http://dx.doi.org/10.14236/ewic/BCSHCI2023.14

A Systematic Lightweight Approach for Natural User Interfaces Prototyping

Baraa Alabdulwahab
School of Computing and Mathematical Sciences,
University of Leicester
University Road, LE1 7RH, Leicester, UK
ba184@leicester.ac.uk

Effie Lai-Chong Law
Department of Computer Science,
Durham University
Upper Mountjoy Campus, DH1 3LE Durham, UK
lai-chong.law@durham.ac.uk

The emergence of Natural User Interface (NUI) two decades ago brought a lot of potential by supporting intuitive and multimodal interaction and leveraging human sensorimotor skills. Systematic prototyping and development of NUI, however, received much less attention, and legacy methods of the previous Graphical User Interface (GUI) were applied to develop NUIs. This has led to usability issues in many cases. In this paper we introduce a systematic lightweight approach to aid designers and developers in prototyping NUIs based on the context of interaction. We demonstrate the application of this approach through the development of a pen-based NUI.

Natural user interface. Graphical user interface. Smartpen. Pen-based interaction

1. INTRODUCTION

Natural user interface (NUI) emerged as part of the post-WIMP design trend in the early 21st century (Mann, 2007). It promised to support intuitive and multimodal interactions by tapping into human sensorimotor skills such as touching, speaking, and gazing. However, prototyping and development methods of NUI received much less attention regardless of the difference between NUI and GUI (graphical user interface) interaction mechanism. A typical GUI-based interaction employs a fixed set of hardware: a screen (possibly with touch feature), a mouse and a keyboard. Development for GUI, therefore, handles screen content in extensive detail (Galitz, 2007) while input and output devices are rarely discussed because they are treated as known pre-conditions to the interaction. This is not the case for the NUI which involves unspecified and growing set of devices (Kortum, 2008) that are designed to facilitate natural interaction for different scenarios.

On the other hand, a number of studies in the NUI literature utilise novel commercial devices without justifying the selection of specific hardware used with regard to the context. For example, studies employing gestures (e.g., Jagodziński & Wolski, 2015; Lai et al., 2018; Phamduy, DeBellis & Porfiri, 2015; Zhao et al., 2014) considered the Microsoft Kinect[1] sensor without rationalising this choice over alternatives such as Leap Motion controllers[2]. Both devices can recognise gestures, but they have different interaction mechanisms (Feng et al., 2021). In some cases, this ‘off-the-shelf’ design approach led to usability issues (e.g., Jin, Ma & Zhu, 2022) who used Microsoft HoloLens[3] for gaze and head posture tracking, and the choice resulted in discomfort because of the weight of the device. The phenomenon of adopting specific hardware is very common in the NUI literature (e.g., Cheng, Yang & Liu, 2016; Jensen & Konradsen, 2018) which seems a legacy from GUI where relatively any screen could fit for the purpose. While this could be attributed to technical advantages such as the availability of SDKs and tutorials (Jagodziński & Wolski, 2015), it supports the hypothesis that natural interaction design is being influenced and limited by hardware (Yu, Denham & Searight, 2022). This issue entails seeking a systematic and pragmatic approach that can aid designers and developers in prototyping NUIs based on the context of interaction. We have been motivated to tackle this challenge.

In this paper, we introduce a systematic lightweight approach that is aimed to support NUI prototyping. We demonstrate the application of this approach through the development of a pen-based NUI. The ultimate goal of our work is to disseminate the use of such a method. Our contribution is to provide an initial and demonstrable systematic solution, which may stimulate further work along this line of inquiry. For our future work, we plan to evaluate the

[1] https://en.wikipedia.org/wiki/Kinect
[2] https://en.wikipedia.org/wiki/Leap_Motion
[3] https://en.wikipedia.org/wiki/Microsoft_HoloLens

approach with a number of designers and developers across a range of application contexts.

2. RELATED WORK

2.1 Formal and Informal UI Prototyping

Formal and informal approaches were applied for prototyping UIs. Palanque et al. (2009) used Petri nets for prototyping high fidelity safety-critical UIs. Formal prototyping, in general, aims to employ rigor and accurate methods to model and document the UI at functional level (e.g., events, iterations and conditions) as well as at physical level (e.g., bouncing button or sliding knob), hence the UI can be developed and assessed systematically (Dix, Ghazali, Gill, Hare, & Ramduny-Ellis, 2009). Examples of formal methods are State Transitions Networks (STN), Petri Nets, Z notation and XML-based models in Davani, Shirehjini and Daraei (2018) and Tsalmpouris et al. (2021). Informal prototyping, on the other hand, uses quick and flexible ways to describe and illustrate the interface morphologically (how it looks) and functionally (how it works). Examples of informal methods are sketches, scenarios, snapshots, storyboards, and natural language in Trætteberg (2002) and Carter and Hundhausen (2010).

Both formal and informal prototyping approaches have strengths and drawbacks which are summarised in Table 1. Bowen, Weyers, and Liu (2022) considered that informal prototyping is the step before the formal stage. However, we argue that applying this sequence might aggravate the workload on designers and that the necessity of such a two-step approach is contingent on the expertise of designers. Novices may need to sketch ideas or use pseudocodes in natural language before translating them with formal methods whereas experts can circumvent informal steps and proceed with formal ones.

***Table 1**. Comparison between formal and informal UI prototyping approaches*

	Formal	Informal
Strengths	• Rigor • Machine understandable • Support design and verification	• Quick and easy • Very expressive • Visual presentation • Rapid modifications • Do not impose constraints
Drawbacks	• Limited to professionals • Very strict • Focus on details from the beginning • Consume time and effort.	• Less accurate (fuzzy) • Do not support automatic verification.

Because of these differences, a further semi-formal prototyping emerged as less strict than formal and more accurate than informal approaches in Chklovski, Ratnakar, and Gil (2005) and Law et al. (2023). Such a pragmatic balance considers using informal data such as sketches to produce quick and lightweight initial design and then to use formal methods to eliminate ambiguities and provide accurate description to the UI when necessary.

2.2 Semi-formal UI Prototyping

Semi-formal prototyping for natural interfaces in Vitzthum (2006), Shaer and Jacob (2009) and Bataille et al. (2022) modelled the NUI in two separate but interconnected layers: a real layer to accommodate physical entities (e.g., a joystick), and a virtual layer to accommodate digital entities (e.g., digital character). Entities and relations within the real layer were usually explained and documented informally using sketches and free text, while those within the virtual layer were modelled and documented formally using state machines, pseudocode and visual (UML-like) modelling languages.

One shortcoming of such semi-formal approaches is that the produced models were more descriptive and less constructive; they aided in depicting and documenting the NUI model, but they did not demonstrate the incremental construction of the interface based on a list of requirements as suggested by De Haan (2015). Incremental prototyping is very beneficial for developing complex systems which involve both hardware and software components (Palanque, Ladry, Navarre, & Barboni, 2009; de Haan, 2015).

Schmidt (2000) proposed a semi-formal and incremental algorithm for developing context-aware devices for ubiquitous computing environments. The algorithm, nevertheless, was meant for implicit interaction only, and could not be used in its original form to prototype NUIs, which involve both implicit and explicit interactions. Few modifications to the algorithm, however, could enhance its potential to achieve this objective as described in the next section (Section 3).

Before presenting our proposed systematic approach for prototyping NUIs, it is relevant to differentiate the notion of implicit and explicit interaction in the field of HCI (Norman, 2010; Schmidt, 2000; Serim & Jacucci, 2019). One major difference pertains to user input and intention. Explicit interactions involve deliberate actions like clicking, typing, and choosing, with users fully aware and engaged whereas implicit interactions occur passively based on user behaviour and context, often without user awareness, causing privacy and ethical concerns. Another difference is that explicit interactions demand effort and customisation while implicit interactions adapt based on observed patterns. Context in explicit interactions relies on user input, while implicit interactions infer context from behaviour and environment. A balance between the two is vital for positive user experience.

3. PROTOTYPING APPROACH

3.1 Schmidt's Algorithm for Implicit Interaction

Schmidt's (2000) work involved researching methods to create prototypes for implicit interaction. He defined implicit interaction as actions performed by the user that are not meant to interact with a computer system, but the system could understand these actions as inputs (i.e., contextualisation).

For example, leaving the office towards the garage at 5.30pm indicates the end of a working day, therefore some tasks such as logging out of the office computer system or locking the door can be performed automatically. A mobile screen that glows (turn on) when the mobile is left up towards the user's face is another example.

Schmidt also defined the notion 'context' as the **environmental or surrounding conditions which can be used to trigger or guide the computer reaction**. Examples of context are location, temperature, surrounding noise, identity of the user who is operating a system, and so on. Building on the notions of 'implicit interaction' and the 'context', Schmidt proposed the algorithm (Figure 1) to help develop devices for his vision of implicit interaction.

```
1. create the set $C$

2. set D = {}

3. for each $C_i \in C$
       define $A_i$ // accuracy
       define $U_i$ // update rate
       identify $S_i$ // a sensor device
                      //that is appropriate
       if cost($S_i$, $A_i$, $U_i$) is acceptable then
               $D = D \cup \{(C_i, S_i, A_i, U_i)\}$
       fi
   next

4. if $D \neq \{\}$ then
       for each vector $D_i$ in D
             define a set of application reaction $R_i = \{(I_{ij}, R_{ij})\}$
             // $I_{ij}$ is input range, application reaction pairs $I_{ij}$
             // $R_{ij}$ is application reaction

   else
       // implicit interaction is not used
       // (either no condition that are useful or too costly)
```

Figure 1. Schmidt (2000) algorithm for implicit HCI (reproduced from original)

The algorithm can be explained as follows:

- On Step 1: C is the set of conditions that define the context and carries information that is useful for the application of interest. Each element C_i stands for one condition, e.g., location, temperature, etc. The set is created by analysing the context (e.g., during requirement gathering).
- On Step 2: D is initialised. It represents the aimed hardware system. At the beginning, the set is empty and has no sensing devices identified yet.
- On Step 3: For each condition C_i, the accuracy A_i and the update rate U_i that are needed to make the measurements useful are defined. Then a sensor S_i that matches these requirements is identified. If the sensor's cost is acceptable, then a vector matching the condition and the sensor is added to the set D.
- On Step 4: If set D is not empty (there are conditions that are feasible to measure) then for each one of the set's condition-sensor vectors (C_i, S_i, A_i, U_i), application reaction R_{ij} is defined based on a specific interaction range (or trigger) identified as I_{ij}.

This algorithm provides a systematic and incremental method for prototyping implicit interaction devices. The outcome can be seen in set D which represents the resulting hardware, and in set {R_i} which represents the software. Nevertheless, the algorithm cannot be applied directly for developing NUIs because it only serves implicit interaction (i.e., it is sensor-based without output), and because it allows a partial implementation: in Step 3, sensors might be skipped if too costly while in Step 4 verification tests only whether set D is empty or not, but the capability of the resulted hardware (set D) to cover the original requirements (set C) is not checked or evaluated. Therefore, modifications in the next section were suggested to overcome these gaps.

3.2 Revised Version of Schmidt's Algorithm for NUI Prototyping

As mentioned in Section 2.2, there were gaps in the original Schmidt's (2000) algorithm. In this subsection, we list two requirements that should be added to the algorithm:

Requirement 1. It should handle both implicit and explicit interactions.

Requirement 2. It should not allow a partial implementation.

Explicit interaction of the first requirement implies that a user will be aware of the existence of the interface and may engage with it physically through direct manipulation or by wearing the interface. This adds a further dimension to the interaction where ergonomics and aesthetics should be cared of. For example, an effective but very heavy or clumsy hardware (e.g., head mounted devices for virtual reality) should be avoided or assessed if the intended interface is wearable. By considering this third physical dimension, along with the two initial requirements above, the algorithm can be as follows:

```
1.    create the set $C$ // conditions or contexts of interest

2.    set $D$ = {}, $D_{in}$ = {}, $D_{out}$ = {}, R = {}

3.    for each $C_i \in C$
          identify $S_i$ // an appropriate sensor
          if adequate($S_i$, tech_req, phys_req, cost) then
              $D_{in} = D_{in} \cup \{(C_i, S_i)\}$
          fi
      next

4.    for each pair $(C_i, S_i) \in D_{in}$
          define a set of application reaction $R_i = \{(I_{ij}, R_{ij})\}$
          // $I_{ij}$ is input range, $R_{ij}$ is application reaction
          $R = R \cup \{ R_i \}$        // record the reaction
      next

5.    for each $R_i \in R$
          for each pair $(I_{ij}, R_{ij}) \in R_i$
              if $R_{ij}$ contains output
                  identify $O_j$ // an appropriate output device
                  if adequate($O_j$, tech_req, phys_req, cost) then
                      $D_{out} = D_{out} \cup \{(R_{ij}, O_j)\}$
                  fi
              fi
          next
      next

6.    $D = D_{in} \cup D_{out}$ // input and output devices assembly

7.    if capability($D$, $C$, R) is accepted then
          return $D$            // the final result
      else
          // Implementation not possible
          // (either no interactions exist or no appropriate hardware)
```

Figure 2. *Revised version of Schmidt's algorithm for NUI prototyping*

The new algorithm in Figure 2 is different from the original version (Figure 1) in the following points:

- On Step 2: We defined three auxiliary sets, D_{in} and D_{out} to hold input and output devices in order, and R to hold all application reactions. At the beginning these sets are empty as no devices or interactions are identified yet.
- On Step 3: After identifying a sensor S_i that matches the condition C_i, we check sensor's adequacy against a criterion that involves three items: technical requirement, physical requirement and cost. Technical requirement (tech_req) covers the accuracy and update rate of the original version (Figure 1) as well as any other relevant technical specification such as power consumption. Physical requirement (phys_req) was added to represent ergonomics and aesthetics which the device should meet, such as weight or dimensions limits. Cost is same as in original version. If the sensor satisfies the criterion, the pair (C_i, S_i) will be added to the input devices set D_{in}.
- On Step 4: All prospective application reactions R_i which belong to each condition-sensor pair (C_i, S_i) were grouped in the set R.
- On Step 5: We iterate over all possible application reactions R_{ij} within the set R in order to identify required output devices. For each output device identified O_j, we check its adequacy against a criterion in comparable assessment to the one applied on input devices (sensors) in Step 3. If the output device satisfies the criterion, the pair (R_{ij}, O_j) will be added to the output devices set Dout.
- On Step 6: We assemble all collected input and output devices in a single system D.
- On Step 7: To avoid a partial implementation, we assess the capability of the final system D to sense original conditions (C) and provide required reactions (R). If assessment's result was accepted, D is then representing the aimed NUI system.

In the following section, we demonstrate the application of the revised algorithm through building a pen-based NUI.

4. DEVELOPING A SMARTPEN SYSTEM PROTOTYPE (SSP)

4.1 Context Analysis for SSP

According to some researchers (Hauger, Paramythis & Weibelzahl, 2011; Lam & Baudisch, 2005), a typical digital reading activity using a PC covers the following interactions: scrolling, navigation through links, text lookup, text input, and zooming (in small screens). Building analogous capabilities for pen-based NUI implies that the SSP should provide affordances in Table 2.

***Table 2**. Context analysis for SSP*

PC system	Smart Pen System
Scrolling	Paper flipping
Links navigation	Changing books, flipping paper, or through tapping with the pen on printed elements to display animated content, e.g., playing video on a screen
Text lookup	Paper flipping, or tapping on printed text to perform a search, results to be shown on a screen
Text input	Handwriting
Zooming	Moving paper closer to eyes, or tapping on printed elements, e.g., images and graphs, to display a higher resolution version

SSP affordances in Table 2 could be translated into the following input and output actions:

- Input actions (user actions): Changing books, flipping paper, tapping with pen, moving the pen on paper for writing or tapping.
- Output actions (system reactions): Displaying content and playing sounds. NB: Content output could either be static (directly printed on paper) or digital which require a correspondent digital display system.

Having the interaction context analysed as above allows SSP prototyping to be performed as follows:

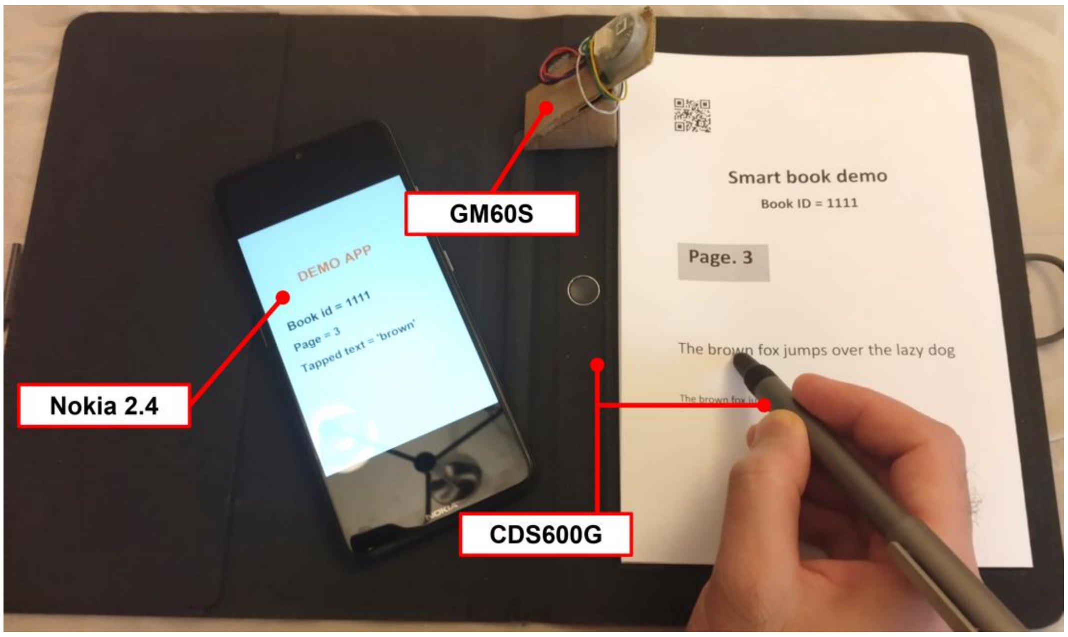

***Figure 3**. The assembly of SSP*

4.2 Implementation

Step 1: Creating Set C
Set C involves conditions of context that a system should sense. It can be written as:

C = {book id, page id, tapping location, tapping status}

In this set book id identifies the current book being used, page id identifies the current page that is being read, tapping location identifies the location of pen tip and tapping status tells if the smart pen is on or off paper.

Step 2: Defining auxiliary sets
$D = \{\}$, $D_{in} = \{\}$, $D_{out} = \{\}$, $R = \{\}$

Step 3: Identifying sensors
For detecting book id we found a QR code scanner (model: GROW GM60-S), which has low response time (< 1s), compact size (~20 x 20 x 7 mm) and low cost. Therefore, the sensor passed the 'adequate' checking in the development algorithm. Printing QR codes, which encode the book id on the external cover of books and fixing the sensor so that it can scan the code, will enable to identify the current book being used. It is worth noting that using the sensor required and additional microcontroller to perform readings. Therefore, we used a microcontroller (model: ESP32 Wemos D1 mini) for this purpose and put both components into a single enclosure. We will label this structure as 'GM60S' for simplicity (Figure 3).

For detecting page id, the same GM60S sensor can be used if QR codes are printed on pages. Practically, adding QR codes to pages will eliminate the need to use a dedicated code for book id, as both identifiers book id and page id can be embedded in codes printed on each paper. Therefore, no additional sensors were added at this step.

For detecting tapping location, we found a specialised digitiser pad (model: CDS600G) which involves a smartpen and a sensing pad. The pad can precisely identify the location of the smartpen and transfer location data using Bluetooth Low Energy (BLE) connection to a host system. The pad's response time (< 50ms), size (~ 400 x 250 mm), weight (< 500g), and cost made it pass the 'adequate' checking of the development algorithm. Putting paper on the sensing pad allows a natural writing through the smartpen, while tracking pen's location digitises what is being written.

For detecting tapping status, the same digitiser pad (CDS600G) could detect the pressure on the pen tip (pressure $\in [0, 100]$). Defining two pressure ranges (pen off paper: pressure ≤ 5; pen on paper: pressure > 5) enables to determine tapping status. Therefore, no additional sensor is required for detecting the tapping status.

Putting both sensors (GM60S + CDS600G) together resulted in an input system that can detect the four conditions of the set C. The system can be modelled as:

D_{in} = {(book id, GM60S), (page id, GM60S), (tapping location, CDS600G), (tapping status, CDS600G)}

Step 4: Defining system reactions.
We define system reactions for each (condition, sensor) pair as follows:

(book id, GM60S) →
R_{book_id} = { (ISBN$_n$, notifyNewBook(ISBN$_n$)) }

(page id, GM60S) →
R_{page_id} = { (n, notifyNewPage(n)) }

(tapping location, CDS600G) →
$R_{tapping\_location}$ = { (loc(x,y), notifyLocation(x,y)) }

(tapping status, CDS600G) →
$R_{tapping\_status}$ = { (down, notifyTapping(down)), (down, multimedia()), (up, notifyTapping(up)) }

R_{book_id} is the system reaction that occurs when book id condition is changed (i.e., when user replaces a book with another). When the condition is changed, the system reacts by throwing a notification 'notifyNewBook', which indicates the new book id value. The similar mechanism applies for the other reactions R_{page_id}, $R_{tapping\_location}$, $R_{tapping\_status}$.

$R_{tapping\_status}$ has the special reaction 'multimedia' which is a request to provide digital content that is relevant to the tapping action. For example, if the user taps over a link to video, then the video should play through a screen and a speaker.

Eventually, the complete list of system reactions is provided in set R as:

R = { R_{book_id} , R_{page_id}, $R_{tapping\_location}$, $R_{tapping\_status}$ }.

Step 5: Identifying output devices.
As stated above, $R_{tapping\_status}$ comprises the method 'multimedia' which is a request to provide digital content through a screen/speaker. Hence, we utilise a smartphone (model: Nokia 2.4) that has a 6.5" screen to provide multimedia output. The technical, physical, and cost specifications of this phone make it pass the 'adequate' checking in the development algorithm. The phone also acts as a host system to receive and process data provided by input sensors GM60S and CDS600G over a BLE connection.

Hence, the output devices set D_{out} can be simply defined as:

D_{out} = { (multimedia, Nokia 2.4) }.

Step 6: Assembling input/output devices.
We assemble devices in D_{in} and D_{out} together to produce the final system (Fig. 3) represented as:

D = { (book id, GM60S), (page id, GM60S), (tapping location, CDS600G), (tapping status, CDS600G), (multimedia, Nokia 2.4) }.

Step 7: Confirming capability of System D.
As all conditions in the set C are matched with adequate sensors, and all output reactions included within the set R are matched with adequate output devices, therefore we can conclude that capability checking of the System D to satisfy interaction requirements is accepted. The overall formal model for the SSP prototype is presented in Figure 4.

C = { book id, page id, tapping location, tapping status };

D = { (book id, GM60S), (page id, GM60S), (tapping location, CDS600G), (tapping status, CDS600G), (multimedia, Nokia 2.4) };

D_{in} = { (book id, GM60S), (page id, GM60S), (tapping location, CDS600G), (tapping status, CDS600G) };

D_{out} = { (multimedia, Nokia 2.4) };

R = { R_{book_id}, R_{page_id}, $R_{tapping\_location}$, $R_{tapping\_status}$ };

R_{book_id} = { (ISBN$_n$, notifyNewBook(ISBN$_n$)) };

R_{page_id} = { (n, notifyNewPage(n)) };

$R_{tapping\_location}$ = { (loc(x,y), notifyLocation(x,y))};

$R_{tapping\_status}$ = { (down, notifyTapping(down)), (down, multimedia()), (up, notifyTapping(up)) };

Figure 4. *Formal model for SPP*

5. TECHNICAL ASSESSMENT

The assessment of the SSP involved testing tapping and paper-flipping actions using a 5-page sample article printed on A5 paper. QR codes were generated using a free online service (The QR Code Generator[4]), and they were placed at 0.5" from the top and left sides of pages. The text size was 9pt (double spaced) for paragraphs and captions, 12pt (bold) for headings and 14pt (bold) for the title.

A total of 160 tapping points, and a total of 20 paper flips were recorded during the technical assessment. The average response time was 0.9 second for tapping, and 1.3 seconds for flipping. The maximum was 1.2 seconds for tapping and 1.9 seconds for flipping, and the minimum was 0.7 for tapping and 0.9 for flipping. The accuracy of tapping was ~97%, and errors (the resulted action did not belong to the tapped word on paper) were linked to short tapping interval (<2 seconds between taps). The accuracy for paper-flipping was 65% (7 out of 20 flips failed to generate a response) and errors were linked to the deformation (bending) of paper that made the QR code sensor unable to recognise codes on the page corner.

Overall, the system performance was acceptable for such a proof-of-concept prototype. Further enhancement could be done by changing the location of QR code.

6. DISCUSSION

We could observe the potential of the development approach (Figure 2) in the following areas:

Incremental development. The development of SSP was done incrementally. We built the input system (D_{in}) using two input technologies, then we defined the reactions based on the input system, and later, we constructed the output system (D_{out}) based on reactions. We finally assembled the overall system (D) and verified its capability to match the requirements. This process which spanned over several weeks reduced the complexity of dealing with hardware. Also, we were able to assess alternatives using different factors (i.e., the method ***adequate***), and to focus development efforts on a single interaction aspect as pieces of hardware were supplied.

Lightweight formal modelling. The formal model of SSP (Figure 4) provides a precise and concise description to the structure and functionality of the system. It clarifies which part of a system is responsible for responding to a specific input and how the response is generated, and through which channel the response is delivered to user (if any). Producing such a model did not involve complex mathematical syntax, nor was it vague similar to plain text and sketches of informal models.

Extension and replication. Through the formal model, it would be easy to extend the system either by adding more sensors (set D) or by improving the system's reaction (set R). Practically, the incremental development of SSP is an example of extending functionalities, from recognising books and papers using a QR code scanner, to a full interaction with paper when another sensor (i.e., the digitiser pad) was added. Furthermore, replicating the prototype using other hardware (e.g., utilising a larger digitiser pad) is very straightforward as we only need to change the corresponding device(s) while the remaining parts of the formal model remain unaltered.

Verification and maintenance. During the technical assessment, we performed dedicated tests (accuracy + response time) for SSP in relation to page flipping. The tests were performed because of our pre-knowledge that there is a specialised sensor that observes paper flipping. However, assuming the case where there are two different teams engaged in prototyping, one for development and the other for testing. The formal model could then inform the testing team about the internal structure and functionality of the system; hence they could perform such dedicated tests. In terms of maintenance, the formal model would help to replace, fix, and re-test the flawed functionality (the QR code recognition in our case) without affecting the other parts of the system.

7. CONCLUSION

The development approach as described in Figure 2 facilitated the creation of the SSP prototype from scratch and the generation of a corresponding formal model. The formal model (Figure 4) provides a precise and concise description of a system's structure and functionality. It clarifies which part of the system is responsible for responding to a specific input, and how the reaction will look. Producing this model does not involve complex mathematical syntax. Nor is the model vaguely similar to plain text of informal approaches applied in many previous studies.

Through the model, it would be easy to extend the system either by adding more sensors or by improving system's reaction. Maintenance-wise, it would be simple to change a sensor (e.g., replacing discontinued products) with new alternatives while the remaining parts of the formal model stay unaltered.

[4] https://www.the-qrcode-generator.com/

While we are able to demonstrate the application of the proposed approach with the case of pen-based UI, we are aware that the scope of the application context needs to be much widened to establish the validity, reliability and applicability of the approach. Furthermore, designers and developers interested in NUIs should have been involved in evaluating the approach to provide feedback for its refinement. These limitations will be addressed in our future work.

Overall, we have taken an initial but significant step for the creation of a systematic lightweight approach for prototyping NUIs based on context of interaction. More research along this line of inquiry is called for.

ACKNOWLEDGEMENTS

The first author would like to acknowledge and thank the *Council for Academics At Risk* (CARA) for their generous support to complete this study as a part of his PhD research.

REFERENCES

Bataille, G., Gouranton, V., Lacoche, J., Pelé, D., & Arnaldi, B. (2022). Unified Model and Framework for Interactive Mixed Entity Systems. Computer Vision, Imaging and Computer Graphics Theory and Applications. VISIGRAPP 2020 (pp. 3-25). Valletta Malta: Springer, Cham.

Bowen, J., Weyers, B., & Liu, B. (2022). Creating Formal Models from Informal Design Artefacts. International Journal of Human–Computer Interaction. doi:10.1080/10447318.2022.2095833.

Carter, A. S., & Hundhausen, C. D. (2010). How is User Interface Prototyping Really Done in Practice? A Survey of User Interface Designers. IEEE Symposium on Visual Languages and Human-Centric Computing (pp. 207-211). Leganes Spain: IEEE.

Cheng, H., Yang, L. and Liu, Z., 2015. Survey on 3D hand gesture recognition. *IEEE transactions on circuits and systems for video technology*, *26*(9), pp.1659-1673.

Chklovski, T., Ratnakar, V., & Gil, Y. (2005). User interfaces with semi-formal representations: a study of designing argumentation structures. IUI '05: Proceedings of the 10th international conference on Intelligent user interfaces (pp. 130-136). San Diego California: ACM.

Davani, A. M., Shirehjini, A. A., & Daraei, S. (2018). Towards interacting with smarter systems. Journal of Ambient Intelligence and Humanized Computing, 187-209.

de Haan, G. (2015). HCI Design Methods: where next? from user-centred to creative design and beyond. ECCE '15: Proceedings of the European Conference on Cognitive Ergonomics 2015 (pp. 1-8). Warsaw Poland: ACM.

Dix, A., Ghazali, M., Gill, S., Hare, J., & Ramduny-Ellis, D. (2009). Physigrams: modelling devices for natural interaction. Formal Aspects of Computing, 613-641.

Feng, Y., Uchidiuno, U.A., Zahiri, H.R., George, I., Park, A.E. and Mentis, H., 2021. Comparison of Kinect and leap motion for intraoperative image interaction. *Surgical Innovation*, *28*(1), pp.33-40.

Galitz, W.O., 2007. *The essential guide to user interface design: an introduction to GUI design principles and techniques*. John Wiley & Sons.

Hauger, D., Paramythis, A. and Weibelzahl, S., 2011. Using browser interaction data to determine page reading behavior. In *User Modeling, Adaption and Personalization: 19th International Conference, UMAP 2011, Girona, Spain, July 11-15, 2011. Proceedings 19* (pp. 147-158). Springer Berlin Heidelberg.

Jagodziński, P. and Wolski, R., 2015. Assessment of application technology of natural user interfaces in the creation of a virtual chemical laboratory. *Journal of Science Education and Technology*, *24*, pp.16-28.

Jensen, L. and Konradsen, F., 2018. A review of the use of virtual reality head-mounted displays in education and training. *Education and Information Technologies*, *23*, pp.1515-1529.

Jin, Y., Ma, M. and Zhu, Y., 2022. A comparison of natural user interface and graphical user interface for narrative in HMD-based augmented reality. *Multimedia tools and applications*, *81*(4), pp.5795-5826.

Kortum, P., 2008. *HCI beyond the GUI: Design for haptic, speech, olfactory, and other nontraditional interfaces*. Elsevier.

Lai, N.K., Ang, T.F., Por, L.Y. and Liew, C.S., 2018. Learning through intuitive interface: A case study on preschool learning. *Computers & Education*, *126*, pp.443-458.

Lam, H. and Baudisch, P., 2005, April. Summary thumbnails: readable overviews for small screen web browsers. In *Proceedings of the SIGCHI conference on Human factors in computing systems* (pp. 681-690).

Law, Y. C., Wehrt, W., Sonnentag, S., & Weyers, B. (2023). Obtaining Semi-Formal Models from Qualitative Data: From Interviews Into BPMN Models in User-Centered Design Processes. International *Journal of Human–Computer Interaction*, 476-493.

Mann, S., 2007, June. Natural Interfaces for Musical Expression: Physiphones and a physics-based organology. In *Proceedings of the 7th international conference on New interfaces for musical expression* (pp. 118-123).

Norman, D. A. (2010). Natural user interfaces are not natural. interactions, 17(3), 6-10.

Palanque, P., Ladry, J.-F., Navarre, D., & Barboni, E. (2009). High-Fidelity Prototyping of Interactive Systems Can Be Formal Too. HCI 2009: Human-Computer Interaction. New Trends (pp. 667-676). San Diego CA: Springer.

Phamduy, P., DeBellis, M. and Porfiri, M., 2015. Controlling a robotic fish via a natural user interface for informal science education. *IEEE Transactions on Multimedia*, *17*(12), pp.2328-2337.

Serim, B., & Jacucci, G. (2019). Explicating" Implicit Interaction" An examination of the concept and challenges for research. In *Proceedings of Conference on Human Factors in Computing Systems (CHI'19)* (pp. 1-16).

Shaer, O., & Jacob, R. J. (2009). A specification paradigm for the design and implementation of tangible user interfaces. ACM Transactions on Computer-Human Interaction, 1-39.

Schmidt, A., 2000. Implicit human computer interaction through context. *Personal technologies*, *4*, pp.191-199.

Trætteberg, H. (2002). Model-based User Interface Design. Norwegian University of Science and Technology.

Tsalmpouris, G., Tsinarakis, G., Gertsakis, N., Chatzichristofis, S. A., & Doitsidis, L. (2021). HYDRA: Introducing a Low-Cost Framework for STEM Education Using Open Tools. Electronics, 3056.

Vitzthum, A. (2006). SSIML/AR: A Visual Language for the Abstract Specification of Augmented Reality User Interfaces. 3D User Interfaces (3DUI'06) (pp. 135-143). Alexandria Virginia: IEEE.

Yu, J., Denham, A.R. and Searight, E., 2022. A systematic review of augmented reality game-based Learning in STEM education. *Educational technology research and development*, *70*(4), pp.1169-1194.

Zhao, Y., Qin, Y., Liu, Y., Liu, S., Zhang, T. and Shi, Y., 2014, February. QOOK: enhancing information revisitation for active reading with a paper book. In *Proceedings of the 8th International Conference on Tangible, Embedded and Embodied Interaction* (pp. 125-132).