen.
- *Mobility*: the ability to use the prototype outside laboratory conditions.
- *Displaying surface*: the size of the prototype screen. The display size impacts not only the performance of tasks on the prototype, but also the cost to fabrication.
- *Cost*: the cost of the prototype fabrication in addition to the setup.

2.1 Foldable Projection Prototypes

This family has many prototypes such as PaperWindows, Foldable Interactive Displays [25], Corba [40], FoldMe [18], FlexPad [33], ProjectAgami [35] and Bendy [26]. Probably this family is interesting to researchers because the projection technique decouples the hardware and the software, which provides the flexibility to evolve each separately. Projection of UIs is not limited to planar surfaces of the device. Foldable Interactive Displays projects UIs on complex foldable surfaces, such as scroll, fan, or umbrella and uses infrared LEDs to track movements by a camera.

This family employs different input techniques. PaperWindows uses the Vicon System to track deformation and interactions. FoldMe, FlexPad and ProjectAgami use a similar input technique in addition to depth sensors in a Kinect. FoldMe projects a UI on a surface with predefined hinges and tracks foldings with an Optitrack motion capture system with 6 infrared cameras. FlexPad can project any interactive content on the foldable device and track input by a Kinect camera and a depth sensor. ProjectAgami, a foldable mobile device that leverages simple 2D origami form to give applications new affordances, uses Kinect depth sensor to track clicks and device deformations.

Prototypes in this family suffer from reduced mobility, because they require a projector and sensors. However, some prototypes solved this problem by employing a mobile projector, the case of Corba [40] and Bendy [26], which target mobile gaming experience. Corba consists of a flexible plastic board with bendable corners and sides. Bendy is a real bendable hardware prototype with embedded sensors to detect bending actions as input to the software.

These prototypes share benefits and limitations. They require minimal constraints on the type of material to build. A prototype can be passive (a piece of paper or a plastic) or augmented with emitters (the case of Corba). The projection equipment handles the input recognition, with the

exception of Bendy where the device is equipped with sensors. They enjoy high folding capabilities and it is quite possible to recognize almost all types of bendings and foldings. Projection prototypes enjoy large display surfaces, the coverage area of the projector. Foldable surfaces are double sided displays that can display different content (the projector recognizes flipping movement) or the same content (the projector does not recognize flipping movements). A main shortcoming is the high cost due to the use of sophisticated equipment in the setup. Another limitation concerns direct input. Although sensors can provide direct input, identifying the precise position on the projected content is quite challenging, and turning an existing UI into a FoUI may require modifying the UI to turn its widgets bigger. Identifying a click on a surface that is parallel to the projection axis is an issue. In all cases, we need to define a mapping between the physical events and the UI widgets.

2.2 Foldable Input Device Prototypes

In this family, devices are developed to control a scene, not to display contents. They enjoy folding capabilities to provide a natural and/or enhanced interaction experience. Gallant et al. [8] constructed Flexible Input Device, a device that consists of a paper sheet augmented with infrared reflectors that are tracked by a simple IR camera for detecting the folding gestures. Using that device, the user can control a UI displayed on a standard screen.

Using a foldable device to interact with the software is exploited further by researchers. Foldio [29] enables users to prototype a hardware device using sensors and 3D printing. While it is not a display, it enables defining and recognizing shape changes of the object, as well as defining custom objects and printing them in 3D. The same approach is followed in Tangrami [39], which provides a digital design toolkit in which the shape and the functionalities of a foldable user interface can be specified using building blocks, called Tangramis. Tangramis are designed in a visual editor, subsequently printed on an ink-jet printer, and assembled.

Devices in this family share the same benefit of providing rich interaction techniques. They are not displays in any means, but this is not a concern for Flecto because we can provide a virtual display for them in a simulated world. However, their cost plays an important role in the decision to use them in a project, which varies depending on the capabilities of the device. Flexible Input Device is more sophisticated than Foldio and Tangrami and it is more expensive to fabricate. All of the three devices enjoy rich folding capabilities. Direct input is not a problem as far as sensors are installed in places, but it takes a special form in this family because we do not have a FoUI. Mobility is higher than the projection family. Flexible Input Device can be used on a desk because it is a wired device, while Foldio and Tangrami can be used inside a room as far as the sensors stay in the range.

2.3 Foldable Hardware Prototypes

These prototypes are real devices that embed the software inside, which in turn is developed specifically for the device to consume its capabilities. Bookisheet [38] is designed to browse contents and is intended to replace the classical book while providing a browsing experience borrowing the metaphor of leafing to turn the pages. It consists of two thin plastic sheets and bend sensors. PaperPhone [22] is a smartphone-inspired flexible E-Ink prototype, that is equipped with bend sensors, enabling users to build their own bend gesture vocabulary and map it to specific actions on the flexible display. PaperTabs [36] is a paper computer with multiple touch sensitive flexible displays with the aim to replace desktop documents. The location of the display is tracked on the desk using an electro-magnetic tracker. Touch and bend sensors in each display allow users to navigate content. Flexkit [12] is a customizable platform of PaperTabs by end-users.

PaperFold [9, 10] is a multi-segmented mobile device for triggering viewport transformations on the graphical interface. It enables users to attach, reorient and fold its touch displays.It defines

folds based on attached displays and senses the movements on the hinges.WristOrigami [42] extends the interaction surface of a smartwatch by connecting its sides to foldable touch displays. It has sensors to detect folding. However, considering the very small size of the touch screens, direct input is very limited. SurfaceConstellations [27] adopts a complementary approach in suggesting a designer software tool to assemble displays into a constellation that is already folded. It employs dedicated hinges to attach display devices together that are detectable by the designer to determine the configuration and adapt the software UI accordingly.

Prototypes of this family are easy to set up, their cost stays prohibitive due to the complex technologies used in their conceptions. Their folding and input capabilities are limited by their technology, therefore user interaction is less supported than in the projection family. However they enjoy a better mobility than the projection family. Their displays are usually small to medium, but they often are extensible. A main shortcoming is being limited by the hardware capabilities. Being a physical hardware devices, they are most fit as a proof of concept prototypes.

2.4 Simulated Prototypes

This family does not contain many members. Foldinterface [37] is the only work we found in this family. It consists of a tool to model the device and another to load sketches for UIs to simulate their projection on the device, in a simulated 3D environment. This research is very interesting in the context of this paper, but has shortcomings we aim to overcome in Flecto. It is a tool with no methodological support on how to design for foldable interfaces. Besides, it does not provide the means to run an executable UI on the simulated device, which prevents testing capabilities and benefits of migrating a UI into a FoUI. The only possible input comes from the hinges of the device, while direct input is not possible.

This approach removes many barriers from the real world. A device with whatever folding capabilities could be always defined, with direct manipulation. High mobility is paramount and only a computer is needed. The display surface can be extended as we wish and the cost is the lowest because no further equipment or fabrication is involved. However, this virtuality implies a main shortcoming: reproducing the same user experience with a real device is limited. Therefore, until an acceptable virtual haptic experience is developed, the prototypes of this family can serve only as early prototypes to communicate ideas among project members.

2.5 Comparison Summary

Table 1 reviews the four families of existing foldable prototypes by illustrating their differences. The simulated family is the least cost demanding and enjoys the highest mobility. The projection family is pretty expensive but enjoys the best folding capabilities. User interactions by physical folding or direct input are globally well supported. Enlarging the display surface is generally possible and sometimes extendable. Mobility is limited, unless a mobile projector is used.

The input device family defines the widest range of physical interactions enabling a good user experience. While their fabrication is becoming less expensive with 3D printers, they are promising to test prototypes in their late conception phases. However, these prototypes have no screens and direct manipulation is not supported.

The hardware prototypes are the most expensive. They suffer from rigidity to modify their capabilities after fabrication, which is constraining for prototyping that requires flexibility. However, these prototypes are too close to the final product and are a better fit for testing the production feasibility and conducting the final tests including usability.

While folding capabilities vary from limited predefined bending and folding to customized folding vocabulary, Flecto should support all of these capabilities to be able to simulate any of them. It shall provide the means to define custom physical movements and events on the device coming

	Family	Folding capabilities	Direct input	Mobility	Displaying surface	Cost
PaperWindows [14]	Projection	bending borders	click, flip, rotate	desk	medium - A4	high
Foldable Interactive Display [25]	Projection	bending	clicks	desk	projection area, extens., irregular	high
Fold me [18]	Projection	folding	clicks	room	medium, extens.	high
Flexpad [33]	Projection	bending	clicks	room	fixed	med.
ProjectAgami [35]	Projection	folding	clicks	desk	large to small	high
Cobra [40]	Projection	bending borders	no	anywhere	fixed, medium	high
Bendy [26]	Projection	bending	no	desk	fixed, medium	high
Flexible Input [8]	Input Device	bending corners	clicks	desk	small, irregular	high
Tangrami [39]	Input Device	foldable, custom.	clicks	room	-	med.
Foldio [29]	Input Device	foldable, custom.	clicks	room	-	med.
BookiSheet [38]	Hardware	bending	clicks	desk	medium	high
PaperPhone [22]	Hardware	bending corners	clicks	desk	small, non extensible, irregular	high
PaperTabs [36]	Hardware	bending	click, touch	desk	med., extens.	high
Flexkit [12]	Hardware	bendin	click, touch	desk	medium	med.
PaperFold [9, 10]	Hardware	fold, attach, reorient	click, touch	desk	small to med.	high
WristOrigami [42]	Hardware	folding	clicks	everywhere	small, extensible	high
Surface Constellations [27]	Hardware	static folding setup	used devices	desk	med. to large	med.
Foldinterface [37]	Simulated	folding	no	everywhere	A4	low

Table 1. Comparison of different foldable prototypes.

from hinges and folding movements, which raises questions on the notation to use for representing folds. We decided to use the notion of Origami in modeling the foldable device, because it is a powerful notation and it has been largely explored in the literature, such as in [7, 20, 39, 42]. We will use the extended notation from the Yoshizawa-Randlett origami notation system [23] that is proposed in Foldinterface because it extends the folding notation to bending.

2.6 The Extended Yoshizawa-Randlett Notation System

The original Yoshizawa-Randlett notation system consists of (Figure 1): (1) a *valley fold* represented by a dashed line with an arrow pointing the direction of the fold, (2) a *mountain fold* represented by a dashed and dotted line and an empty arrow pointing the direction of the fold, (3) a *valley/mountain fold and unfold* represented by an arrow that combines the valley/mountain fold and unfold movements together, (4) a *pleat fold* which is a fold-unfold on a corner, (5) a *turn over* represented by a plain arrow making a spiral, while the invisible lines behind the model are represented with a dotted line and finally (6) a *rotate* represented by the plain arrows up to a specified angle.

The extended notation consists of (Figure 2): (1) a *mountain bending* represented by a series of plain arrows pointing upwards and aligned to draw an imaginary line, (2) a *valley bending*

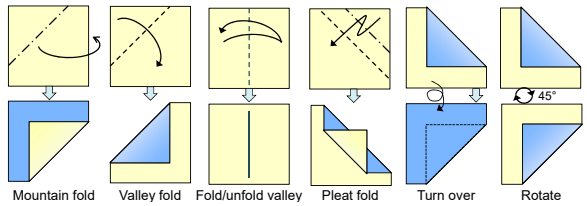

Fig. 1. The original Yoshizawa-Randlett notation [23].

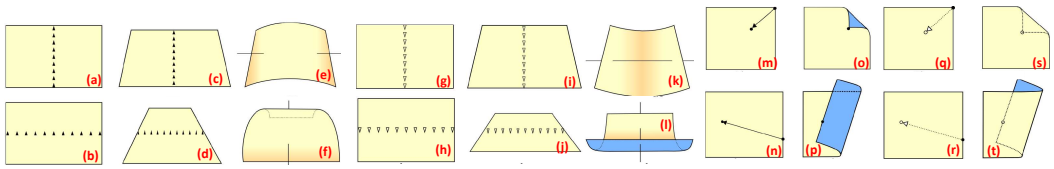

Fig. 2. The extended notation for foldable GUIs [37].

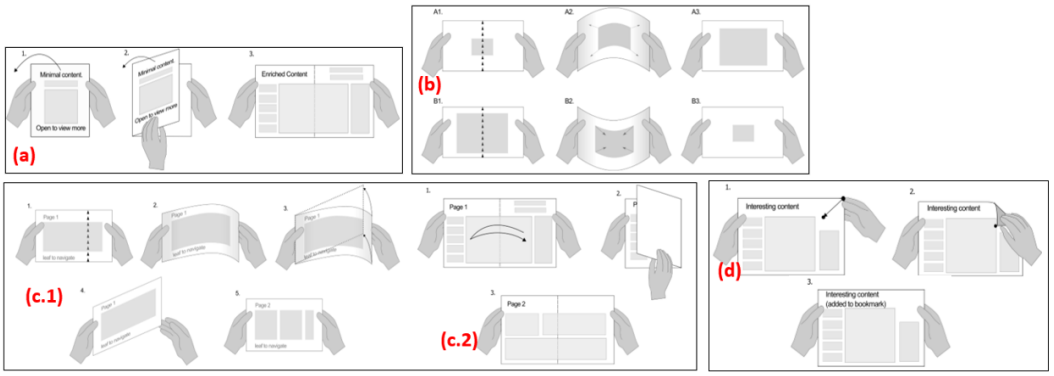

Fig. 3. A catalog of user interactions on foldable displays: (a) Content enrichment, (b) Zooming, (c) Content navigation: two styles: a leafing gesture (c.1) and valley fold (c.2), (d) Bookmarking.

represented by a series of empty arrows pointing downwards and aligned to draw an imaginary line, (3) a *valley corner-bending* represented by a plain straight arrow as opposed to a curved plain arrow for valley fold, and the starting and ending points are noted by a plain circle and finally (4) a *mountain corner-bending* represented by a dotted straight empty arrow and the starting point is noted by a plain circle while the ending point is noted by an empty circle, reflecting that the end position is behind/in front of the model.

2.7 A Catalog of User Interactions

Based on our literature review, we draw a non-exhaustive catalog of possible user interactions with foldable displays. We use the extended Yoshizawa-Randlett to represent them.

(1) **Content enrichment:** one major advantage of having foldable displays is that the screen surface can be easily extended by unfolding. As the display surface increases, the content needs to be adapted to take full advantage of the new configuration. The ability of foldable screens to extend their display surface provides a feature defined here as content enrichment. The example in Figure 3-a depicts a folded device that displays a minimal version of contents (e.g., abstract, minimal version of a web page) which can be enriched by opening the screen.
(2) **Zooming:** as shown with FlexView [2], the valley and mountain bending could be a nice alternative for zooming in and out on foldable screens (Figure 3-b).

(3) **Contents navigation:** unlike conventional displays, foldable devices provide natural ways to navigate content. For example, on a diptych screen the valley fold-unfold gesture could be used for navigation (Figure 3-c.2) and leafing for browsing contents (Figure 3-c.1).
(4) **Bookmarking:** another example of interesting user interaction is bookmarking using a corner valley bending. Indeed this gesture is assimilated to bookmark a page of a real book by folding its corner. In the context of web browsers it represents a natural way of adding a page to the bookmark list.

2.8 Flecto Requirements

Flecto shall simulate the widest list of prototypes we have reviewed in this section, with the premise to be able to simulate future ones. Therefore, we elicit the following requirements:

(1) Provide the appropriate software means to model the physical structure of the foldable device. Flecto shall allow to define displays of the device and its surfaces.
(2) Define the possible physical movements on the foldable device, by employing a notation.
(3) Define interaction capabilities possible for the user.
(4) Provide a 3D simulator for the device.

The 3D simulator shall satisfy the following requirements:

(1) Render the physical device in a 3D environment.
(2) Simulate the physical movements on the device.
(3) Project the content of the UI on the foldable device screen.
(4) Simulate physical events on the device and trigger corresponding UI events.
(5) Support direct input to interact with the projected UI content as if on a planar one.

3 A FORMAL REPRESENTATION OF FOLDABLE DEVICES AND THE FLECTO METHOD

We present the models for simulating a foldable device using UML: a model for the device surfaces, a model of the user interactions, and a model for mapping the physical device structure and events with the application. We present a running example on using these models together.

3.1 The Physical Surface Model

Modeling the surfaces of the foldable device is essential for rendering the device in a 3D world. While many characteristics can describe the physical surface, Flecto requires a sufficient subset to render the device in 3D and to simulate interactions. From this perspective, we define the foldable device model using Unified Modelling Language (Figure 4).

A foldable device has one or more surfaces. A surface may have different attributes (regidity, flexibility, maleability...) if necessary. It has up to two displays, one on each side. Displays carry no pertinent information to Flecto, apart from their existence or not. Therefore, we model the existence of a display on a surface by the boolean attributes *frontDisplay* and *backDisplay* in the class Surface . Besides, a display might be switched on or off based on the folding state. Therefore, we introduced the attributes *frontDisplayState* and *backDisplayState* which can have one of two values: *visible* or *hidden*.

Two surfaces can be linked together by a Fold. A fold can be a MountainFold, a ValleyFold, a MountainBend or a ValleyBend. A fold can be opened or closed by a specific angle, and in both cases, we need to know which surface is moving. Thanks to folds, we can determine the state of the device and the placement of its surfaces at every moment.

An alternative modeling approach of surfaces is applied in Aspecta [32], a simulator for full coverage displays. ASPECTA enables the user to model the surfaces of a room by projecting a

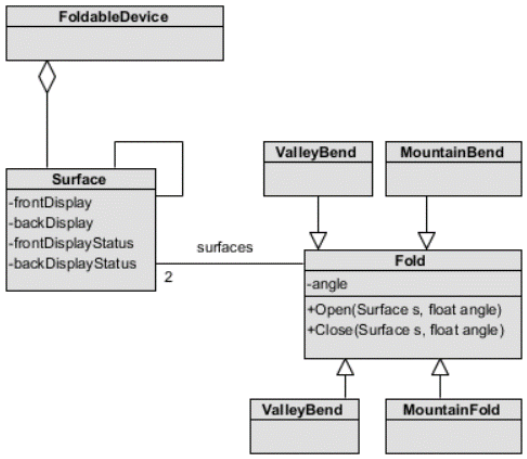

Fig. 4. The foldable device model.

mesh on the wall and allowing the user to increase the number of points in the mesh and to control them to mimic the wall surface. However, the ASPECTA approach is very sophisticated to use in modeling foldable devices because the surfaces are plain ones. The physical model allows identifying available displays for content projection, in addition to possible deformations of the device. At some point, the physical interaction will be mapped with the FoUI. Therefore, it is quite important to define the physical structure to define the possible user interactions.

3.2 The User Interaction Model

The interaction in a classical graphical UI is defined using input and output widgets. Foldable screens introduce a new type of interaction: the physical deformation of the screen by folding, turning and rotating the device, which is covered in our user interaction model. The direct input should be addressed when developing the UI of the software, following well-known approaches for developing UIs, such as model-driven approaches [3, 15–17] among others. This separation between the direct input model and the physical interaction is fruitful to allow developing the UI independently from Flecto models, and at the same time being able to turn it into a FoUI. The user interaction model aims at defining when a physical action on the device (i.e., opening a fold) becomes a meaningful interaction to the system (i.e., enrich contents). We model interactions in the form: the user opens a fold, when the angle of the fold reaches 45 degrees, trigger the action "enrich the content" on the display.

Figure 5 depicts the UML model for Event-Condition-Action (ECA) rules inside the rectangle. An ECA Rule consists of an Event that represents a physical event generated from the device, a Condition that is a logical expression applied on the events above to select only meaningful ones, an Action which is a code in the foldable device application that responds to the triggered event. It can carry out certain functionality in the application including adapting the UI according to the event (like adapting to the display size). Besides, a physical device can produce 4 events: an OpenFoldEvent and a CloseFoldEvent, both have information on the opened/closed surface and the angle, a FlipEvent which has the angle and the flipping direction (Vertical/Horizontal) and a RotateEvent with the rotation angle.

3.3 Mapping the software and the foldable device

Flecto aims at enabling users to evaluate an early prototype of a FoUI on a simulated device. We expect four use cases:

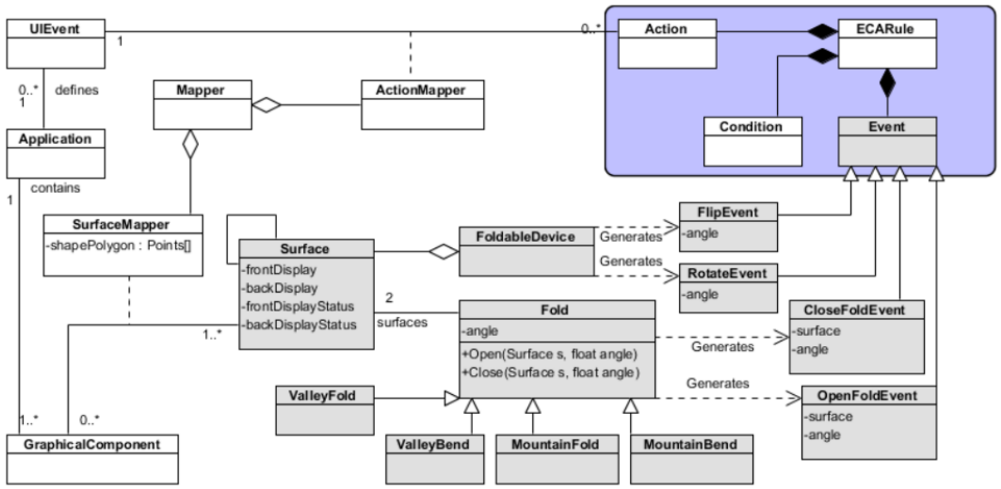

Fig. 5. The UML Class diagram. In grey: the physical device model and the physical events. In white: the mapping model with the application. Inside the rectangle: the ECA Rules.

- Prototype the device: the user has a software/prototype and wants to prototype the device to run that software UI.
- Prototype the software: the user has the foldable device and wants to prototype the software.
- Prototype none: the user has both and wants to evaluate the software on the foldable device.
- Prototype both: the user is exploring the domain of foldable devices and FoUIs.

Flecto supports the first use case by projecting FoUIs on different displays of the device. It enables mapping the FoUI widgets to the foldable displays.

We can support the second use case by providing the means to prototype the software. Another solution is to provide the means to integrate existing prototypes in the tool, which is the way followed by Flecto. A UI prototype could be a simple static/sketch prototype or an executable one. Flecto supports sketches by mapping them to the foldable displays directly, based on the state of folding of the device. This way, the device controls the flow of sketches and defines the navigation model. In the case of an executable prototype, it is the prototype that defines the navigation model. Therefore, Flecto shall transfer physical events to the prototype mapping model, which handles the navigation and decides what to project on each foldable display.

The other use cases mix the previous ones. If no prototype is available, we have all the freedom to develop both. If we have a prototype for the device and the software, we are confronted to the constraints of use cases one and two together and we can benefit from Flecto for both.

In order to integrate the widest range of software prototypes within Flecto, we need a mapping model and a well-defined integration method. The model and the method should not be so restrictive on the way to design and develop the prototype itself. Flecto requires two types of information that should be provided by a prototype: (1) existing widgets in the software prototype and how to distribute them on the foldable display, (2) identify the user inputs in the software prototype to map to physical events generated by the foldable device (using ECAs). Figure 5, in white, illustrates the UML class diagram for the mapping model and relation with the other models.

3.3.1 Modeling the software prototype. The prototype is defined in the mapping model using the Application entity, which consists of:

- GraphicalComponents: the graphical widgets used to design the UI. they can have a hierarchy (like in Form-Panels-Widgets). Flecto requires information on the distribution of graphical

components on the foldable surfaces, not on the complete structure of the graphical components. Therefore include only the top-level components that will be projected. A graphical component can be displayed on different foldable displays, and a foldable display may display different graphical components.
- UIEvents: the UI user inputs defined by UI widgets. UI events in an application are numerous. FLECTO requires information only on the set of UI events that will be triggered by ECA rules. Therefore we include this set only in the model and we ignore the others.

3.3.2 Mapping the software prototype and the physical model. Mapping is achieved using a MAPPER. There are two types of mappers: Surface Mapper and Action Mapper. A SurfaceMapper maps graphical components and surfaces, while an ActionMapper maps UI events and physical events.

When mapping a graphical component to a surface, we need to specify the display on the surface (front or back) to project on. A graphical component can be projected on two displays, which is the case of contents enrichment where the UI form is projected on two adjacent displays. Therefore, we need to specify the part of the graphical component to project on each display. The attribute shapePolygon in is added in the SurfaceMapper class for this purpose. In general, the distribution of graphical components on different surfaces can take one of two types: (a) *Vertical distribution*: to project a component on several foldable displays, we decompose the graphical component into its children components and then project each component on a display and (b) *Horizontal distribution*: when the vertical distribution is not suitable, we slice the graphical component into several parts defined in polygons and map each polygon to a foldable display.

3.4 A Running Example

We apply the models on the content enrichment example presented in Figure 3-a. The application scenario is: Assume we have a planar-screen 2D UI that consists of two UI windows: the first displays the minimum version of content and the second window displays the enriched content. In the initial state, the UI displays the minimal content. When the button Read More is pressed, the enriched content appears. On the enriched-content screen, we may press the Back button to move back to the previous window. Assume we have a prototype for a foldable device that consists of two surfaces connected with a hinge. The first surface has a display on each side while the other has a display only on the front side. Our objective is to render the UI on the foldable device as follows: In its initial state, the foldable device is closed and the minimal content appears on the front display of the left surface. When opening the fold at an angle higher than 45, the enriched content is displayed on the internal displays. When closing the fold at an angle lower than 45, the enriched content disappears (Figure 6).

We define two surfaces for the device: A and B. The attributes frontDisplay and backDisplay of the surface A have the value "true". The attribute frontDisplay of the surface B has the value "true". We define a ValleyFold between these two surfaces. This fold generates two events OpenFoldEvent and CloseFoldEvent that will contribute in the ECA rules. We define an ECA rule as follows: When opening the fold (event) by an angle higher than 45 degrees (the condition), display the enriched content (the action). We also define a second ECA rule as: When closing the fold (event) by an angle lower than 45 degrees (the condition), display the minimal content (the action). We define the Prototype Application as follows:

- Define two GraphicalComponent objects screen1 and screen2 for the minimal and the enriched content windows consequently. Ignore children widgets because the projection happens on the window level.
- Define two UIEvent objects, event1 and event2 for the two push buttons Read More and Back respectively.

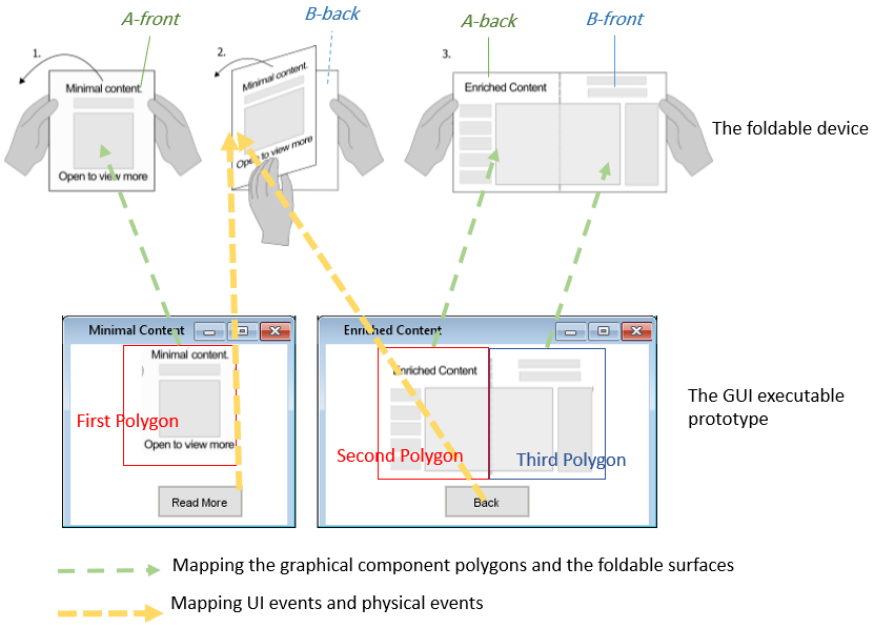

Fig. 6. An example on using the models.

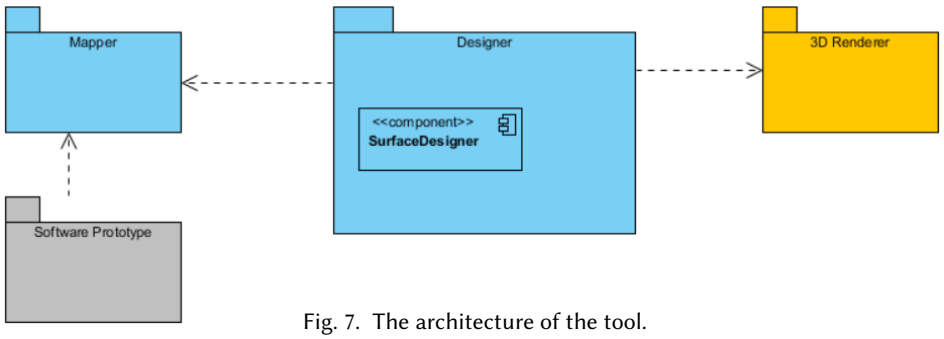

Fig. 7. The architecture of the tool.

We map graphical components and displays as follows: project the complete screen1 on the display A-front, project the left half of screen2 on the display A-back and project the right half of screen2 on the display B-front. We map UI events and physical events as follows: map the event OpenFoldEvent with the UIEvent event1 and map the event CloseFoldEvent with the UIEvent event2.

3.5 The Flecto method

The running example demonstrates the huge amount of data to manage in the models which could be difficult for users. This illustrates the need for a step-wise method to guide the development of the prototype and to help the user to gather and organize data into artifacts. In order to build the models effectively and efficiently, we define the Flecto method, a step-wise method with well-defined artifacts. Figure 8 depicts the main steps to prototype a foldable device using Flecto and the artifacts produced from each step. An overview of the main steps are listed below, while the step-by-step tutorial on the method is presented in the appendix A to conserve space.

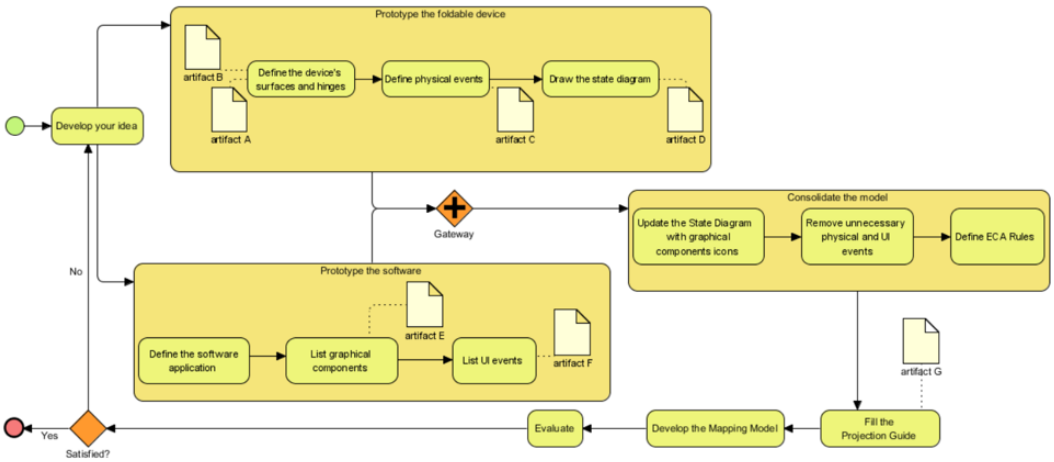

Fig. 8. The Flecto method.

(1) Develop your idea
(2) Prototype the foldable device
  (a) Define the device
  (b) Define physical events
  (c) Draw the state diagram
(3) Prototype the software
  (a) Define the software application
  (b) List graphical components
  (c) List UI events
(4) Consolidate the models
  (a) Update the State Diagram
  (b) Define ECA Rules
  (c) Refine physical and UI events
(5) Fill the Projection Guide
(6) Develop the Mapping Model
(7) Evaluate

4 FLECTO SOFTWARE SUPPORT

Even with the method presented above, prototyping for foldable devices is hard to achieve without a tool to support the method. Herein, we present details on the tool that supports the prototyping of foldable devices. This tool is published as an open source project on the link https://github.com/iyadk/Flecto.

4.1 Architecture

Figure 7 depicts the architecture of the tool. In this figure, we note the following:

(1) *Designer*: the main package in the tool. It contains all the needed classes to define surfaces and folds. This package also contains the graphical designer for physical foldable surfaces.
(2) *Software Prototype*: we support both static/sketches and dynamic/executable UI prototypes. The tool makes direct use of sketches without any additional code. For dynamic prototypes, the developer is invited to implement a java interface to integrate with the tool. The Software Package is grayed in Figure 7 because the UI development is out of the scope of this paper.
(3) *Mapper*: the classes for mapping the prototype and the device for interaction and projection.
(4) *3D Renderer*: is responsible for rendering the models in a 3D virtual environment. It enables physical interactions with the device in addition to direct input to the projected UI.

4.2 Foldable Surfaces and the Designer

A surface is a directed graph, where edges are the surface borders and vertices are the surface vertices. We assume the device cannot have disconnected surfaces, therefore, every surface is connected to at least another with a hinge. On the left side of Figure 9, a device holds two hinges.

On the same figure to the right, we see the representation as a directed graph. The directed graph identifies the surfaces on the device, based on the defined hinges.

Figure 10 illustrates the surface designer. It allows the user to define the hinges on the device's surface using the mouse, in addition to defining ECAs on the hinges. Using the left mouse button draws a valley-fold, while using the right mouse button draws a mountain fold. The designer graphically distinguishes the surfaces by assigning colors to them. We can give surfaces names as well to enhance their usage later. On the left side of the designer relies the settings part to define projection and rendering parameters, mapping with the UI prototype, in addition to defining hinges properties and ECAs. Based on the surface dimensions introduced when creating the surface, we define rendering parameters for the 3D/ 2D worlds by setting the X and Y ratios for each world.

Besides, the designer allows to map the tool with the UI prototype, by providing the jar file for the mapper class. The tool allows running a simulation of the device to test folding capabilities. In fact, it uses a pre-defined mapper to generate a default UI for projection. The simulator can be run by pressing the button Simulate 3D with 2D. When the simulator runs, the generated UI prototype appears on the screen as a set of 2D forms, one for each surface, as illustrated in Figure 11-a. Each

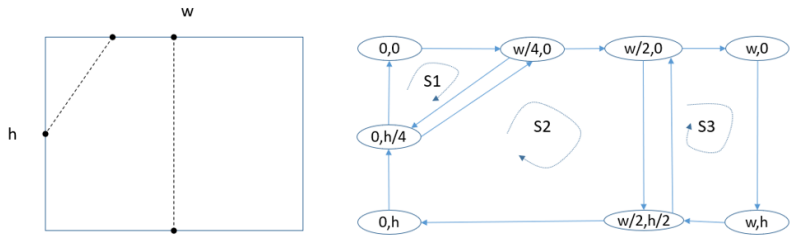

Fig. 9. The directed graph representing the surfaces and the hinges.

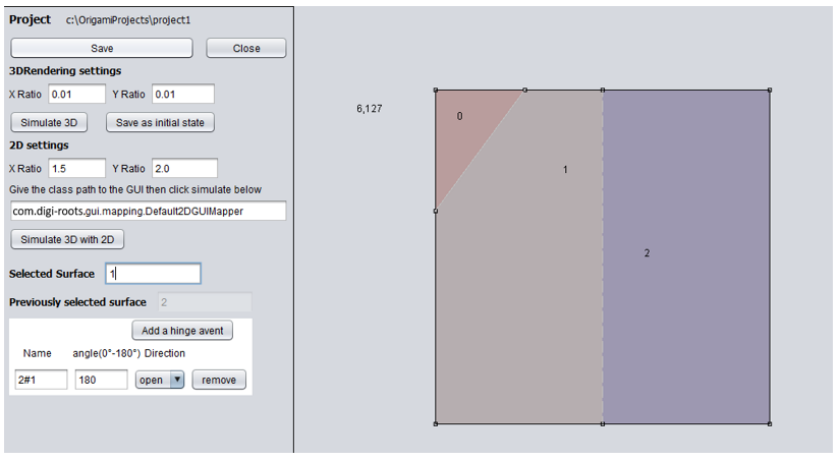

Fig. 10. The surface designer tool.

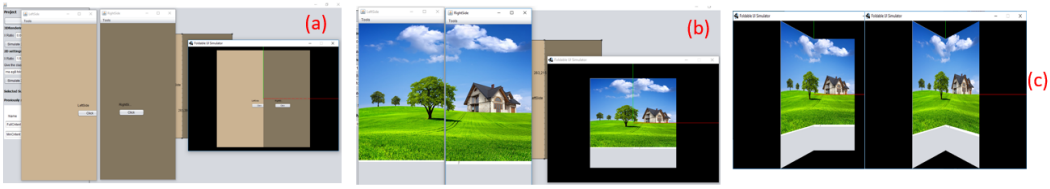

Fig. 11. Running the simulator using (a) the default UI and (b) a static UI prototype (screenshots). (c) Interaction in the 3D world.

form has a menu that allows to display an image on it, as depicted in Figure 11-b. In the virtual reality world, we can interact with the surface to do folding and unfolding moves.

4.3 The 3D Renderer

We evaluated several 3D engines to make the best choice for Flecto. The list includes MigriXML [28], Instant Reality, Unity3D and jMonkey. Flecto uses texturing features extensively, which was a decisive factor in our choice of jMonkey. In fact, these features are either paid or not well supported in the other tools compared to jMonkey. The list of our reasons for choosing jMonkey are: (1) the ability to customise and create the geometrical form to define the surface, (2) the advanced texturing features for free, (3) written in Java, the language we used for the designer and (4) it is an Open source which allows other researchers to contribute to Flecto.

4.4 The Mapper

The mapper is a Java class and it should be created by a developer. It has an instance of the UI prototype. The objective of this class is to implement the Flecto guide. The UI prototype should expose its UI structure and components for this class to be able to map its GraphicalComponents to the foldable surfaces.

5 CASE STUDY: THE SCIENTIFIC ARTICLE BROWSER

In order to evaluate the usefulness of Flecto, we conducted a case study to enact the Flecto Method to develop a fully functional application using the models and tools of Flecto. The interested reader can visit the appendices for a step-by-step tutorial on applying the Flecto method to develop the case study. The tutorial explains how to develop the case study following the method independently of Flecto tools. For conciseness, we present here the outlines of the case study, using the Flecto tools, and discuss the outcomes while referring to figures from the appendix.

5.1 Developing the Scientific Article Browser

The foldable device is used to browse scientific articles to replace the quantity of printed papers, while maintaining an equivalent user experience. The device is compact and can be held in one's pocket. For this purpose, the device is foldable twice in the middle. When fully-opened, upper corners can be folded/unfolded to allow navigation on the content. Its dimensions are an A4 paper.

Figure 15 depicts the designer and the device. Notice the valley folds in light lines, and one mountain fold in a darker line. There are two other valley hinges on the upper corners. These hinges enable folding the device twice, and on the corner, which can be tested in the 3D renderer. The renderer respects all physical constraints among hinges. For instance, when the device is fully closed, it is impossible to fold/unfold corner hinges. On another side, when creating a new device, the designer asks the user to fill the desired dimensions, which we filed to the A4 ones.

Once satisfied with the device design, we moved to defining physical events using ECA rules. The Flecto designer allows the user to define these rules by graphically selecting the hinge (clicking on the forming surfaces) and then setting the angle of the event and the direction. But in its current version, for performance reasons, the Flecto designer supports only ECA conditions with equality, such as 'angle=30', while other types of conditions are not supported yet. Conditions with inequalities allow recognizing continuous events from the device such as zooming in and zooming out. Continuous events trigger the projection process at a high frequency which causes latency and flickering in the renderer. Even though the Flecto model supports all types of conditions, we opted to halt their support in the designer currently. However, we can simulate zooming by defining a sequence of physical events on the hinge while increasing the angle.

Next, we developed the UI prototype illustrated in the figure 17, which is a simple 2D UI to browse an article. It first displays the title and the authors of the article, when the More button is clicked it opens the abstract, the table of content and the references. When the Read Article button is clicked, the first page appears with two navigation button forward and backward. This UI is projected on the device as follows: the initial state of the device is fully folded, which corresponds to the initial state of the GUI. Therefore, we project the article title and authors names on the fully folded device. When the device is half opened, we project the abstract, the table of contents on one side and the references on the other side. When the device is fully opened, the first page is displayed and the user can move forward and backward by folding the upper corners. The outcome of this step is a well defined projection guide that defines what part of a panel on the UI prototype is projected on which surface. This guide is implemented between the UI prototype and the device.

The next step consists of mapping the physical events with the UI events. The physical event of opening the device when fully folded is mapped to the click of the button More. The physical event of opening the half-opened device is mapped to the Read Article button click. The physical events of folding the upper corners are mapped to the Previous page and Next page buttons clicks consecutively. The physical event of folding the fully-opened device is mapped to the Back button click. This mapping is implemented in the interface between the UI prototype and the device.

We used the 3D renderer for evaluation. It enables simulating the device and the software together, as depicted in Figure 19. It allows direct interaction that is helpful when the projected content is too long and the user wants to scroll down to see more by clicking on the projected scroll down, as illustrated in Figure 20.

5.2 Summary

First of all, the development of the complete case study took one working day, with all the discussions between the authors which concluded several iterations and refinements on the models. This is a short time to prototype a foldable device and application. The time to develop the software prototype is not included. Second, as the case with prototypes, Flecto helps to communicate visually on the foldable product and to suggest and evaluate interaction techniques on the fly on the simulator. Flecto served to facilitate the communication among the team members on the specifications of the product, both on the device design and the software application. Third, Flecto helps in identifying requirements for the Foldable Device Application. The projection guide identified how the application's UI architecture could be adapted to fit the projection on the device, as well as to define how to handle the events produced by the device. Forth, with the general idea on the final product established among the team members, we could address other issues like the material of the product in addition to usability-related issues in the final product. Flecto served to reach a quick agreement among the team members on the product functionality and saved the time for more advanced issues.

For all these reasons, we think that Flecto satisfies its objective in providing a preliminary prototype that helps researches develop rapid foldable prototypes.

6 CAPTURING THE USER FEEDBACK

Researchers are potential users of Flecto. In order to get a meaningful feedback on Flecto from users, we conducted an experiment on 13 researchers (1 female) working in the field of the Computer Science and mainly in HCI. The average age of participants was 31 years. The duration of the experiment was 45 minutes. We first introduced the participants to the domain of foldable and flexible displays. We presented a selected set of videos from the research including PaperTabs, PaperFolds, ReFlex and FoldMe. Then we explained the challenge in designing foldable devices. Later, we presented Flecto as an approach to address this challenge. After demonstrating and

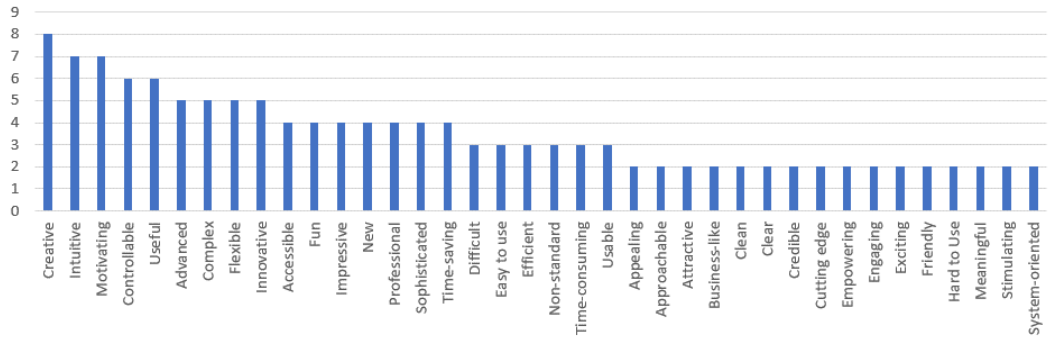

Fig. 12. The frequency of top 38 cards chosen by the participants.

explaining Flecto, each participant freely explored the facilities of Flecto regarding surface design, interaction design and finally rendered their design using the 3D renderer for evaluation during 15 minutes. To preserve the naturalness of this process, no particular target was defined. This session also included a questions and answers discussion between the participants and the experimenter. At the end of the session the participant were asked to fill a questionnaire to express their feedback on the tool, and then we had an open discussion to express their critics and suggestions. We held several sessions, each with 2-4 participants on different days.

The questionnaire consisted of three parts. The first part is dedicated to information on the participant. The second is dedicated to suggestions and comments from the participant. As for the third part we used the Microsoft Product Reaction Cards questionnaire [1]. The Microsoft Product Reaction Cards questionnaire is based on the 118 words used by Microsoft. Each word was presented with a check-box and the participant is asked to choose the words that best describe her evaluation of Flecto. The participant was free to choose as many or as few words as they wished.

The results of our experiment are presented in Figure 12. This figure brings out the words: Creative, Intuitive and Motivating among others. These words reflect the reaction of participants (the researchers) towards the tool, which we interpret as a positive attitude towards the tool and its approach to address the challenge of creating foldable applications as expressed in the introduction.

Another, yet interesting, outcome from our experience is the critics, suggestions and comments from the participants after evaluating Flecto by themselves. By analysing them, we came with the following results. One participant out of the 13 was familiar with the subject of foldable and flexible displays. Others have already heard about products in the market. However, 9 out of the 13 participants suggested applications for foldable devices they are interested in evaluating using Flecto. The table 2 lists the suggested applications by participants. This result gives us a positive sign that Flecto stimulated the participants to think “foldable” and encourages us to work further on enhancing the tool, as they suggested in table 3. However, this is in no means a conclusive evidence on the tool.

Another interesting feedback came from the participant number 6, expressing disagreement to the approach because the tool misses the amusing part of foldable displays, which is the physical movements. We completely agree to this critic, but this aspect concerns the final product, while Flecto addresses the early prototype. Measuring and enhancing the usability of the device is not in the scope of Flecto at this stage. Other two participants (7, 13) expressed their doubts about the ability of the tool to replace a real prototype because it is hard to mimic the user experience of using a real device. We also agree to their critic and we argue that Flecto is not intended to replace the real device at this stage.

Suggested application	Participant
E-Book reader	1,4,7
Newspaper, with augmented content	1,7,11,12
Painting device with pallet and panels	2
Games	3,7
Photo viewer/Editor	4,9
E-Fidelity cards with purchase commands, publicity and shopping cart	5
Radiography navigation apps	9
Restaurant menus	9
Health monitor device on the wrist	10
3D navigator in buildings	12

Table 2. Participants' suggested applications.

Suggestions and comments	Participant
Improve projection ratio	1
Conduct a business study to commercialize	1
Improve the simulator rendering	3
Address culture issues in the device (support RTL languages)	4
Support natural folding	5
Extensibility: support attaching new devices	5
Support the speed of move	9
Intensity of pressing	9
Move the tool to the web to enhance collaboration	11

Table 3. Participants' suggested enhancements.

7 CONCLUSION AND FUTURE WORK

We presented in this article our approach to prototype foldable user interfaces with Flecto. We follow a model-driven approach that consists of a set of models, a method that explains how to use the models and a tool to support the approach. We demonstrated the use of the tool through a detailed case study. The case study illustrated in details how to execute the steps of the method, how to generate the artifacts and how to use the tool. We also conducted an experiment on 13 users to get their feedback using Microsoft Product Reaction Cards.

Flecto provides the appropriate means to model the physical structure of the device through its designer. Furthermore, physical movements and interaction capabilities can be defined on the device and tested using the 3D simulator, which renders the surface in a 3D environment and project the 2D UI prototype on different displays. The simulator produces physical events on the device and triggers actions on the UI prototype. Besides, it enables interaction with the projected content using direct input as if it was displayed on a planar display. For all of the above, we state that Flecto fulfils the requirements stated in section 2.4. Flecto addresses the challenge stated in the introduction by: providing a cheap and a fast way, to preliminary prototype and evaluate foldable widgets, following a methodological model-driven approach.

The use case on the scientific article browser demonstrates in details how to carry out the Flecto method to prototype a foldable device, and how to produce all the artifacts of the method. It also demonstrates how to map the software prototype with the device in the simulator to do the evaluation.

On the other side, the evaluation we conducted in sections 5 and 6 identified shortcomings in the simulator. There is still a shortcoming in the way of opening and closing the hings. The simulator can not give the real feeling of opening and closing a real device, and thus loosing the feedback on the user experience with the device. The evaluation covers only the technical part on the requirements and specifications for the real prototype and the application.

Although Flecto is intended to be used for preliminary prototyping, the need to simulate the user experience with the real device is very appealing to implement in future versions of Flecto. In fact, it is very difficult, if possible, to give the user the real feeling of the device. Theoretically,

we can imagine enhancing the simulator by implementing haptic interaction with the device to simulate the tension when folding the device and to give the touch feeling of the surface.

An alternative approach to enhance the user experience is to couple Flecto with members of the foldable input devices prototypes family. With this coupling, the physical device provides the real feeling to the user while Flecto orchestrates the projection of the UI (in the simulator) and the mapping with the events of the software application. This coupling would complement one's missing interesting characteristics from the other approach. Besides, we can enhance the navigation in the simulator by providing the user with a virtual reality mask. Employing immersive virtual reality techniques could also enhance the user experience in the way to navigate in the simulator.

Finally, suggestions from participants in section 6 are also on our todo list. One interesting feature is to support "Extensibility: support attaching new devices " in the simulator. The other suggestion on moving the tool to the web is interesting because it gives the opportunity to others to participate in the evolution of Flecto.

REFERENCES

[1] Joey Benedek and Trish Miner. 2002. Measuring Desirability: New methods for evaluating desirability in a usability lab setting. In *Proceedings of UPA 2002 Conference* (Orlando, FL, USA).

[2] Jesse Burstyn, Amartya Banerjee, and Roel Vertegaal. 2013. FlexView: An Evaluation of Depth Navigation on Deformable Mobile Devices. In *Proceedings of the 7th International Conference on Tangible, Embedded and Embodied Interaction* (Barcelona, Spain) *(TEI '13)*. ACM, New York, NY, USA, 193–200. https://doi.org/10.1145/2460625.2460655

[3] Gaëlle Calvary, Joëlle Coutaz, David Thevenin, Quentin Limbourg, Laurent Bouillon, and Jean Vanderdonckt. 2003. A Unifying Reference Framework for multi-target user interfaces. *Interacting with Computers* 15, 3 (06 2003), 289–308. https://doi.org/10.1016/S0953-5438(03)00010-9 arXiv:http://oup.prod.sis.lan/iwc/article-pdf/15/3/289/7800242/iwc15-0289.pdf

[4] M.Sheelagh T. Carpendale, David J. Cowperthwaite, F.David Fracchia, and Thomas Shermer. 1995. Graph Folding: Extending Detail and Context Viewing into a Tool for Subgraph Comparisons. In *Proceedings of the Symposium on Graph Drawing*. 127–139.

[5] Joëlle Coutaz, Christophe Lachenal, and Sophie Dupuy-Chessa. 2003. Ontology for Multi-surface Interaction. In *Proceedings of 7th IFIP TC13 International Conference on Human-Computer Interaction* (Zurich, Switzerland, September 1-5, 2003) *(INTERACT'03)*, Matthias Rauterberg, Marino Menozzi, and Janet Wesson (Eds.). IOS Press, Amsterdam, 447–454.

[6] Pierre Dragicevic. 2004. Combining Crossing-based and Paper-based Interaction Paradigms for Dragging and Dropping Between Overlapping Windows. In *Proceedings of the 17th Annual ACM Symposium on User Interface Software and Technology* (Santa Fe, NM, USA) *(UIST '04)*. ACM, New York, NY, USA, 193–196. https://doi.org/10.1145/1029632.1029667 See video at https://www.youtube.com/watch?v=OvC2hKp6G8s.

[7] Alexandra Fuchs, Miriam Sturdee, and Johannes Schöning. 2018. Foldwatch: Using Origami-Inspired Paper Prototypes to Explore the Extension of Output Space in Smartwatches. In *Proceedings of the 10th Nordic Conference on Human-Computer Interaction* (Oslo, Norway) *(NordiCHI '18)*. Association for Computing Machinery, New York, NY, USA, 47–59. https://doi.org/10.1145/3240167.3240173

[8] David T. Gallant, Andrew G. Seniuk, and Roel Vertegaal. 2008. Towards More Paper-like Input: Flexible Input Devices for Foldable Interaction Styles. In *Proceedings of the 21st Annual ACM Symposium on User Interface Software and Technology* (Monterey, CA, USA) *(UIST '08)*. ACM, New York, NY, USA, 283–286. https://doi.org/10.1145/1449715.1449762 See video at https://www.youtube.com/watch?v=e6jJp-AWFFY&feature=emb_logo.

[9] Antonio Gomes and Roel Vertegaal. 2014. PaperFold: A Shape Changing Mobile Device with Multiple Reconfigurable Electrophoretic Magnetic Display Tiles. In *Proceedings of ACM International Conference on Human Factors in Computing Systems, Extended Abstracts* (Toronto, Ontario, Canada) *(CHI EA '14)*. ACM, New York, NY, USA, 535–538. https://doi.org/10.1145/2559206.2574770 See video at https://www.youtube.com/watch?v=IBdFQ6ICD-4.

[10] Antonio Gomes and Roel Vertegaal. 2015. PaperFold: Evaluating Shape Changes for Viewport Transformations in Foldable Thin-Film Display Devices. In *Proceedings of the Ninth International Conference on Tangible, Embedded, and Embodied Interaction* (Stanford, California, USA) *(TEI '15)*. ACM, New York, NY, USA, 153–160. https://doi.org/10.1145/2677199.2680572 See video at https://www.youtube.com/watch?v=IBdFQ6ICD-4.

[11] Yukinori Hidaka, Ken Ootsuka, and Naoto Kinoshita. 2018. DISPLAY DEVICE AND ELECTRONIC APPARATUS. http://www.freepatentsonline.com/y2018/0336851.html

[12] David Holman, Jesse Burstyn, Ryan Brotman, Audrey Younkin, and Roel Vertegaal. 2013. Flexkit: A Rapid Prototyping Platform for Flexible Displays. In *Proceedings of the Adjunct Publication of the 26th Annual ACM Symposium on User Interface Software and Technology* (St. Andrews, Scotland, United Kingdom) *(UIST '13 Adjunct)*. ACM, New York, NY, USA, 17–18. https://doi.org/10.1145/2508468.2514934

[13] David Holman and Roel Vertegaal. 2008. Organic User Interfaces: Designing Computers in Any Way, Shape, or Form. *Commun. ACM* 51, 6 (June 2008), 48–55. https://doi.org/10.1145/1349026.1349037

[14] David Holman, Roel Vertegaal, Mark Altosaar, Nikolaus Troje, and Derek Johns. 2005. Paper Windows: Interaction Techniques for Digital Paper. In *Proceedings of the ACM International Conference on Human Factors in Computing Systems* (Portland, Oregon, USA) *(CHI '05)*. ACM, New York, NY, USA, 591–599. https://doi.org/10.1145/1054972.1055054

[15] Iyad Khaddam, Hanaa Barakat, and Jean Vanderdonckt. 2016. Enactment of User Interface Development Methods in Software Life Cycles. In *13th International Conference on Human Computer Interaction, RoCHI 2016, Iasi, Romania, September 8-9, 2016*, Adrian Iftene and Jean Vanderdonckt (Eds.). Matrix Rom, 26–35.

[16] I. Khaddam, N. Mezhoudi, and J. Vanderdonckt. 2015. Adapt-first: A MDE transformation approach for supporting user interface adaptation. In *2015 2nd World Symposium on Web Applications and Networking (WSWAN)*. 1–9. https://doi.org/10.1109/WSWAN.2015.7209080

[17] Iyad Khaddam, Nesrine Mezhoudi, and Jean Vanderdonckt. 2015. Towards Task-Based Linguistic Modeling for Designing GUIs. In *Proceedings of the 27th Conference on l'Interaction Homme-Machine* (Toulouse, France) *(IHM '15)*. Association for Computing Machinery, New York, NY, USA, Article 17, 10 pages. https://doi.org/10.1145/2820619.2820636

[18] Mohammadreza Khalilbeigi, Roman Lissermann, Wolfgang Kleine, and Jürgen Steimle. 2012. FoldMe: Interacting with Double-sided Foldable Displays. In *Proceedings of the Sixth International Conference on Tangible, Embedded and Embodied Interaction* (Kingston, Ontario, Canada) *(TEI '12)*. ACM, New York, NY, USA, 33–40. https://doi.org/10.1145/2148131.2148142 See video at https://www.youtube.com/watch?v=LZyClk2rslI.

[19] Hoonsik Kim, Young Do Kim, and Suk Choi. 2018. Foldable display. http://www.freepatentsonline.com/9891670.html

[20] Yuichiro Kinoshita, Kentaro Go, Reiji Kozono, and Kohei Kaneko. 2014. Origami Tessellation Display: Interaction Techniques Using Origami-Based Deformable Surfaces. In *CHI '14 Extended Abstracts on Human Factors in Computing Systems* (Toronto, Ontario, Canada) *(CHI EA '14)*. Association for Computing Machinery, New York, NY, USA, 1837–1842. https://doi.org/10.1145/2559206.2581172

[21] Hyuk-Jun Kwon, HongShik Shim, Sunkook Kim, Woong Choi, Youngtea Chun, InSeo Kee, and SangYoon Lee. 2011. Mechanically and optically reliable folding structure with a hyperelastic material for seamless foldable displays. *Applied Physics Letters* 98, 15 (2011), 151904. https://doi.org/10.1063/1.3576906 arXiv:https://doi.org/10.1063/1.3576906

[22] Byron Lahey, Audrey Girouard, Winslow Burleson, and Roel Vertegaal. 2011. PaperPhone: Understanding the Use of Bend Gestures in Mobile Devices with Flexible Electronic Paper Displays. In *Proceedings of the ACM International Conference on Human Factors in Computing Systems* (Vancouver, BC, Canada) *(CHI '11)*. ACM, New York, NY, USA, 1303–1312. https://doi.org/10.1145/1978942.1979136 See video at https://www.youtube.com/watch?v=_69pIorQ59U.

[23] Robert J. Lang. 2011. Origami Diagramming Conventions - Diagramming, Part 1. Web site. Retrieved December 12, 2019 from http://langorigami.com/article/origami-diagramming-conventions/.

[24] Haklim Lee, Hanseok Chae, and Wonseok Joo. 2018. MOBILE TERMINAL. http://www.freepatentsonline.com/y2018/0183911.html

[25] Johnny C. Lee, Scott E. Hudson, and Edward Tse. 2008. Foldable Interactive Displays. In *Proceedings of the 21st Annual ACM Symposium on User Interface Software and Technology* (Monterey, CA, USA) *(UIST '08)*. ACM, New York, NY, USA, 287–290. https://doi.org/10.1145/1449715.1449763

[26] Jessica Lo and Audrey Girouard. 2017. Bendy: Exploring Mobile Gaming with Flexible Devices. In *Proceedings of the Eleventh International Conference on Tangible, Embedded, and Embodied Interaction* (Yokohama, Japan) *(TEI '17)*. ACM, New York, NY, USA, 163–172. https://doi.org/10.1145/3024969.3024970 See video at https://www.youtube.com/watch?v=mgG7E7p8SoY.

[27] Nicolai Marquardt, Frederik Brudy, Can Liu, Ben Bengler, and Christian Holz. 2018. SurfaceConstellations: A Modular Hardware Platform for Ad-Hoc Reconfigurable Cross-Device Workspaces. In *Proceedings of the ACM International Conference on Human Factors in Computing Systems* (Montreal QC, Canada) *(CHI '18)*. ACM, New York, NY, USA, Article 354, 14 pages. https://doi.org/10.1145/3173574.3173928 See video at https://www.youtube.com/watch?v=--Yxc7lD2gg.

[28] José Pascual Molina Massó, Jean Vanderdonckt, and Pascual González López. 2006. Direct Manipulation of User Interfaces for Migration. In *Proceedings of the 11th International Conference on Intelligent User Interfaces* (Sydney, Australia) *(IUI '06)*. ACM, New York, NY, USA, 140–147. https://doi.org/10.1145/1111449.1111483 See video at https://www.youtube.com/watch?v=w-lG9Ho7z54.

[29] Simon Olberding, Sergio Soto Ortega, Klaus Hildebrandt, and Jürgen Steimle. 2015. Foldio: Digital Fabrication of Interactive and Shape-Changing Objects With Foldable Printed Electronics. In *Proceedings of the 28th Annual ACM Symposium on User Interface Software & Technology* (Charlotte, NC, USA) *(UIST '15)*. ACM, New York, NY, USA, 223–232. https://doi.org/10.1145/2807442.2807494 See video at https://www.youtube.com/watch?v=gA17s9MyfiM.

[30] Michael Ortega and Alix Goguey. 2019. BEXHI: A Mechanical Structure for Prototyping Bendable and EXpandable Handheld Interfaces. In *Proceedings of the 2019 ACM International Conference on Interactive Surfaces and Spaces* (Daejeon, Republic of Korea) *(ISS '19)*. Association for Computing Machinery, New York, NY, USA, 269–273. https://doi.org/10.1145/3343055.3359703

[31] Michaël Ortega, Jérôme Maisonnasse, and Laurence Nigay. 2017. EXHI-Bit: A Mechanical Structure for Prototyping EXpandable Handheld Interfaces. In *Proceedings of the 19th International Conference on Human-Computer Interaction with Mobile Devices and Services* (Vienna, Austria) *(MobileHCI '17)*. Association for Computing Machinery, New York, NY, USA, Article 4, 11 pages. https://doi.org/10.1145/3098279.3098533

[32] Julian Petford, Miguel A. Nacenta, Carl Gutwin, Joseph Eremondi, and Cody Ede. 2016. The ASPECTA Toolkit: Affordable Full Coverage Displays. In *Proceedings of the 5th ACM International Symposium on Pervasive Displays* (Oulu, Finland) *(PerDis '16)*. Association for Computing Machinery, New York, NY, USA, 87–105. https://doi.org/10.1145/2914920.2915006

[33] Jürgen Steimle, Andreas Jordt, and Pattie Maes. 2013. Flexpad: Highly Flexible Bending Interactions for Projected Handheld Displays. In *Proceedings of the ACM International Conference on Human Factors in Computing Systems* (Paris, France) *(CHI '13)*. ACM, New York, NY, USA, 237–246. https://doi.org/10.1145/2470654.2470688 See video at Video:https://www.youtube.com/watch?v=AZwUW7HqcQg.

[34] Miriam Sturdee and Jason Alexander. 2018. Analysis and Classification of Shape-Changing Interfaces for Design and Application-Based Research. *ACM Comput. Surv.* 51, 1, Article 2 (Jan. 2018), 32 pages. https://doi.org/10.1145/3143559

[35] Dominique Tan, Maciej Kumorek, Andres A. Garcia, Adam Mooney, and Derek Bekoe. 2015. Projectagami: A Foldable Mobile Device with Shape Interactive Applications. In *Proceedings of the 33rd ACM International Conference on Human Factors in Computing Systems, Extended Abstracts* (Seoul, Republic of Korea) *(CHI EA '15)*. ACM, New York, NY, USA, 1555–1560. https://doi.org/10.1145/2702613.2732801 See video at https://www.youtube.com/watch?v=AhTHDxe_lqI.

[36] Aneesh P. Tarun, Byron Lahey, Audrey Girouard, Winslow Burleson, and Roel Vertegaal. 2011. Snaplet: Using Body Shape to Inform Function in Mobile Flexible Display Devices. In *Proceedings of ACM International Conference on Human Factors in Computing Systems, Extended Abstracts* (Vancouver, BC, Canada) *(CHI EA '11)*. ACM, New York, NY, USA, 329–334. https://doi.org/10.1145/1979742.1979701 See video at https://www.youtube.com/watch?v=ol_uu5pMmq8.

[37] Jean Vanderdonckt, Iyad Khaddam, and Radu-Daniel Vatavu. 2020. The Foldinterface Editor: A Visual Tool for Designing User Interfaces for Foldable Displays. In *Companion Proceedings of the 12th ACM SIGCHI Symposium on Engineering Interactive Computing Systems* (Sophia Antipolis, France) *(EICS '20 Companion)*. Association for Computing Machinery, New York, NY, USA, Article 1, 6 pages. https://doi.org/10.1145/3393672.3398490

[38] Jun-ichiro Watanabe, Arito Mochizuki, and Youichi Horry. 2008. Bookisheet: Bendable Device for Browsing Content Using the Metaphor of Leafing Through the Pages. In *Proceedings of the 10th International Conference on Ubiquitous Computing* (Seoul, Korea) *(UbiComp '08)*. ACM, New York, NY, USA, 360–369. https://doi.org/10.1145/1409635.1409684 See video at http://mobblog.cs.ucl.ac.uk/category/ubicomp2008/.

[39] Michael Wessely, Nadiya Morenko, Jürgen Steimle, and Michael Schmitz. 2018. Interactive Tangrami: Rapid Prototyping with Modular Paper-folded Electronics. In *Proceedings of 31st Annual ACM Symposium on User Interface Software and Technology, Adjunct Proceedings* (Berlin, Germany) *(UIST '18)*. ACM, New York, NY, USA, 143–145. https://doi.org/10.1145/3266037.3271630 See video at https://www.youtube.com/watch?v=A8VCApp_R6M.

[40] Zi Ye and Hammad Khalid. 2010. Cobra: Flexible Displays for Mobilegaming Scenarios. In *Proceedings of the ACM International Conference on Human Factors in Computing Systems, Extended Abstracts* (Atlanta, Georgia, USA) *(CHI EA '10)*. ACM, New York, NY, USA, 4363–4368. https://doi.org/10.1145/1753846.1754154

[41] Zhen Zhang, Victor H. Yin, Cyrus Y. Liu, Paul S. Drzaic, Sungwon Bae, Chun-hao Tung, Kiarash Vakhshouri, Sunggu Kang, and John Z. Zhong. 2017. Electronic Devices With Flexible Displays. http://www.freepatentsonline.com/y2017/0336831.html

[42] Kening Zhu, Morten Fjeld, and Ayça Ünlüer. 2018. WristOrigami: Exploring Foldable Design for Multi-Display Smartwatch. In *Proceedings of the ACM International Conference on Designing Interactive* (Hong Kong, China) *(DIS '18)*. ACM, New York, NY, USA, 1207–1218. https://doi.org/10.1145/3196709.3196713 See video at https://www.youtube.com/watch?v=1_2D79zntIk.

A THE FLECTO METHOD TUTORIAL

A.1 Develop your Idea

Develop your idea on using foldable devices. You may consult some of the references that are already cited in this article and watch videos if possible. This could inspire your imagination and creativity.

A.2 Prototype the Foldable Device

Based on your idea, work on prototyping the device. Define the surfaces and the folds of the device.

This step consists of three sub-steps and produces the artifacts A,B,C and D that are depicted in the annex A.

A.2.1 Define the device. Fill the artifact A: The Surfaces in table 5 and the artifact B: The Hinges in the table 6. Follow the instructions in each artifact.

A.2.2 Define the physical events. After identifying the hinges, define physical events that will be generated from the device. You should clearly state the hinge, the physical event produced and the condition. Fill the artifact C: ECA Rules in table 7 and follow the instructions in it. The action part of an ECA rule will be filed in step 4.b in the last column.

A.2.3 Draw the state diagram. The foldable device's state is based on the folds of its surfaces. In a specific state, the device can have part of its surfaces folded and others not.

The state diagram is used to show how the device changes from one state to another and the physical event that caused this transition. Figure 14 illustrates the form of the state diagram. The transitions should be selected from the artifact C: ECA Rules in table 7. This diagram will be updated in the step 4.a after creating the software prototype. The figure specifies what should be done at this step and what will be added at step 4.a.

A state of a device is defined by the user. A device with two surfaces and one hinge, may have three states: closed, half opened (angle 90) and fully opened (angle 180). The Open physical event changes the state from closed to half opened and from half opened to fully opened. The difference relies in the angle, that turns a physical event into a meaningful interaction.

A.3 Prototype the Software

Independently from the previous step, prototype the software. Your prototype could be static sketches for the screens, or an executable UI prototype. You may start with the static sketches first and change later to the other.

This step consists of three steps and produces the artifacts E and F.

A.3.1 Define the software application. Explain the purpose of the software, and how it works in simple words. If you want to evaluate a running prototype, complete your development.

A.3.2 List graphical components. From your software prototype, list the graphical components. Graphical components are the forms used in developing the prototype. At this stage, list only top-level ones. You will refine this list later, if needed. If you are using static sketches, every sketch is a graphical component.

Fill the artifact E: Graphical Components in table 8 and follow the instructions in it.

A.3.3 List UI events. Now you have your prototype ready, list the UI events. List the buttons, links and other events on the UI and in different forms or graphical components.

Fill the artifact F: UI Events in table 9 and follow the instructions in it.

A.4 Consolidate the Models

Now that you understand well your foldable device and your foldable application, put the models together and remove non-pertinent information to create a consolidated model. Only keep the minimum information required, by removing the events that are not susceptible to be triggered by any physical event. By the end of this step, define ECA rules.

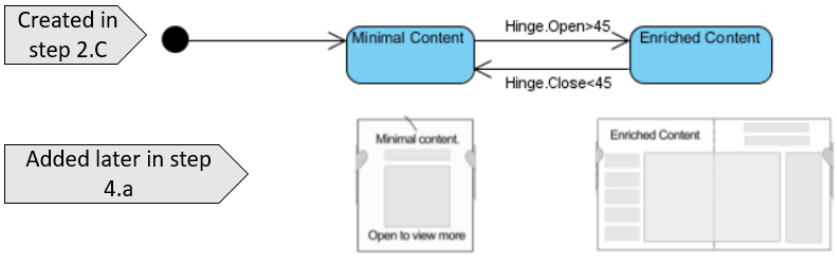

Fig. 13. The State Transition Diagram for the running example.

State	Surface	Graphical Component	Polygon (X0,Y0)	Front face	Back Face
Minimal Content	Left surface	Screen 1	Ploygon starts from (0,0)	off	on
	Right surface	Screen 2	-	off	off
Enriched Content	Left surface	Screen 2	Ploygon starts from (0,0)	on	on
	Right surface	Screen 2	Polygon starts from (W/2,0), W: the width of Screen 2	on	off

Table 4. The Projection Guide for the running example.

A.4.1 Update the State Transition Diagram. It is time to update the state diagram artifact produced in step 2.c. Map every state with the graphical component(s) that will be displayed in that state. This would visually illustrate the flow of the UI when the state changes.

Let us create the state diagram for the running example presented in the previous section. The foldable device has two states: the "minimal content" and the "enriched content". We recommend giving meaningful names to the states to easily recall them, and avoid using technical terms for names like "first fold opened" and/or "first fold closed". It is also a good practice to attach an icon or a screenshot of the graphical components that will be projected on the device on each state. Figure 13 illustrates the updated state diagram of the running example.

A.4.2 Refine physical and UI events. With the help of the state diagram, review the list of physical events. Try to map each physical event with the corresponding UI event. This could help you to add missing events. Remove the events that do not map with any counterpart.

Change the artifacts C and F according to your findings.

A.4.3 Define ECA Rules. Complete the "Action" part of the artifact 2.C with the name of UI event that is mapped to the physical event.

A.5 Fill the Projection Guide

Map the software prototype's screens with device's surfaces and states. If you need to slice some graphical components to fit the device screens, use polygons to define these slices. Define also the state of each display if it is on or off, in each state.

Fill the artifact G: The Projection Guide in table 10 and follow the instructions in it.

If you are using static sketches, you need to slice your sketches to fit into the different displays.

The table 4 is an example on how to fill the Projection Guide in the case of the running example.

A.6 Develop the Mapping Model

The Projection Guide contains all the information to implement the projection part of the mapper. The ECA Rules artifact contains all the information on mapping physical events and the UI events. Develop your mapper class based on these two artifacts.

A.7 Evaluate

Once you have your prototypes ready, evaluate the complete system. Based on the feedback coming from your evaluation, you may decide to re-iterate the process after all.

B THE FLECTO METHOD ARTIFACTS LIST

Surface	Description	Front Display	Back Display
List the surfaces of the device	describe the surface and its position on the device	✓: a front display exists X: a front display does not exist	✓: a back display exists X: a back display does not exist

Table 5. artifact A: The Surfaces.

Hing Surfaces	Friendly Name
List the two surfaces forming the hinge	Give it a friendly name

Table 6. artifact B: The Hinges.

Hinge	Event Description	Physical Event	Condition	Action
Use hinge friendly name	Describe the event and when it is triggered	Open, Close, Rotate or Flip events	The condition on the angle of the hinge. It could take the form: >30, =21, >=45, <60 ...	The UI Event that will be triggered when the physical event satisfies the condition. **This part is filed in the step 4.b**

Table 7. artifact C: ECA Rules.

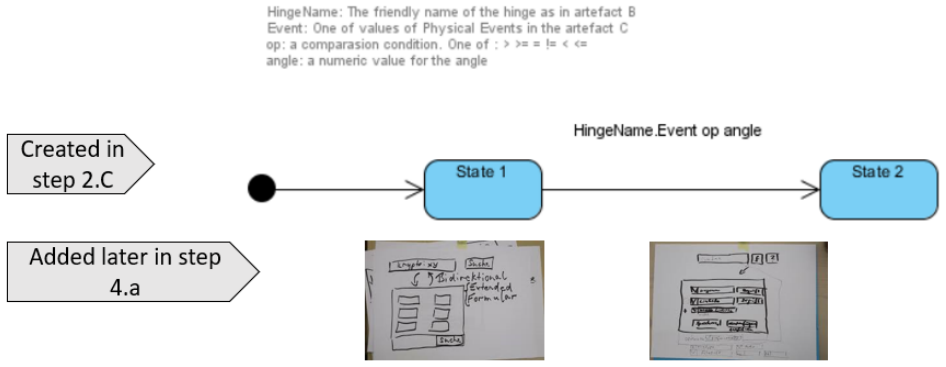

Fig. 14. artifact D: The State Diagram.

Graphical Component	Description
state the name of the graphical component	Describe the component

Table 8. artifact E: The Graphical Components.

UI Event	Graphical Component	Description
List the important UI events generated by the UI prototype	The graphical component that contains the event	Provide a simple description. This list is refined in the step 4.c

Table 9. artifact F: The UI Events.

State	Surface	Graphical Component	Polygon (X0,Y0)	Front face	Back Face
List all the states of the device	For each state, list the surfaces of the device	For each surface, map the graphical components	For each component, choose the polygon to project on the surface in the selected state	The state of the front side of the surface: on|off	The state of the back side of the surface: on|off

Table 10. artifact G: The Projection Guide .

C A TUTORIAL ON USING THE FLECTO METHOD TO DEVELOP AN APPLICATION

This is a tutorial on how to apply the Flecto Method to prototype a fully functional foldable device with its FoUI. The tutorial demonstrates step-by-step how to carry out the method and how to fill the artifacts in each step. The Flecto method is independent from the Flecto tools. However, the tutorial makes explicit reference to the tools if there is a difference between in performing the step manually or using the Flecto tools. 7

C.1 Develop your Idea

We want to create a device to browse scientific articles. This device is expected to reduce the use of papers, while giving the user the same experience. It should be compact and easy to hold in one's pocket.

C.2 Design the Device Prototype

C.2.1 Define the device. Our foldable device can be folded twice in the middle into a small rectangle, like a piece of paper. When completely unfolded, the two upper corners can be folded as well to move forward and backwards.

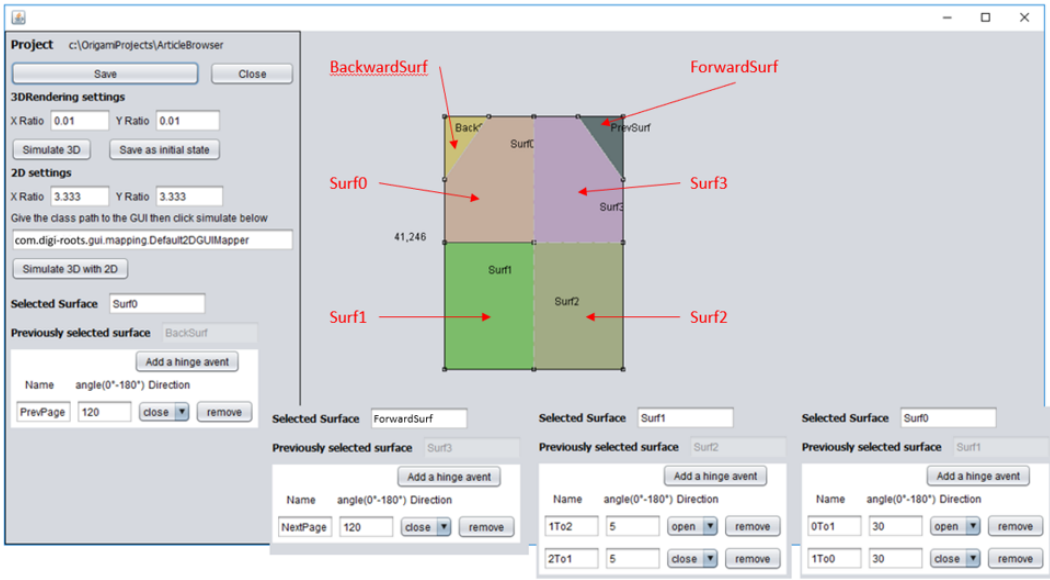

Fig. 15. Design the foldable device and define the physical events using Flecto Designer.

Surface	Description	Front Display	Back Display
Surf0	The top-left part of the device excluding the top-left triangle	✓	✓
Surf1	The bottom-left part of the screen.	✓	✓
Surf2	The bottom-right part of the screen.	✓	✓
Surf3	The top-right part of the screen excluding the top-right triangle.	✓	✓
BackSurf	The top-left triangle of the device.	✓	✓
NextSurf	The top-right triangle of the device.	✓	✓

Table 11. Use Case artifact A: The Surfaces.

Hinge surfaces	Friendly name
Surf0,Surf1	HalfOpenHinge
Surf1,Surf2	FullOpenHinge
BackwardSurf,Surf0	BackHinge
ForwardSurf,Surf3	NextHinge

Table 12. Use Case artifact B: The Hinges.

Figure 15 shows the designer and the surface. We notice the vally folds in light lines, and one mountain fold in a darker line. These hinges allow to fold the device in the middle vertically and then another fold in the middle horizontally to make a small rectangle with 1/4 of the total surface. When the device is fully folded, it can be carried out in a pocket. The initial state of the device is fully folded. We fill the artifacts A and B as in the tables 11 and 12.

If you are using Flecto, these artifacts are filled automatically inside the tool.

Hinge	Event Description	Physical Event	Condition	Action
HalfOpenHinge	This event partially opens the closed device	Open	=30	More
HalfOpenHinge	This event fully closes the partially-opened device	Close	=30	Back
FullOpenHinge	This event fully opens the half-opened device	Open	=5	Read Article
FullOpenHinge	This event partially closes the full-opened device	Close	=5	Back
~~BackwardHinge~~	~~This event opens the top-left corner of the device~~	~~Open~~	~~=120~~	
BackwardHinge	This event closes the top-left corner of the device	Close	=120	Previous Page
~~ForwardHinge~~	~~This event opens the top-right corner of the device~~	~~Open~~	~~=120~~	
ForwardHinge	This event closes the top-right corner of the device	Close	=120	Next Page

Table 13. Use Case artifact C: ECA Rules. All steps are included.

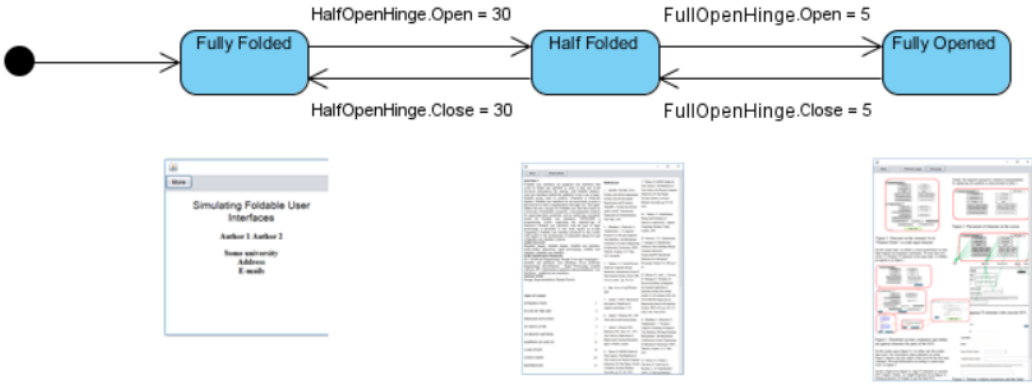

Fig. 16. The Use Case artifact D: The State Diagram.

C.2.2 Define physical events. We define the physical events in the artifact C: ECA Rules in the table 13. The hinge is identified by its two surfaces. For a visual guidance, please refer to Figure 15 of the designer where the name of the surface is displayed.

If you are using Flecto, the physical events can be created automatically for you. In fact, you can define the ECA rules directly. The figure on the designer shows how to define physical events on the surfaces diretly. You have to click on two surfaces (depicted in the fields "Selected Surface" and "Previously selected surface") to select the hinge in between, and then to define the physical events and the conditions. The tool gives a name for the event to facilitate mapping this event to the UI event later inside the mapper. The way to define our four physical events on the designer is displayed on the bottom of the figure, from left to right: hinge(Surf0,BackwardSurf), hinge(ForwardSurf,Surf3), hinge(Surf1,Surf2) and hinge(Surf0,Surf1).

C.2.3 Draw the State Transition Diagram. The device passes by three states:

(1) Fully Folded: When it is in the initial state and all the folds are folded.
(2) Half Folded: When the horizontal hinge is opened, and the vertical one is still fully closed.
(3) Fully Opened: When All the hinges are opened.

The artifact D is the state diagram which is illustrated in Figure 16. Notice that the screenshots are presented on the figure, but they will be added later in step 4.a.

If you are using Flecto, you do not need to define the state diagram because is it created automatically inside the tool, based on the events you define on hinges. The tool enables you to test at any time the functionality of the device and the folding operations.

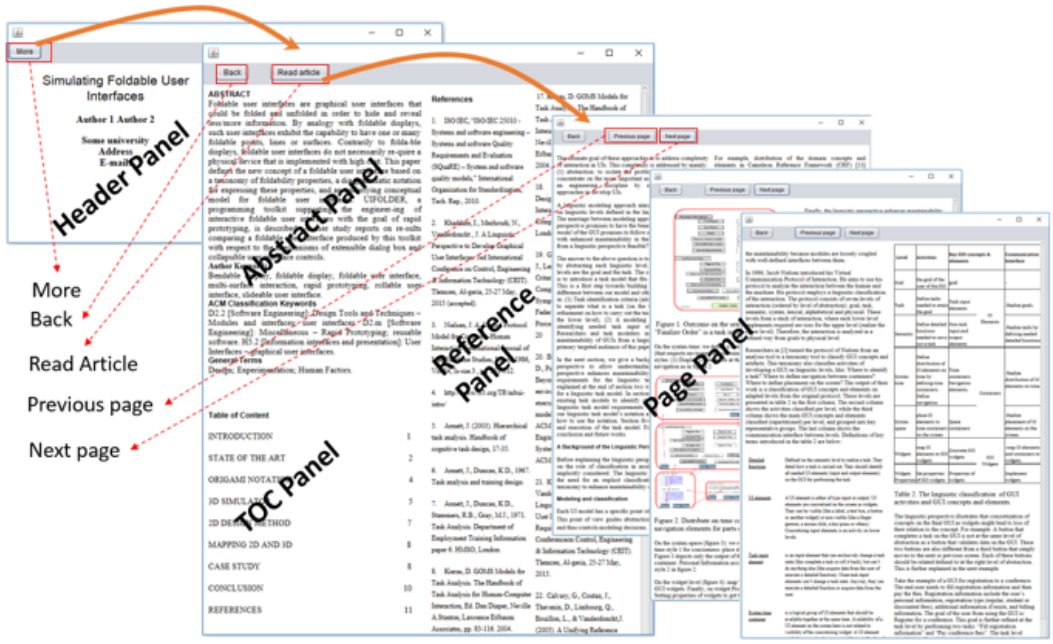

Fig. 17. The UI prototype for the Scientific Articles Browser case study.

C.3 The Application Prototype

C.3.1 Define the Software Application. Our software prototype consists of three forms. The first form of the UI contains only the article's title and the authors. To read the article, click on the "More" button and start browsing the article's pages. The second form displays the abstract and the table of content on the left side and the references on the right side. The third form displays the content of a page of the article (could be a one-column or a two column article). It displays a page at a time of the article. On every page, we have a next button and a back button to navigate on the article. The UI prototype is depicted in Figure 17.

As we mentioned earlier, The architecture of the software prototype is the concern of the developer. Flecto tries to impose the minimum requirements to integrate the software. Therefore, this part is not the concern of Flecto.

C.3.2 List Graphical Components. Figure 17 demonstrates the hierarchy of Graphical Components in our UI prototype. The first form contains a "Panel" we call the "Header". The second form consists of a "Panel" for the abstract, another for the "Table of Content" and a third for the "Refereneces". The other pages contain one panel to display the article pages, that we call the "Page" panel.

The artifact E of Graphical Components is depicted in the table 14. We filed only the top-level Graphical Components in our UI prototype. This step is mandatory to perform, even if you are using Flecto.

C.3.3 List UI events. There are numerous events that are triggered on the UI, but you should not bother to mention all of them. Consider navigation events (buttons and other UI events that move from one form to another or changes the view on the form) at the first place, in addition to other events you may consider important. This list should not be exhaustive at this step. You will come pack to it in the step 4.2. For our use case, we filed the artifact F on UI Events as depicted in the table 15. Other events are of less interest to our use case. This step is mandatory to perform, even if you are using Flecto.

Graphical Component	Description
Header Panel	A panel that contains the title of the article and the authors.
Abstract Panel	A panel that displays the abstract of the article.
TOC Panel	A panel that displays the table of content.
Reference Panel	A panel that contains the references used in the article.
Second Form	It contains the abstract, the TC, the references panel and the "back" button.
Page Panel	A panel that displays the pages of the article.
Third Form	It contains the page panel and the "back" button.

Table 14. The Use Case artifact E: The Graphical Components.

UI Event	Graphical Component	Description
More	Header Panel	Opens the second form that contains the abstract, the TOC and the references.
Back	Second Form, Third Form	Go back to the previous form.
Read Article	Page Panel	Opens the third form contains the Page Panel.
Previous Page	Third Form	Move to the previous page of the article on the Page Panel
Next Page	Third Form	Move to the next page of the article on the Page Panel

Table 15. The Use Case artifact F: The UI Events.

C.4 Consolidate the Models

One interesting benefit of using Flecto in this step is to provide visual feedback on the consolidation process. You can learn by trial and error, thanks the designer and the simulator. Besides, the artifacts are updated for you automatically when you change any thing visually on the designer.

C.4.1 Update the State Transition Diagram. Figure 16 demonstrates how the state diagram is updated with the screenshots of the UI prototype.

C.4.2 Define ECA Rules. With the updated state diagram, the UI events list and the physical events list, we can complete the ECA rules by filling the "Action" column in the artifact C in table 13. Fill the action column with a UI event for each rule.

C.4.3 Refine Physical and UI Events. In the table 13 that depicts the artifact C on physical events, we notice the existence of physical events that do not trigger any ECA rule. Opening the BackwardHinge does not trigger any ECA rule, therefore it can be removed. The same goes for the ForwardHinge.

It is also possible to notice that we missed some rules, which leads to defining new physical events or even UI events. This is the right moment to think about theses possibilities.

C.5 Fill the Projection Guide

Fill the artifact G of the Projection Guide is depicted in the table 16. The main difficulty in filling this artifact is to successfully imagine what are the visible side of surfaces at each state. If you are running the method manually, consider using a paper to simulate the device in different states and identify the visible surfaces. If you are using the Flecto Tools, things become visual and easier to identify. You can run the tool and test foldings and surfaces to fill this artifact.

Notice how we slice the forms to fit on the different displays. When the device is fully folded, the backside of the following surfaces are visible: Surf3, ForwardSurface and Surf2. We project the

State	Surface	Graphical Component	Polygon (X0,Y0)	Front face	Back Face
Fully Folded	Surf0	Header	0,0	off	off
	BackwardSurf	Header	0,0	off	off
	Surf1	TOC	0,0	off	off
	Surf2	Abstract	0,0	off	on
	Surf3	Header	0,0	off	on
	ForwardSurf	Header	W/4,H/2	off	on
Half Folded	Surf0	Abstract	0,0	off	on
	BackwardSurf	Abstract	W/4,0	off	on
	Surf1	TOC	0,0	on	on
	Surf2	Reference	0,H/2	off	on
	Surf3	Reference	0,0	off	on
	ForwardSurf	Reference	0,0	off	on
Fully Opened	Surf0	Page	0,0	on	on
	BackwardSurf	Page	0,0	on	on
	Surf1	Page	0,H/2	on	on
	Surf2	Page	W/2,H/2	on	on
	Surf3	Page	W/2,0	on	on
	ForwardSurf	Page	W*3/4,0	on	on

Table 16. The Projection Guide for the case study.

```
public interface Mapper2D {
    TextureInfo getTextureImage(String surfaceName);
    void runGUI(OrigamiShape shape);
    void exitGUI();
    Component getComponent(String surfaceName);
    OrigamiHingeEventListener getHingeEventListener();
}

public interface OrigamiHingeEventListener {
    void onHingeEvent(OrigamiEdgeAttributes.OrigamEdgeEvent event);

    public void onMouseClickedEvent(Point2D point, String surfaceName);

}
```

Fig. 18. The Java interfaces to implement by the mapper.

Abstract component on the surface Surf2. We project the Header component on the backside of the Surf3 surface. this surface is not rectangular because it misses the top-right triangle. Therefore the projection of the Header component will miss a part, that is projected on the surface ForwardSurf.

C.6 Develop the Mapping Model

Based on the project guide, we developed a class in Java that slices the graphical components as described in the guide. Our prototype is written in Java so the choice of Java for the mapper is trivial. The class implements an interface that defines the essential functionalities for the projection. It also defines the mapping of the ECA rules and the UI actions. The interface structure is depicted in Figure 18

The constraint we impose on the prototype is to enable accessing its graphical components. Technically, creating public getters to the needed components. This meets our objective to leave the technical choices on the software prototype to the developer.

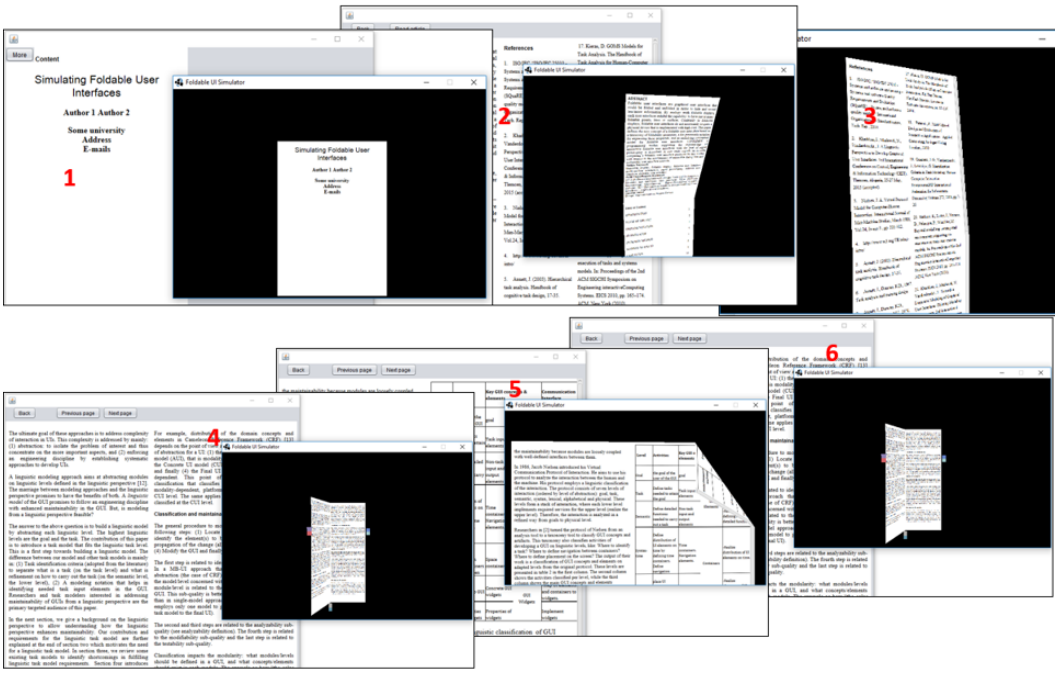

Fig. 19. The Scientific Articles Browser in the simulator.

In Figure 15 for the Flecto Designer, there is a field to introduce the name of the Mapper class. Give the full class name of your class in this field. The value of this field by default is the class Default2DGUIMapper. This is the default mapper class that simply create a form for each surface and allows you to test you static UI sketches.

C.7 Evaluate

Without a tool support, evaluating such a model is hard to achieve. The Flecto Tools provide the necessary means to simulate the device and the application running on it a virtual environment, where the user can interact with the system and evaluate its benefits.

To run the simulator from the designer, press the button "Simulate 3D". If the full name of the mapper class we developed in the previous step is indicated in the designer, the simulator will run simulating the device and the software application. Figure 19 illustrates the execution in the simulator on several steps. In the figure in (1), the device is fully folded and we can see the paper title and authors appearing on the first page. In (2), when opening the device, we see how the displays in the inside of the device are activated and project the abstract and the table of content. in (3) we turn look at the other side of the device to see the references of the paper projected. In (4) we see how we open the device to start reading the article. Once the folds are opened, we start seeing the article's pages on the inside displays. In (5) we see how we move to the next page by folding the top-right corner of the device, and in (6) we see how the page has changed to display the next page. In the back ground of the simulator, we showed how the 2D application is running in synchronization with the device. The physical events are transferred to the software application and the forms of the UI are projected on the device surfaces.

Physical events are not the only type of events transferred from the device to the application. In fact, the simulator allows the user to directly interact with the content displayed on the device. For instance, if you have a button on the graphical component, you can click on it in the simulator

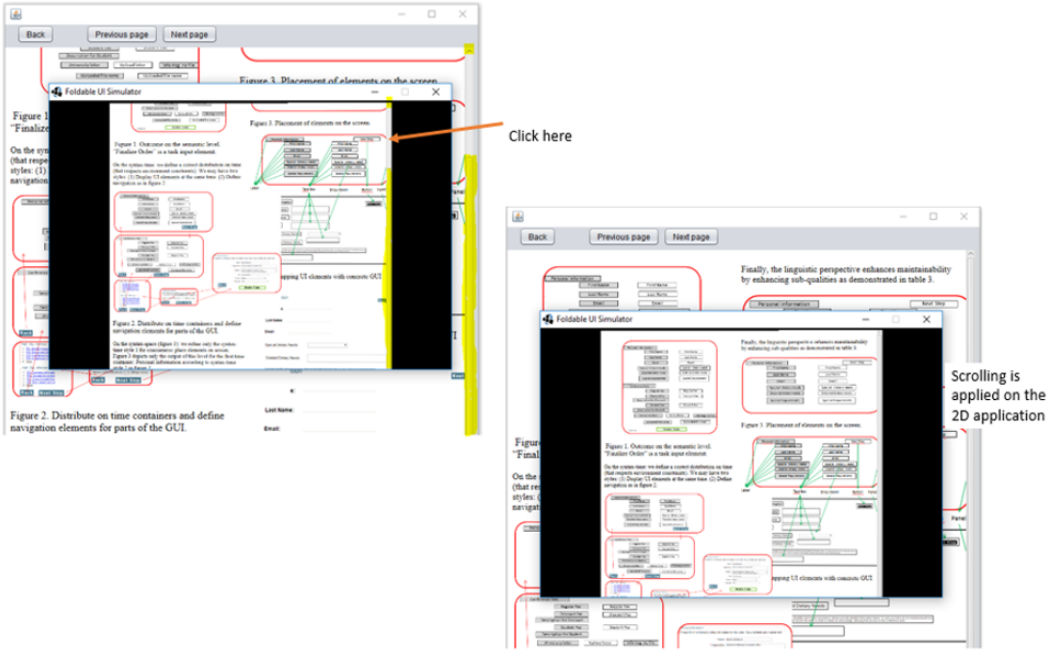

Fig. 20. The user can interact with the virtual display to scroll down.

and this action will be transferred to the software application to simulate a click on that button. To demonstrate that in our article browser, assume we have a long page we want to scroll down. The user can click on the horizontal scroll bar in the simulator and the software application will receive this event and scroll down is scroll down on its content. Consequently the projected content on the simulator is synchronized with the changes on the software application. This is illustrated in Figure 20.

Received July 2020; revised August 2020; accepted September 2020