Software Architecture and Software Usability: A Comparison of Data Entry Form Designing Configurable Platforms.

Lawrence Fatsani Byson
Computer Science Department
University of Malawi
Zomba, Malawi
lbyson@cc.ac.mw

Tiwonge Davis Manda
Computer Science Department
University of Malawi
Zomba, Malawi
tmanda@cc.ac.mw

Abstract--- **The objective of the research was to discuss how software architecture shapes the usability of configurable software in data entry-form design. The research process was conducted in three stages. The first stage focused on usability studies for DHIS2 custom form editor from which empirical data was collected from 11 participants. The second phase centred on experimentation with Sketch2Code and Commcare. The final stage focused on solution prototyping and evaluation. The research found out that usability is enhanced in configurable platforms through the availability of interface elements for achieving desired goals with the platform without the need for writing code and meta-design. Constraining factors to usability include lack of functionality to advance the appearance of interfaces beyond the basic outlook and having predefined functions with limited room for innovation outside the predefined range. The research also found out that software architecture enables software usability by providing mechanisms for cross-platform compatibility with similar applications, provision of boundary resources for further customisation and through meta-design**

Keywords---Usability, Software Architecture, Design, Configurable Platform.

I. INTRODUCTION

Traditionally, software development has involved two sets of distinct teams: software developers and end-users. Software developers are people with a programming background, involved in the design and implementation of software products. In a traditional software development setup, end-users provide software specifications and wait for software developers to actualise the requirements into the desired software product. Thus, end-users are consumers of the final product from the software developers. This process of development has been deemed costly in terms of human resources, time and money [1]. Good software developers are costly and limited in number. Employing software developers also means that organizations have to deal with an overhead of human resource management issues. Further to this, the process of end-users giving software specifications to software developers and then waiting for the software developers to implement solutions is reported to take time due to differences in priorities between the software developers and end-users [1].

To address problems inherent to traditional software development, organizations are progressively adopting configurable software platforms. Configurable software platforms provide for end-user software development, through customization of software interfaces and behaviour via interaction with graphical user interface (UI) elements and configuration files [2], [3]. Thus, with the advent of configurable software platforms, the implementation of software solutions is shifting from total reliance on software developers towards increased participation of end-users. In turn, providing for end-user development of software applications may lead to reduced software development time and costs [4], [5].

Despite their benefits, configurable software platforms are not without challenges. A key challenge associated with configurable software platforms is that they may have complex interface designs a result of which users may experience usability challenges, which may affect their productivity [6]–[8] User interfaces are a common means through which users interact with software, meaning they are a key to the acceptance of software products. Thus, interface design issues may also negatively impact user experiences for those working with such configurable platforms [3], [7]. As such, to make these interfaces effective for target user groups, it is critical to design them based on principles of human interface design [9]. The three well-known human interface design principles include Jakob Nielsen – 10 usability heuristics for user interface design, Ben Shneiderman – The eight golden rules of interface and Bruce Tognazzini – Principles of interaction design [10].

This paper discusses how software architecture shapes the usability of configurable software in data entry-form design. To achieve this, the paper focusses on the utilisation of configurable software platforms in Malawi's health sector. There is an increased uptake of configurable platforms in the development of patient care and National Health Management Information System (HMIS) solutions in resource-constrained settings such as Malawi, which face a dearth of highly skilled software developers. Notable configurable platforms in use include Commcare and the District Health Information System 2 (DHIS2). Commcare is a mobile data collection platform where a user does not need to write a single line of code. It has features for offline data collection, tracking data over time, incorporating multimedia and multi-language support [11]. DHIS2 is a configurable web software platform developed for the collection, validation, analysis, and presentation of aggregate and patient-based statistical data [12]. It enjoys usage in over 60

countries (Malawi inclusive), most of which are developing countries [13].

The paper uses DHIS2 a primary case, analyzing the experiences of platform end-users in the configuration of data collection forms. Based on noted challenges, the paper compares the DHIS2 platform with other platforms (Commcare, Sketc2Code) in regard to form design, in order to draw lessons for design improvement. The choice of DHIS2 is motivated based on the platform's wide usage in Africa. At the same time, literature shows that the platform can be complex to use. For example, in Uganda, electronic forms for a health commodity ordering system were reported to have use flexibility challenges due to the poor design of the forms as the custom editor in DHIS2 limited the extent to which designers could design forms [14]. Our interactions with colleagues from other countries also show that countries often rely on a small set of experts for configuring national software products, due to platform complexity and usability issues. It is therefore hoped that lessons drawn herein will benefit other countries beyond Malawi.

A. Configurable Software Platforms

A Software platform is defined as a software-based product or service that serves as a foundation on which outside parties can build complementary products [15]. A software platform provides the core functionality of what the platform is expected to do, but it is extendable [8]. To facilitate the extension of functionality, platforms include an interface that allows third parties to develop apps that extend the functionality of the platform [8]. Platform owners concentrate on the development of the platform core and boundary resources, leaving the development of the actual applications in which end-users will interact with the third-party software developers.

Configurable software platforms inherit the properties of software platforms but use the concept of configuration. The word configure means "to arrange how something, such as a computer system or software, is organized, so that it can be used for a particular task"[16]. Thus, configurable software provides end-users with possibilities to 'configure' them to fit the end-users' needs without custom programming. Configurable software platforms achieve this by including one or more configuration utilities that expose the application framework, such that end-users may re-configure the application for any purpose. Application frameworks provide a standard structure through which graphical user interface (GUI) elements can be created as they define the underlying code structure of the application in advance. The application framework takes all the complexities of interfacing with the platform core. In configurable platforms, the importance of making user interface components interactive, usable and flexible cannot be overlooked as it affects users' innovations and design processes through the platform [17].

B. Challenges with Configurable software platforms

Despite the increasing use of configurable software platforms, there are several challenges. One such challenge is platform architecture design [6], [7], [18]. A good platform architecture needs to have four desirable properties namely simplicity, resilience, maintainability and evolvability. But in balancing these properties there are always tradeoffs as some properties are negatively correlated whereby increasing one property decreases another property [18].

Another challenge with configurable software is the process of testing [19]. On top of non-configurable software testing challenges that exist like test case generation, test case selection, and test case prioritization, configurable software platforms add to the list the challenge of testing all possible configurations of the system [19]. Another challenge is the actual development of the software platform to satisfy more than one stakeholder requirement at the same time and uncovering the potential configuration for the software platform to satisfy requirements [3]. Literature has also outlined software quality as another challenge. Software developers cannot always conduct all quality assurance to each possible configuration that users might come up with [20].

Another key challenge associated with configurable software platforms is that they can be complex and users may experience various usability challenges which may affect their productivity [6], [8], [21]. In turn, the complexity and interface design issues in configurable software platforms may also negatively impact user experiences for those working with such platforms [3], [7]. To enhance designers' experiences in working with configurable software platforms researchers must continue to investigate and correct usability challenges associated with various configurable software platforms [22].

C. Software architecture and software usability

ISO/IEC 9126-1 defines usability as the capability of a software product to be understood, learned, used and be attractive to the user, when used under specified conditions. Usability is a core aspect of the system development process to improve and enhance system facilities and to satisfy users' needs and necessities. Usability confirms if a software product has good utility, is efficient, effective, safe, easy to learn, easy to remember, easy to use and to evaluate and provides job satisfaction to the users. Adopting these aspects in the system development process, including the sustainable design will measure and accomplish users' goals and tasks by using a specific technology [23].

Software architecture is defined as a conceptual blueprint that describes how the ecosystem is partitioned into a relatively stable platform and a complementary set of modules that are encouraged to vary, and the design rules binding on both [17]. Until the late '90s, software usability and architecture were taken as separate entities. As such, software usability only focused on the presentation of interfaces elements and functionality which resulted in architecture designers giving it a blind eye when designing software architecture [24]. As more research was conducted on software usability, it was found out that many usability concerns reach deeply into the systems architectural design [24].

Since the architecture is the blueprint for the software which is to be developed, a lack of usability considerations at design time may require extensive and costly re-architecting of software systems, should usability issues be discovered during use[24], [25]. When this happens, projects often cannot afford

the additional cost and ship products that are not as usable as they could be [26]. Examples of usability requirements that are affected by the architecture include the availability of shortcuts, form field validation and recovery from failure.

II. Methodology

The research was conceptualised into three stages. The first stage was usability studies for DHIS2 followed by experimentation with Sketch2Code and Commcare. The final stage was prototyping and evaluation.

A. Process 1 - Usability Evaluation

The research process involved usability studies with 11 users from five organisations who actively use DHIS2. The study participants were required to design a custom data entry form for an HIV Testing and Counselling Combined Quarterly Report. This task was to be accomplished using the DHIS2 custom form editor as shown in Figure II-1.

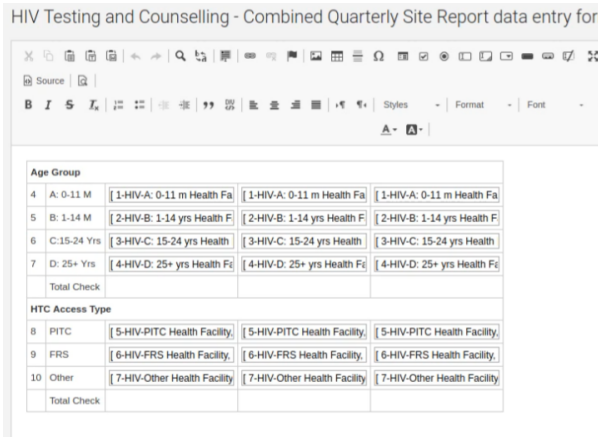


Figure II-1: Designing a data entry from in DHIS2

Study participants were split into two groups (first-time users and experienced users) based on their proficiency in using the DHIS2 custom form editor. The group of first-time users comprised DHIS2 developers or implementers who had not used DHIS2 custom form editor before. Those who had used the editor before were grouped as "experienced users". From the 11 respondents, 6 were experienced users while the remaining 5 were first-time users. Each group was assigned similar tasks to accomplish (see Table *I*).

Table I: Form Design Tasks

Task Number	Tasks For Experienced Users	Task For First-time Users
1	Insert form name	Insert form name
2	Add section headers	Add section headers
3	Create Table with rows for input fields and labels	Create a table on each section to hold the data elements and input fields
4	Add the corresponding data element in each cell	Add form labels for data elements
5	Styling form	Attach the data elements to each form label
6		Format the cell field to have the same width and height

Data were collected using video recordings, observations and interviews. 11 videos were recorded from which data on usability metrics were extracted. The following data were collected: time taken to complete a specified task by a specific designer (in seconds), the number of errors committed during the process, the number of tasks completed by each designer, most used functionalities through the different icons which were used frequently. Structured interviews were also conducted based on Nielsen's 10 heuristics. Notes were taken while observing how the study participants were designing the forms

B. Process 2 - experimentation with Sketch2Code and Commcare

This step was carried out to explore other form editing applications. Two applications were chosen: Microsoft Sketch2Code, and CommCare.

Microsoft Sketch2Code is a solution from Microsoft which uses (AI) to transform hand-drawn user interfaces sketches to valid HTML mark-up code. Sketch2Code brought a new dimension into the research as it involved the use of artificial intelligence which is one of the fields in Computer Science whose applications cannot be overlooked as it has revolutionised the way things are done [27]. Regarding Sketch2Code, the first author conducted experiments with three participants, who were required to design part of the HIV Testing and Counselling Combined Report. On top of the three software developers, software and user manuals (Documentation) were used to get insights into the software. These included Sketch2Code Lab (Online Learning resource on how to use Azure Custom Vision for Sketch2Code.

1) Sketch2Code Sketching process

Before testing with the users, the first author did a sample test of Sketch2Code with the form fields that are on the HIV Testing and Counselling Combined Quarterly Site Report. The first trial started with scanning the form in use to picture format. The scanned copy lost some quality as compared to the original copy. After searching on the internet for a tool that could convert a PDF to an image (.jpg), SmallPDF was found which is an online tool used to convert files into different formats without losing quality. When the generated form was run on the Sketch2Code, the result was weird as the model could not recognise most of the form features from the copy created due to a couple of factors including conformity with the structure of the copy with Sketch2Code.

To experiment with the users, the following procedure was followed. The first step was to enquire from users on how they design their data collection forms for different applications. The

next step was to explain how they could use Sketch2Code to come up with digital prototypes from paper sketches. This involved outlining how they could design the paper sketches to conform to the structure of Sketch2Code.

2) CommCare Evaluation Process

CommCare consists of two components namely CommCare Mobile and CommCare HQ. CommCare Mobile is a mobile-based portion of CommCare used for data collection and service delivery. CommCare Mobile can be used on a phone or tablet and, in rare instances, through a computer. Through CommCare Mobile, a user can access a mobile application. While CommCare HQ is a website that is used for application management and reporting. Through the CommCare HQ website, users can design applications, access data, and manage mobile users. CommCare HQ receives the data submitted by frontline workers using CommCare Mobile.

The evaluation of CommCare was done by designing a similar form to which was designed in DHIS2.

C. Process 3 – Prototyping and Evaluation

For the third stage of the study, three designers were involved of which two were from one organisation. The three prototyped the following: Resizing table rows columns and input fields; and grouping icons on the menu. First, the study participants were asked to design their solutions using paper (Sketches). From their designs, questions were asked on the rationale for their proposed design. From the discussions that ensued, challenges were noted and similar implementations in other applications were researched to deepen our understanding of how we could address noted challenges. For example, on table designing an example of inserting tables in Google Docs was used.

D. Additional Data collection

For all the platforms, the study also reviewed documentation and feedback from user communities for the three platforms. The focus in the documentation was on how the software is designed, its architecture, software licence as well as user manuals.

III. RESULTS

The research findings are presented in three sections based on the three research process phase which were followed: Part 1 - Usability evaluation, Part 2 - experimentation with Sketch2Code and Commcare and Part 3 – Prototyping and Evaluation

A. Part 1 - Usability evaluation

1) Formatting Forms in DHIS2 CKEditor

It was found that the editor did not completely support form styling, at times requiring designers to manually edit the actual source code of forms they had designed. This was done by clicking the *'source'* option on the CKEditor menu. From the source code, designers could edit wherever they wanted to and the changes would then reflect on the forms designed. It should be noted that editing source code requires someone to have a computer programming background which is against the core principle of configurability, where the goal is that designers should create their solutions on using a configurable platform without the need for programming skills.

2) Challenges with Table Dimensions

It was found out that tables had an auto fixed width by default, which could be changed by specifying the size in pixels after right-clicking a table. When one wanted to add a data element to a specific cell in a row, other cells in the row shrunk in size, which made it difficult for designers to select shrunk cells. Based on experience working with other editors, designers tried to increase the width and height by clicking and dragging the cell borders, but to their surprise, nothing happened.

3) Formatting forms

To circumvent form editing limitations such as above-mentioned, some designers reported that they use Microsoft Excel or LibreOffice Calc spreadsheet applications to design forms, after which they copy and paste the designs in DHIS2's CKEditor. The rationale for choosing these applications was that they offer the flexibility to merge cells, hide columns, resize cell width and height.

B. Part 2 – Comparisons between different configurable platforms that aid inform design

This section presents findings on how the three selected configurable platforms compare in aiding form designing. These three tools include DHIS2, Sketch2Code and CommCare. A comparison was done in terms of design flexibility, software licensing, available functionality and complexity of technologies in use.

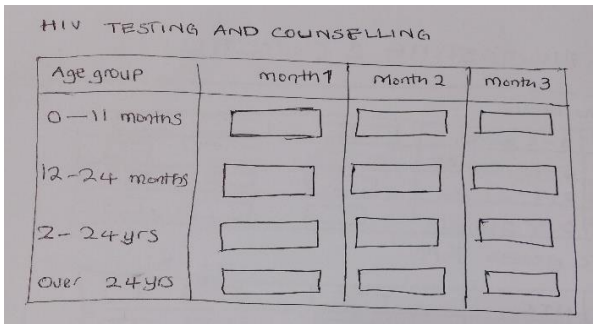
1) Findings on Sketch2Code

Most of the time when software developers want to design data entry forms, the first steps involve designing the form on paper as it helps them to discuss and gather requirements as well as agree on the design before they start the actual development. Then they write HTML (for structure) and CSS (styling)) code as a way of translating the agreed paper sketches into first digital prototypes of the form. When transforming paper sketches with the AI different results were produced. Going through each generated output, it had a well-defined HTML structure (elements, attributes) and bootstrap 4, a CSS framework was added as well to the file. Figure III-1 shows a paper sketch and an HTML form generated by Sketch2Code. During the designing and testing process three constraints were noted:

- The image quality of sketches affected the result of sketches generated by Sketch2Code.
- The environment in which pictures of sketches were taken affected the quality of sketches (lighting, shadows, etc.).

- Sketch2Code has its defined structure of icons and options which it can recognise, implying when designing sketches, designers had to conform to that standard syntax. It was noted that if the sketches did not conform to the standards one would get undesired results

Paper Sketch



Sketch2Code

HTML Form

Figure III-1: Paper Sketch to HTML form

A notable positive aspect with Sketch2Code was that HTML source code generated by the platform could be imported into the DHIS2 CKEditor, created largely well-structured forms.

2) Form Designing Process in CommCare

As has been outlined in the section above, to design the form for data collection one uses the CommCare HQ app. One needs to register first before they can create their application. To create a form, one has to specify an application name and whether they want to collect data as a survey (Collect data once) or as a Case list (Track items over time). After selecting one's preference, one can add questions to their survey or case list.

When adding a question one needs to specify the datatype of the values that will be collected on a specific data element and those options are presented when ones click on the Add Question button. After selecting a preferred data type, one is required to fill in details for the Display Text/Label, Question ID and indicate whether the question is mandatory or not. To the right of the designing window, is a phone simulator where designers can log in and test the look and feel of the form being designed.

3) CommCare vs DHIS2

Much as both platforms are used for creating and designing data entry forms, there are differences in terms of how they handle some activities. The differences are on how one can design a form on both platforms, the target devices for the designed forms and preview function.

When creating a data entry form in DHIS2 one has to go through the maintenance app, to create a data element and then create a data set, after which they can proceed with designing a form. Thus to get to the point of designing a form, one has to go through three app interfaces. In Commcare, everything is done within the same window. Literature shows that presenting data and information on one page increases systems efficiency [28].

The target device for the final usage of the form for CommCare is a mobile phone which has limited screen size for while DHIS2 custom forms are rendered only on a computer and not on a mobile application which renders only default forms. As such the focus when designing in CommCare is the order in which the variables will appear and the logic behind. For DHIS2, the designer often designs custom forms with the thought that the forms will be used on a computer with a large screen and it has to be presented on a single window.

Further to the above, in CommCare one can easily preview the form being designed using the phone simulator and test the logic and the interaction before you publish it (see Figure III-2). DHIS2 has the preview option but it does not give the look and feel of the final product and you cannot add data to it. The only way to see what you have done if they work is to save it and go to the Data Entry app to see the changes.

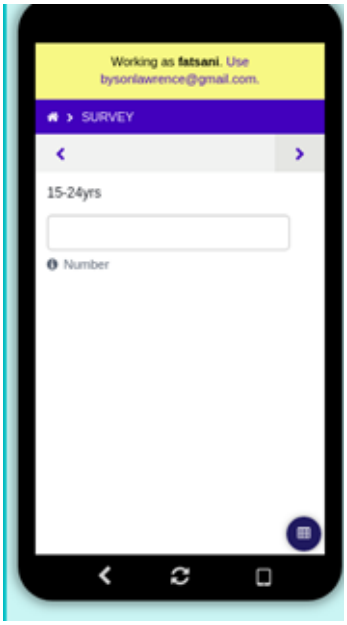


Figure III-2: Preview of a form being designed in CommCare

C. Part 3 - Prototyping and Evaluation

After looking at the different tools for designing data entry forms, the first author came up with prototypes to discuss with designers as a way of designing with users. The focus was on the following tasks: resizing rows, columns and input fields, dragging and dropping data elements and grouping of icons on the menu were prototyped. Both paper and software prototypes were created on this stage.

1) Paper Prototypes

Resizing rows, columns and input fields and grouping icons on the menu were paper prototyped.

a) Resizing rows, columns and input fields

Most of the challenges which are faced with the DHIS CKEditor editor are to do with form formatting. For example, to reduce or increase cell width, designers have to go to the source code and add styling to the code. When asked on how this can be implemented, two solutions were brought forward, the first one was to allow users to edit both the height and width of a cell. While the other was of the view to resize the width only as there are rare cases in which they edit the height of rows, columns or input fields.

b) Group icons on the menu

On this part, different options on how to present the grouping of our menu icons were presented as categories. Presentation of menu options as icons was preferred by everyone with a suggestion to display tooltips on the icons when one hovers over an icon.

2) Software Prototype

After the above experiments, a software prototype was developed. On the menu, most of the icons were removed to only remain with those that are applicable to form designing to maintain a minimalist design and not to clutter the screen with unused components.

As a way of removing confusion in the handling of rows and columns, the insertion of tables was implemented similar to most word processors like Google Docs, Microsoft word where there is a visual representation when inserting a table. Instead of entering the number of rows and columns, the designer hovers over graphics which symbolise a table with rows and columns.

Another element that was developed was the drag and drop functionality. In this solution, the designer does not have to type the form field name as it was witnessed during the usability studies. This solution automatically picks the data element name which was defined in DHIS2 as the default name and provides the possibility of editing it. This reduces the time spent in entering the form labels when creating the custom forms.

IV. Discussion

A. Enabling factors of usability

Usability is defined as the degree of a product's (in this case a software application) potential to accomplish the goals of the user [30]. It is also defined as the ease of use and learnability of application software for the end-user [6]. Relating these two definitions to how designers designed the forms in DHIS2, the available interface design elements made the editor usable as all the designers were able to achieve the goal of designing the form. The built-in capabilities of the editor enabled designers to customise the forms which agree with [27] and [31] findings on the extent to which DHIS2 allows customisation flexibility as an enabling factor.

In the software platforms that were investigated, HTML + CSS code generated in Sketch2Code when used in DHIS2 editor was able to create a structured form similar to the one generated in Sketch2Code. Also, experienced designers reported using other software to design forms like Microsoft Excel, Google sheets to design form and only to paste the designed forms into DHIS2. These different workarounds attest to Li's (2018) findings that the existence of a workaround happens when the technical design fails to meet established work routine and contextual conditions as DHIS2 custom form editor has design flaws.

B. Constraining factors of usability

The usability of the DHIS2 custom form editor application was negatively affected through limited functionality that different the editor provided. In a configurable platform, the goal is to equip the designer with all tools which will enable them to create a solution without the need for a software developer [34]. DHIS2 custom form editor was not able to provide all the required functionality on the user interface like changing cell width as such designers were forced to change by specifying pixels which was challenging to designers who do not have a programming background. This agrees with the findings of [35] where he pointed out that the DHIS2 custom form editor provides a minimum fit between the system interface and the desired forms due to the limited functionality of the editor. In configurable platforms, users are supposed to create applications from the resources that are available on the user interface without having to twerk the source code to get desired results. This usability challenge could be argued to stem from the constraints of software architecture design

Still on architecture, Sketch2Code and DHIS2 custom form editor provide a set of predefined functions or constructs which the designer chooses when designing. Thus, if the structure of the form field does not conform to the predefined design the designer will have to redesign the form which frustrates the user and affects productivity [36]. Available functionality in the form design editors centres on the assumption that the designer has a programming background as witnessed in DHIS2 where technical terms are used as well as the need for writing code to tweak form User Interface. In Sketch2Code also, the whole process assumes that the one doing the design has a programming background.

C. Software architecture design in shaping usability

Software Architecture specifies how a software system should be organized and the overall structure of that system. It identifies the main structural components in a system and the relationships between them [9].

1) Cross-platform compatibility

Cross-platform compatibility focuses on developing an application/system which can work seamlessly across other platforms. The capability of cross-platform compatibility is dictated in software architecture as it provides the constraints and enablers for compatibility. DHIS2 allowed elements defined in other platforms to run inside it due to the flexibility of the architecture in providing cross-platform compatibility. This enhances productivity among designers as they can still achieve their goal of designing a form in DHIS2 for data collection even though they have to use different platforms and integrate the results.

2) Predefining elements

Meta-design is focused on objectives, techniques, and processes that allow users to act as designers. As such the software architecture should provide constructs on which designers can design their innovative solutions. CommCare provides a predefined list of elements as input fields. While in DHIS2, one does not specify the type of input field that one wants when creating a form, it is picked automatically based on how the data element was defined.

3) Boundary resources

Boundary resources provide mechanisms on how software application developers can extend functionality and improvements on existing applications. DHIS2 has an API that gives access to resources within the platform core. The architecture provides flexibility for customisation which further echoes what literature says on how the platform architecture shapes usability through provision on room for customisation [35], [37]. This enabling factor enabled the research to come up with a prototype. The created prototype used the API to get data elements from DHIS2. The data element name was automatically added to the form as a label when a specified data element was selected in the designed editor. This improved the time taken to design the forms as with the built-in CKEditor designers have to rewrite names of data elements or form fields on form designs, instead of picking the default name which was given to the data element as it was created.

V. Conclusion

Software architecture affects the usability of the software. Cross-platform compatibility provides users with the flexibility of achieving their goals regardless of the platform they are working on provided that what they will design will also run on other platforms. Also, the availability of boundary resources encourages development and innovation beyond a configurable platform's core developers. As such different software developers can create workarounds on how they can improve usability through apps and functionalities which can be used by designers thereby enhancing the usability of the platform.

VI. References

[1] S. W. Zhang and Z. H. Li, 'A Configurable Platform of Application System and its End-User-Oriented Configuration-Developing Pattern', *AMR*, vol. 219–220, pp. 1415–1418, Mar. 2011, doi: 10.4028/www.scientific.net/AMR.219-220.1415.

[2] J. Toman, 'Learning to Adapt: Analyses for Configurable Software', University of Washington, 2019.

[3] A. Misaka, 'Requirement analysis technique for configurable platform : case study', Master of Applied Science, Carleton University, Ottawa, Ontario, 2013.

[4] M. Carr, 'Configurable software solutions— Change is good, right?', *Locus Technologies*, Aug. 25, 2016. https://locustec.com/blog/configurable-software-solutions-change-good-right/ (accessed Apr. 22, 2019).

[5] L. P. Herman, 'Usability and use documentation in a health information system: The case of District Health Information System 2 in Malawi', University of Oslo, 2016.

[6] L. Battle and L. Chessman, 'Designing Configurable and Customisable Application', presented at the UPA Conference, Jun. 06, 2012, Accessed: Aug. 27, 2019. [Online]. Available: https://www.designforcontext.com/insights/designing-configurable-and-customizable-applications.

[7] G. Fischer and E. Scharff, 'Meta-design: design for designers', in *Proceedings of the conference on Designing interactive systems processes, practices, methods, and techniques - DIS '00*, New York City, New York, United States, 2000, pp. 396–405, doi: 10.1145/347642.347798.

[8] A. Tiwana, *Platform Ecosystems Aligning Architecture, Governance, and Strategy*. Elsevier, 2014.

[9] J. Morrison, 'Interface Design & Usability', *netzstrategen*, Nov. 22, 2018. https://netzstrategen.com/koennen/user-experience/interface-design-usability (accessed Apr. 07, 2019).

[10] B. Birch, '10 principles that form my user interface design strategy', *Together Incredible | Improve the digital experience*, Jan. 03, 2019. https://togetherincredible.com/10-principles-that-form-my-user-interface-design-strategy/ (accessed Apr. 22, 2019).

[11] Dimagi, 'CommCare by Dimagi | Data Collection App', 2019. https://www.dimagi.com/commcare/ (accessed Jun. 23, 2019).

[12] dhis2, 'Overview', Jan. 01, 2018. https://www.dhis2.org/overview (accessed May 20, 2018).

[13] DHIS2 Documentation Team, 'DHIS2 Implementer guide', Aug. 23, 2018. https://docs.dhis2.org/2.30/en/implementer/html/dhis2_implementation_guide_full.html#d0e2872 (accessed Aug. 23, 2018).

[14] M. Li, 'Utilising the Space for User Participation', University of Oslo, 2018.

[15] A. Ghazawneh and O. Henfridsson, 'Balancing platform control and external contribution in third-party development: the boundary resources model: Control and contribution in third-party development', *Information Systems Journal*, vol. 23, no. 2, pp. 173–192, Mar. 2013, doi: 10.1111/j.1365-2575.2012.00406.x.

[16] 'CONFIGURE | meaning in the Cambridge English Dictionary'. https://dictionary.cambridge.org/dictionary/english/configure (accessed Apr. 22, 2019).

[17] A. Tiwana, B. Konsynski, and A. A. Bush, 'Coevolution of Platform Architecture, Governance, and Environmental Dynamics', *Information Systems Research*, vol. 21, no. 4, pp. 675–687, Dec. 2010, doi: 10.1287/isre.1100.0323.

[18] A. Tiwana, 'Platform Architecture - an overview | ScienceDirect Topics', 2014. https://www.sciencedirect.com/topics/computer-science/platform-architecture (accessed Oct. 30, 2019).

[19] X. Qu, 'Testing of Configurable Systems', in *Advances in Computers*, vol. 89, Elsevier, 2013, pp. 141–162.

[20] M. Große-Rhode, R. Hilbrich, S. Mann, and S. Weißleder, 'Achieving Quality in Customer-Configurable Products', in *Relating System Quality and Software Architecture*, Elsevier, 2014, pp. 233–261.

[21] G. Fischer, 'Symmetry of ignorance, social creativity, and meta-design', *Elsevier Science B.V.*, vol. 13, no. 7–8, p. 5, 2000.

[22] M. de Reuver, C. Sørensen, and R. C. Basole, 'The Digital Platform: A Research Agenda', *Journal of Information Technology*, vol. 33, no. 2, pp. 124–135, 2017, doi: 10.1057/s41265-016-0033-3.

[23] H. M. K. Abdoasslam, 'Measuring Usability For Application Software Using The Quality In Use Integration Measurement Model', Universiti Tun Hussein Onn, Malaysia, 2016.

[24] B. E. John and L. Bass, ‘Usability and software architecture’, *Behaviour & Information Technology*, vol. 20, no. 5, pp. 329–338, Jan. 2001, doi: 10.1080/01449290110081686.
[25] I. Sommerville, *Software Engineering*, 10th ed. England: Pearson Education Limited, 2016.
[26] B. E. John, N. Juristo, L. Bass, and M. Sanchez-Segura, ‘Avoiding “We can’t change THAT!”: Software Architecture & Usability’, p. 104, 2004.
[27] R. Khanna, ‘How is Artificial Intelligence changing the Manufacturing Industry in 2018?’, *Ishir*, Jul. 05, 2017. https://www.ishir.com/blog/4654/artificial-intelligence-in-manufacturing-industry.htm (accessed Apr. 19, 2019).
[28] Microsoft, ‘Microsoft AI lab’, Aug. 23, 2018. https://www.ailab.microsoft.com/ (accessed Nov. 20, 2018).
[29] B. Hellard, ‘Microsoft’s AI-powered Sketch2Code builds websites and apps from drawings’, *Alphr*, Aug. 30, 2018. https://www.alphr.com/go/1009840 (accessed Nov. 20, 2018).
[30] J. Melin, ‘Making data useful to health workers by increasing usefulness and usability of their tools An experiment to increase health workers ability to detect symptoms of health issues with the District Health Information System (DHIS2) Tracker Capture Android app’, University of Oslo, Norway, 2018.
[31] S. Krug, *Don’t Make Me Think*. United States of America: New Riders, 2014.
[32] Kirakowski, J. and McNamara, N, ‘Functionality, Usability, and User Experience: Three areas of Concern.’, *Interactions*, vol. 13, no. 6, pp. 26–28, Nov. 2006.
[33] Z. Ismanov and I. Ni, ‘Patient information system for specialized newborn care units in Malawi Mobile implementation of DHIS2 Tracker in neonatal hospital wards’, University of Oslo, Norway, 2018.
[34] N. Hansson and T. Vidhall, *Effects on performance and usability for cross-platform application development using React Native*. 2016.
[35] M. Wäljas, K. Segerståhl, K. Väänänen-Vainio-Mattila, and H. Oinas-Kukkonen, ‘Cross-platform service user experience: a field study and an initial framework’, p. 10, Jul. 2010.
[36] A. A. Gizaw, B. Bygstad, and P. Nielsen, ‘Open generification’, *Information Systems Journal*, vol. 27, no. 5, pp. 619–642, 2017, doi: 10.1111/isj.12112.
[37] R. A. Majid, N. L. M. Noor, W. A. W. Adnan, and S. Mansor, ‘Users’ Frustration and HCI in the Software Development Life Cycle’, *IJIPM*, vol. 2, no. 1, pp. 43–48, Jan. 2011, doi: 10.4156/ijipm.vol2.issue1.5.
[38] L. K. Roland, T. A. Sanner, and E. Monteiro, ‘Architectures of large-scale participatory design’, *Scandinavian Journal of Information Systems*, vol. 29, no. 2, pp. 3–33, 2017.