How to cite this article:

Rukhiran, M., & Netinant, P. (2020). A practical model from multidimensional layering: Personal finance information framework using mobile software interface operations. *Journal of Information and Communication Technology, 19(3),* 321-349.

A PRACTICAL MODEL FROM MULTIDIMENSIONAL LAYERING: PERSONAL FINANCE INFORMATION FRAMEWORK USING MOBILE SOFTWARE INTERFACE OPERATIONS

Meennapa Rukhiran & Paniti Netinant
College of Digital Innovation and Information Technology, Rangsit University,Thailand

meennapa_ru@rmutto.ac.th; paniti.n@rsu.ac.th

ABSTRACT

End user involvement is crucial in improving software development processes. Hence, nowadays user interface (UI) and user experience (UX) are particularly concerned with end user interactions in many software designs as most methodologies have inconsistencies between design and implementation. Besides, it is relatively difficult to make changes in complex software and personal finance application is one of the more complex software to design, develop, and adapt. This paper proposes the development of a mobile personal finance application using informative multidimensional layering. We have separated functional data cutting across the relationships of three categories and datasets showing operational semantics of dimensions, and combined layers of three-dimensional information including aspect elements through components. This study is concerned with the corresponsive composition of end user features using visual interfaces. It is illustrated in a Three-layer User Interface Composition Model to transfer and compose layers, functional data, aspect elements, and components to Graphical User Interfaces (GUIs). Therefore, an integrated view of the software system would make the design and implementation consistent to support our framework in a more straightforward manner.

Received: 20/3/2020 **Revised:** 3/3/2020 **Accepted:** 20/3/2020 **Published:**11/6/2020

There have been a few studies which presented practical models of mobile informative multidimensional layering. This research applied aspect orientation and informative multidimensional layering to present a better features model for mobile personal finance application. We deliver a practical framework in the application in all four phases of analysis, design, implementation, and evaluation. In addressing the gap, this research proposes a clearer operation of three-dimensional models, functional data, and aspect elements that cut across through informative multidimensional layering.

Keywords: Functional data, multidimensional data, mobile, software, user interface.

INTRODUCTION

User Interface (UI) plays an important role in software development (Sadowski & Zimmermann, 2019). Leach (2016) presented six primary activities (requirements, design, coding, testing and integration, delivery and maintenance) that a team of developers should devote to a software development life cycle. User interfaces are discussed in the analysis process. By accepting document software needs, the requirements of UI designs must include a layout plan of menus, screens on a software system, and a requirement traceability matrix. The process of UI design is specified after the creation of process modeling, data modeling, and architecture design are conducted. These basic designs has led us to comprehend the relationship of functions, processes, data flows, hardware, software, and infrastructures (Dennis, Wixom, & Roth, 2012) and to manage an end user information system involving users (Tonder & Wesson, 2012; Usoro, 2013). Lastly, the interface design illustrates how end users can use the software application. Another principle which improved on Graphic User Interface (GUI), and User Experience (UX) design was first introduced in 1990. UX is defined as a person's perception in response to the use of a product (Hinderks, Schrepp, Mayo, Escalona, & Thomaschewski, 2019). The end user experience has led designers and developers to identify consumer satisfaction in the process. In the software development industry, the experience of a good user refers to an evaluation of an end user interface that is easy to learn and efficient to use (aesthetics, joy of use, and attractiveness). However, UI design management is highly concerned that developers spend more time designing UI (Desolda, Ardito, Costabile, & Matera, 2017; Kennard & Leaney, 2010). Hays (2014) claims that complex user interfaces can encounter a delayed reaction time and unsatisfied users. Moreover, sophisticated user interfaces can reduce productivity (Sadowski, & Zimmermann, 2019). Hence, we believe that

paying attention to designing UI according to the users' needs has tremendous advantages in software progress.

It is crucial to focus on developing a prototype software life cycle. All phases of the software project are established from important information and functional requirements of stakeholders. The needs of information requirements, changes, operations, and representations are attributes of software quality and the major concerns of collaborative access through a software system (Jallow, Demian, Anumba, & Baldwin, 2017). These concerns has led us to the initial research of designing multidimensional layering for supporting functional data and aspect elements of a house bookkeeping software design (Rukhiran & Netinant, 2017a). Our personal finance application design is based on three-dimensional layering. The layers have the coordinates of X, Y, and Z, that is, an axis belongs to expenditure, income, and liabilities, respectively. Each dimension is separated into relative subdimensions. We define functional data as a correlative relationship of the system information that cuts across in the layering of three dimensions. Moreover, we have applied the principle of separating concerns for identifying aspect elements. An aspect element is a group of crosscutting functionalities. We can find these aspects repeatedly in the processes of software development and then group them together. This is called a set of aspect elements.

While many research have been continuously applying Aspect-Oriented Software Development (AOSD) in order to achieve a more effective and efficient approach, we have rarely found that the principle of separation of concerns is applied from the first analyzed phase till the end user review phase. One disadvantage of the software design is that sometimes the UI is implemented separately and explicitly from the software modeling (Jelinek & Slavik, 2004). Kennard and Leaney (2010) observed that any phases in designing need to be consistently, concerned. Moreover, our previous work had proposed the concept design of separating concerns (Rukhiran & Netinant, 2017b). Our current research concentrates on the challenge of a fine granularity information design that is a significant design concern to define a set of data, functional data, and aspect elements of layers. The important research is how to develop a framework that can practically, simply, enthusiastically, and aesthetically operate information, functions, aspects, and layers with higher and better separation of concerns. We present a prototype of the software development life cycle in all four phases of analysis, design, implementation, and evaluation. Therefore, to address the gap, this research will bring about more clearly, operations of the three-dimensional model and functional data and aspect elements that cut across the dimensions. In areas of visual user interfaces, the corresponsive design of an end user composition is the cooperation between layering of dimensions and separating concerns of each section. The interactive application design is based on an outcome of composition sections at a weaving time. These concerns has led us to challenge

the development of informative multidimensional layering for supporting semantic operations, representations, and separation of concerns from design to mobile software interface operations. We will present the integrated views to support a higher and better composition of interface operations in which a personal finance information application is intended to make the system design and interface consistent with the separation of concerns in various aesthetic interface operations. Consequently, the Aspect-Oriented Approach (AOA) seems to fully support the UI and UX as proposed . Besides, the evaluation of GUIs to handle flexibility, efficiency of use, aesthetics and minimalist design is enabled by applying different sections (layering) on the screen.

BACKGROUND

This section provides an overview on a separation of concerns, a survey of a major concept using separation-based UI, aspect-oriented approaches, multidimensional layering, and previous research in designing and developing a personal finance information framework. We emphasize challenges to an aspect-oriented approach for refining attributes of software quality and formerly proposed solutions. We have defined the semantic operations of informative three-dimensional layering among functional data, aspect elements, and layering of the execution design stages in this research contribution.

Separation of Concerns

Separation of concerns is defined as a key principle of software design and implementation (Panunzion & Vardanega, 2014a; Panunzion & Vardanega, 2014b). Basically, a concern is divided as a part of the software that represents a single functionality. To handle the separation of concerns, an aspect orientation is approached through new abstractions and composition mechanisms (Kiczales et al., 1997; Netinant & Elrad, 2016). The principle of the AOSD is to augment modularizations of crosscutting concerns (AI-Hudhud, 2015; Tanter, Figueroa, & Tabaerau, 2014). The concerns can be called by a component, which is dependent on a weaver. Weaving is the process of systematizing aspects and other elements (AI-Hudhud, 2015; Jelinek & Slavik, 2004). The evolution strategy of AOSD focuses on expressing the rules and definition of events, conditions, and actions for supporting changes in computation environments (Zhang & Rong, 2009). The dynamic evolution is concerned with a running time. The first rule is an addition of a base component. The second rule is an addition of an aspect component. The third rule is an addition of an aspect connector. The fourth rule is an addition of attachments. Therefore, the separation of concerns can result in a reusable, extensible, and adaptable system (Diaz, Romero, Rubio, Soler, & Troya, 2005).

Separation of Concerns in UI

Modularity is a fundamental concept of separating software modules using components. A component aims to design data and functions for additional restrictions (Barricelli, Cassano, Fogli, & Piccinno, 2019) that can push forward from design to code. The final result transfers to the end users using specific interfaces. Kennard and Leaney (2010) proposed the concept of a large architectural design. A clear separation between layers should be considered and the design should not lead us to a disorganized model. The separation of concerns, including contents, applications, and devices should be decoupled without any limitations in end user interactions. Thus, the user is able to focus only on reaching information (Latizina & Beringer, 2012). Gibbs, Dascalu, and Harris (2015) presented a separation-based UI for role specifications. They proposed an architecture of a separation based UI diagram using Domain Specific Language (DSL) to connect to the UI and codes for improving the flexibility of software platforms, frameworks, and tools. By applying the separation of concerns, a composition paradigm is developed to manipulate at different levels (Ardito et al., 2015). The presentation layer of UI design illuminates in supporting many device platforms and user levels. Mirbel and Rivieres (2003) focused on separating views of UI and Business Domain (BD). The article enables one to draw an application model dividing it into two views as mentioned. UML dependencies and UML actions relating to the design are specially provided. Therefore, the association between UI and BD supports the analysis and design phase of the software development beneficially.

Aspect-Oriented Approach

The challenges in software design projects led us to develop higher quality attributes throughout the software development life cycle (Silveira, Cunha, & Lisboa, 2014). The separation of concerns was applied for many reasons, such as to reduce complexity, improve modularity, to enable compensability, extensibility, reusability, and adaptability (Diaz et al., 2005; Pekilis, 2002; Raheman, Maringanti, & Rath, 2018). The separation of concerns delivers the principle of designing and programming paradigms of an aspect-oriented approach using the execution of weaving instead of calling the functionalities directly, to design individual concerns, including in programming languages (Sommerville, 2014) such as class, method, and procedure, etc. The modularization is improved by defining new constructions. The encapsulation of crosscutting concerns is divided into single modules named aspects. The aspect elements are the smallest functions that can be cut across a code program. Hoffman and Eugster (2008) posited the ability of aspects that are not only the separation of concerns but the modularization is transformed into reusable components. The goals of the design are to reduce coupling and

increase cohesion by counting the number of modules named as pointcut. A pointcut is defined as a state selection of particular joint points. The separation of concerns in designing mobile software was proposed by Netinant and Elrad (2016). A Communication Closed Layer (CCL) provides the implementation of an aspect-oriented approach for avoiding code tangling. The process of the layer can support a clean integration between the components (processes) and composition layer software. An aspect-oriented approach has two principles: (1) to decompose a software system into a group of aspects known as concerns (Butting, Eikermann, Kautz, Rumpe, & Wortmann, 2019; Kumar, Kumar, & Iyyappan, 2016) and (2) to compose crosscutting concerns between aspects and core modules using a weaving process of a joint point (Muck & Frohlich, 2014).

Review on Multidimensional Layering

A multidimensional system is captured via linear transformations by D'Andrea (1999). A set of multinomial functions can handle systems with many inputs and outputs, equations and operators. The set of functions performs through a multidimensional system. Pedersen and Jensen (1999) stated that multidimensional layering deals with complex data. The multidimensions represent a set of categories with as many relationships as the dimensions along with the hierarchical presentations. The layering strategy is one of the decomposition techniques for a software engineering solution, such as to comprehend a complex software system and to solve different perspectives from different audiences (Eeles, 2001). The primary strategy of the multidimensional layering is for software reusability and maintainability. The layering influences structures of software models. A layered multidimensional modeling was described by Boukraa, Boussaid, Bentayeb, and Zegour (2013) for supporting complex entities. The entity is separated from the UML classes. The set of classes can be composed of a whole conceptual entity using layers. The layer is used to share the same entities (objects). The layer of data cube provides details of structures in each dimension that can be called orderly. In the field of an aspect-oriented approach, Multi-Dimensional Separation of Concerns (MDSC) is widely used for software architecture (Lin-lin et al., 2008). MDSC allows for separation of concerns to execute multidimensions and to refine them into concerns, simultaneously. This dynamic ability can address new concerns and configure relationships between components without changing the behaviour of the system.

Early Studies on House Bookkeeping Software Design Using Aspect-Oriented Approach

Our recent AOSD is designed to support house bookkeeping software by separating functional data from the aspect elements. In this recent work, we

proposed functional data in a three-dimensional layering to present relationships among sets of data. There are three dimensions (income, expenditure, and liabilities) divided into a series of data concerns. Each dimension is categorized into smaller datasets shown in our latest work (Rukhiran & Netinant, 2017a). The data of house bookkeeping is divided into three concerns as shown in Table 1.

Table 1.

Set of Data for House Bookkeeping and Personal Finance

Dimension	**Sub-dimension**	**Functional data**
Income (I)	Earned Income (EI)	working, owning a business, consulting, gambling
	Portfolio Income (PoI),	trading paper assets, selling real estate, investment
	Passive Income (PaI)	rental income, bonus, insurance, retirement, interest, bank interest, stocks
Expenditure (E)	Daily Expenses (DE)	food, transportation
	Personal Expenses (PE)	clothing, travel, sports, books, social & entertainment
	House Expenses (HE)	mobile phone bill, Internet, repairing equipment, parking fee
	Family Expenses (FE)	tuition fee, alimony, medical fee, donation
Liabilities (L)	Current Liabilities (CL)	credit card debt, home equity loan, interest, taxes, rental mortgage
	Long-term Liabilities (LL)	bonds payable, notes payable, bank loan, deferred revenue, mortgage

The functional data initially records from one field to n fields in table names. Hence, whenever there is a set of n-tuples we let the functional data set Functional Data $\cup \{F_1, F_2, F_3, \ldots, F_n\}$. The aspect elements are defined as a set of computational properties (e.g., insert, update, delete, day, month, year, and total) which starts corporately from more than one aspect to m

aspects. The aspect element is a sequence of methodologies from 1 to m. The aspect element sets Aspect Elements ∪ $\{A_1, A_2, A_3, \ldots, A_m\}$. An object is an execution of calling the aspect elements and the functional data using crosscutting concern at a higher level. We assume a weaver to call the object of the final execution by using the functional formula n x m for crosscutting concerns shown in Figure 1. Weaving is the process of transforming to solve the scattered solutions and avoid tangled methodologies.

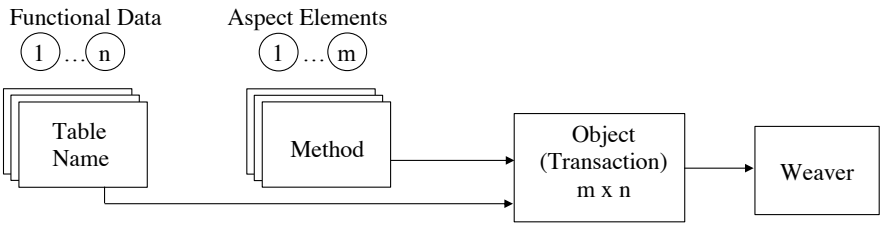

Figure 1. Execution design of aspect-oriented approach.

Composition in Aspect Orientation

The main function of an aspect-oriented technique is to decompose concerns and methods into aspects. The aspects can be cut across functional components, and composing components and aspects to execute in an implementation phase. A composition is defined as an act of putting the various elements together (Oxford, 2011). Pekilis (2002) stated that the composition of correspondences within a particular software component or system that the construction depends on designing formalism. The composition rules are to integrate relationships of abstract declarations and implementations of base elements (classes), crosscutting elements (aspects), and crosscutting relationships (classes and aspects) (Silveira, Cunha, & Lisboa, 2014). Rosenmuller, Siegmund, Thum, and Saake (2011) focused on the multi-dimensional variability model which is an extension of the MDSC design of Tarr, Ossher, Harrison, and Sutton (1999). Examples of a variability dimension are: the execution environment of a program (e.g. operating system and hardware), the context at a running time (e.g. time, space and user), and non-functional properties (e.g. security and quality of service). There are three alternative mechanisms (inheritance, superimposition and aggregation) to compose the variability model for a clean separation of variability dimensions. This composition technique can be highly reused modules.

MULTIDIMENSIONAL LAYERING DEFINITION

Separation of Data, Function and Layers

A new idea of the dimensional layering is to describe the sets of data divided into three concerns. The layers consist of concerns which have the coordinates of X, Y and Z, that is, an axis belongs to expenditure, income, and liabilities, respectively. Each dimension is separated into relative subdimensions. The composition of the sets of data can represent the dataset of personal finance for data execution and management. In Figure 2, the data of personal finance can express an infinite series from 1 to n. In fact, the column name containing sets of data relates to an infinite number of planes for supporting three-dimensional coordinate layering. The Cartesian product is generalized across three categories of datasets. The one-dimensional layer is a line of each axis. For the formal notation of one layer, we set the formal notation = {{I}, {E}, {L}}. For example, the Cartesian product of an income is denoted by I. The two-dimensional layer is a coordinate plane between two axes. For the formal notation of two layers, we set the formal notation = {{I, E}, {I, L}, {E, L}}.

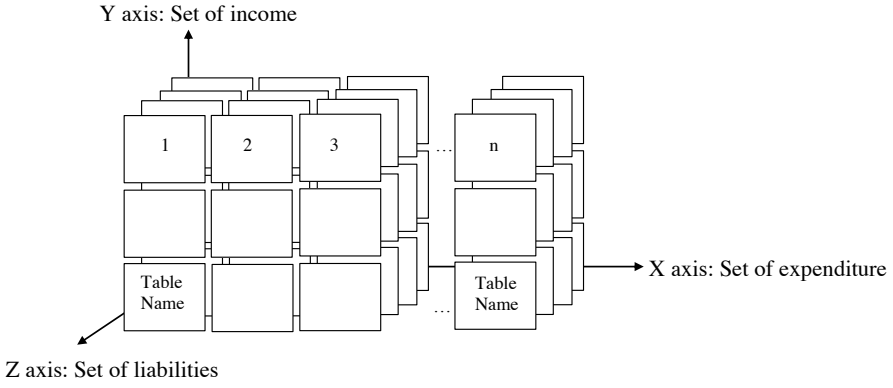

Figure 2. Infinite series of a set of house bookkeeping dimensions.

For example, the Cartesian product of an income record and an expenditure record is denoted by I x E. We set the cross product I x E = {(i, e) | i ∈ I and e ∈ E}, I x L = {(i, l) | i ∈ L and i ∈ L}, E x L = {(e, l) | e ∈ E and l ∈ L}. The three-dimensional layer is a coordinate plane among three axes. For the formal notation of three layers, we set the formal notation = {{I, E, L}}. For example, the Cartesian product of an income record, an expenditure record and a liabilities record is denoted by I x E x L. Thus, the concept of Cartesian product can be extended to more than three sets. We define the concept of

an ordered n-tuple. The order b-tuple is a set of n categories that we have divided as a subdimension of each dimension. We express D as a dimension using the set notation D = $\{D_1, D_2, D_3, \ldots, D_n\}$. Income = $\{I \in D \mid I$ is a set of income records$\}$. Expenditure = $\{E \in D \mid E$ is a set of expenditure records$\}$. Liabilities = $\{L \in D \mid L$ is a set of liabilities records$\}$. The functional data is a Cartesian product from one dimension to three dimensions. In addition, the aspect element is a sequence of methodologies from 1 to m. An aspect sets $A \cup \{A_1, A_2, A_3, \ldots, A_m\}$. For all crosscuttings of the dimensions and aspects, the function formula n X m is the formulae. Thus, the design can support the multidimensional layering among its dimensions.

SEMANTIC MULTIDIMENSIONAL LAYERING

The multidimensional layering is described as the sets of data divided into three concerns. The sets of data can represent the composition of dimensional layering with the different information. The functional data is decomposed from the information input. Figure 3 shows the formulation of cutting points on three-dimensional layering. The layering has provided three different semantics: an income layer, an expenditure layer, and a liabilities layer. The layer provides the appropriate contextual information for data manipulation. Each dimension consists of a set of multi-layers. For example, the y-axis of an income layering, Income = $\{I_1, I_2, I_3, \ldots, I_n\}$, refers to one layering of subdimensions which is divided from a user's data categories (e.g. a passive income and an earned income). There are two types of quantifiers to express the formal notations for computing functional data from datasets. The universal quantifier $(\forall)$ is for a selection of all income records from a layering. This is expressed in Equation 1 as follows,

$$\forall \text{income}, \text{income} > 1 \tag{1}$$

The existential quantifier $(\exists)$ is for some income records in the universe. This is expressed in Equation 2 as follows,

$$\exists \text{income}, \text{income} > 1 \tag{2}$$

The quantifiers can also be used to express through the layering of two dimensions or more. For example, a selection of a display component is to compare between all categories of income and some categories of expenditure. The sample of two-layering composition named functional data is set in Equation 3 as follows,

$$\forall \text{income U} \exists \text{expenditure} \tag{3}$$

The dataset of the dimensional layering from one to one horizontal or vertical or oblique line can be taken to execute with some aspects through the weaver. A combination of functional data is from a set of data between layers. A transformation of weaving including the functional data and the aspect elements are cut across by a method call. A symbol of crosscutting concerns is assigned using $\otimes$. A set of data in the multidimensional on multi-layers is designed supporting the crosscutting concerns (functional data $\otimes$ aspect).

Operational semantics of dimensions and layers are expressed in Equations 4 to 7. For instance, the layering of an income dimension computes to display an amount of salary categories in May 2019. The transformation of weaving must be executed through the functional data of an income layering and the aspects are: type, total, month, and year (Figure 3 [1]). We let the type = {$Income_{salary}$: Salary Earn Income ($\Gamma_{\subseteq}^{\cdot}$), we express a type aspect to call subdimensions}, the total = {sum(): $\sum \mathbf{n}$}, the month = {May}, the year = {2019}. For each execution, an amount of salary categories in May 2019, is computed in Equation 4.

$$\exists Income_{salary} \otimes TotalU(Type, Type = salary)U(Month, Month = May)U(Year, Year = 2019) \quad (4)$$

$$\text{or } \otimes \sum_{\exists income_{salary} \in Income}^{n} (MayU2019)$$

However, at the same pointcut, an amount of salary categories can represent the information differently. The aspects are composited relatively but the same aspects can be executed with different semantics depending on the parameters as shown in Figure 3 [2]. We call an amount of income in May 2018. We let the type = {$Income_{salary}$}, the total = {sum(): $\sum \mathbf{n}$}, the month = {May}, the year = {2018}. By computing, the statement is assigned in Equation 5.

$$\exists Income_{salary} \otimes TotalU(Type, Type = salary)U(Month, Month = May)U(Year, Year = 2018) \quad (5)$$

The formula is expressed using a composition of two layering such as for a cutting point from one horizontal and vertical layer to become two layering in order to compute relatively between two dimensions. For instance in Figure 3 [3], the financial statement computed a balance of income and expenditure from 1st–15th March 2019. We set the type = {Income, Expenditure}, the total = {sum(): $\sum \mathbf{n}$}, The day = {1, 2, 3, …, 15}, the month = {March}, the year = {2019}. This is expressed in Equation 6 as follows,

$$(\exists Income \otimes TotalU(Day, Day = \{1, 2, 3, \ldots,15\})U(Month, Month = March)U(Year, Year = 2019))U$$

$$(\exists Expenditure \otimes TotalU(Day, Day = \{1, 2, 3, \ldots,15\})U(Month, Month = March)U(Year, Year = 2019))$$

or

$$(\exists Income\ U\ \exists Expenditure) \otimes TotalU(Day, Day = \{1, 2, 3, \ldots,15\})U(Month, Month = March)U(Year, Year = 2019)) \quad (6)$$

The three layering is designed supporting the computation of three dimensions for showing the amount of income, expenditure, and liabilities. In Figure 3 [4], the relation of a cutting point is called from one horizontal, vertical and oblique line. The comparison of three domain concerns can represent a balance of income, expenditure, and liabilities in June 2019. We let the type = {Income, Expenditure, Liabilities}, the total = {sum():Σ_n}, the month = {June}, the year = {2019}. This is computed in Equation 7 as follows,

$$(\exists \text{Income} \otimes \text{TotalU(Month, Month = June)U(Year, Year = 2019))U(}\exists \text{Expenditure} \otimes \text{TotalU(Month, Month = June)U(Year, Year = 2019))U(}\exists \text{Liability} \otimes \text{TotalU(Month, Month = June)U(Year, Year = 2019))}$$

or

$$(\exists \text{Income U } \exists \text{Expenditure U } \exists \text{Liability}) \otimes (\text{TotalU(Month, Month = June)U(Year, Year = 2019)} \quad (7)$$

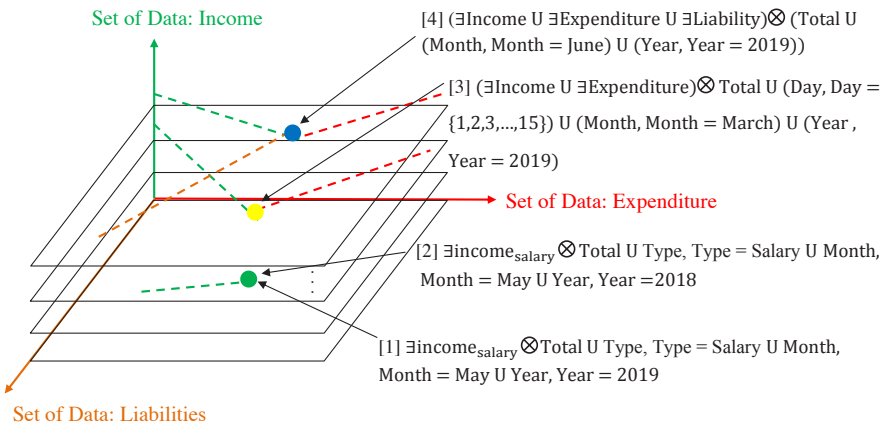

Figure 3. Formulation and execution of cutting points on three-dimensional layering.

INFORMATIVE MULTIDIMENSIONAL OPERATION FRAMEWORK

By composing the three-dimensional layering and the aspect elements for data execution and management, an integration stage of layering through components using a weaver is shown in Figure 4. The composition of the

informative multidimensional layering is the extensional framework of the early stage of this study (Figure 1). We illustrate the combination of layering and pointcuts for data manipulation. There are two method calls for the execution of weaving. The combination is composed of functional data and aspect elements. The weaver is an analytical operation to call particular objects (three-dimensional layering and pointcuts) into a component. The components are created following the personal finance application requirement specifications. The samples of the component names are DisplayTotal, CompareStatement, InsertAccount, UpdateAccount, and DeleteAccount. For example, the DisplayTotal component displays the total income. In running time, the component consists of many jointly crosscutting points such as income, date, type, and total aspect. Each one is a sequence of methodologies from 1 to n components depending on the particular call of an end user interaction.

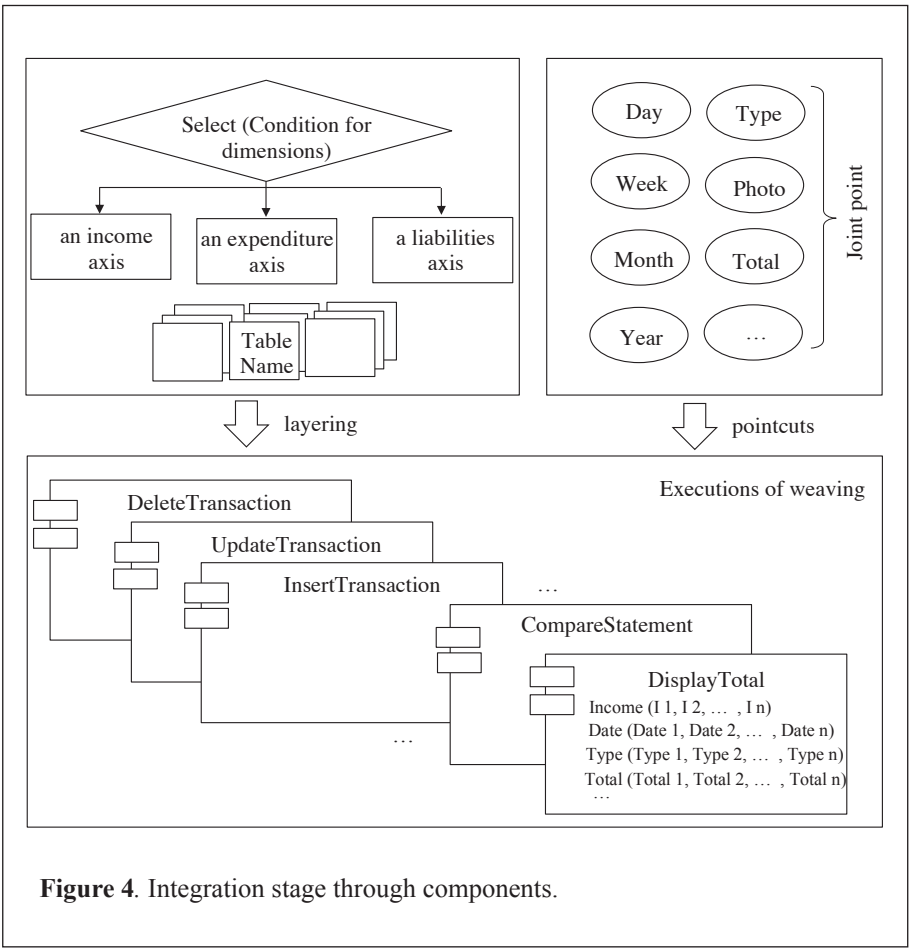

Figure 4. Integration stage through components.

To support the design of the multidimensional information layering, Figure 5 represents the architecture of a Three-layer User Interface Composition Model (TUICM). The UI layer on top, facilitates the visual page that allows the end users to interact with menus and retrieve information records on output devices. The UI layers communicate with components and data layers of a software system for manipulating data from the lower layers. The component layer provides components that decompose functional data (FD) and aspect elements (AE). Each FD and AE is called to express an execution to the top layer. The data layer manipulates data acquisition from the different calling stages through the three-dimensional layering.

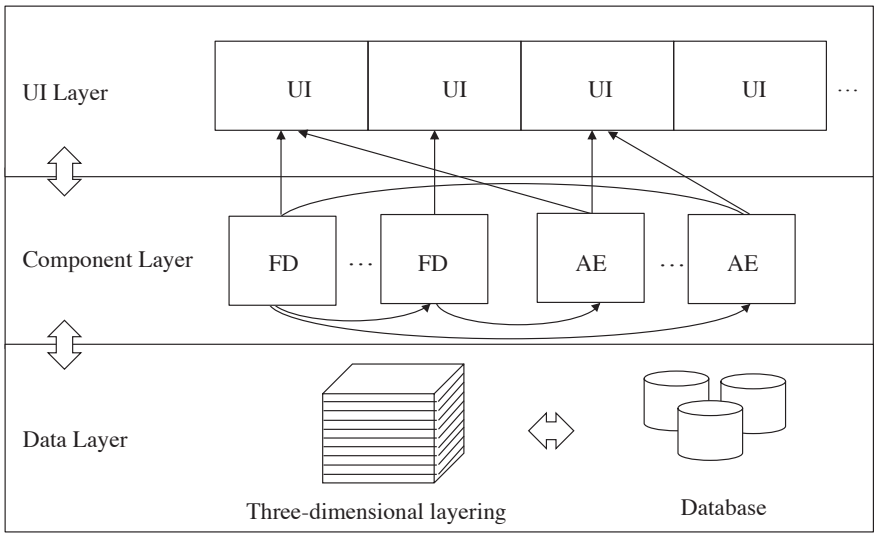

Figure 5. Three-layer user interface composition model.

With the given definition and execution designs of the three-dimensional layering personal finance including the functional data and aspect elements, we could alter the multidimension layering into the user interface operations. The interface views are based on the conceptual design of the previous sections. The multidimensional layering can combine functional data, aspect elements, and the layers. We believe that our integration stage (Figure 5) enables transfer of the model to the UI design in an orderly manner. There are three different executable programs (compile-time, run-time, and weave-time) during the active life of AOA. The first stage is compiling time. The interface prototypes (Figure 6-7) are designed for displaying layouts on a computer screen to

the end user. To transform the principle of separating concerns into the end user review phase, our UI approach is divided into three sections (functional data, aspect element and composition section). The second stage is running time. We express an interaction of an end user selection to the buttons. By separating a UI design into three sections, layout buttons are provided in each section. The third stage is weaving time. A section of weaving executions are specified such as to call a dimension of an income axis to display an amount balance of income and categories of income (Figure 6), and to call two and three dimensions for comparing a financial statement (Figure 7). A dynamic selection is provided in this design when the end user selects and/or deselects the buttons. Then the execution of the system will display a set of output data differently.

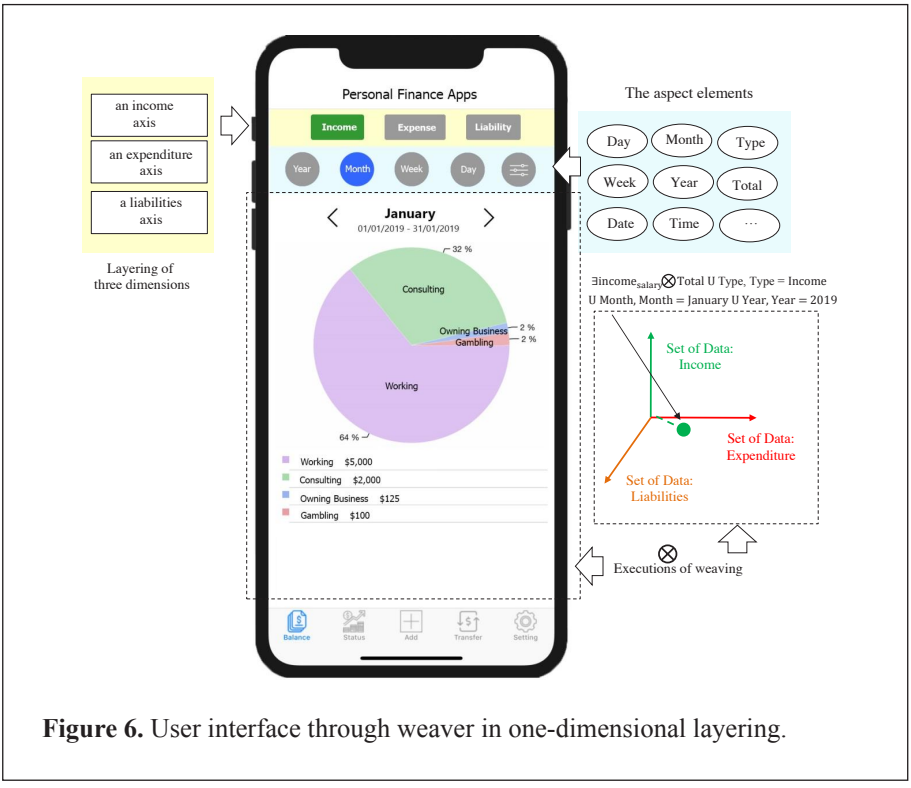

Figure 6. User interface through weaver in one-dimensional layering.

Another component is a CompareStatement in Figure 7. We have applied dynamic weaving through the layering of three dimensions in order to compare a number of financial statements. The functional data can be selected for crosscutting concerns from three method calls (an income, an expenditure, and a liabilities menu). The execution statement depends on the current stage of selection controls through the buttons. Figure 7 shows the multidimensional data operation in different method calls through the interfaces and operational

semantics of dimensions and layers. The comparison of responsiveness also provides layering of the time series in a set of aspect elements. Thus, the month button is specified as an aspect element that is selected in the lower section. At the weaving point, the execution of weaving enables an adaptive display of the total number of recordings depending on the end user selection. Therefore, the composition of the separate designs seems to support the crosscutting layering views of the functional data and the aspect elements.

Figure 7. User interface through weaver in multidimensional layering.

PRACTICAL EVALUATION

We conducted an end user evaluation to assess our user interfaces of the mobile personal finance application prototype. The end user usability of the mobile application was designed to study the interaction of the framework composition through the visual user interfaces.

Objectives

In order to evaluate the end user usability, the following objectives were identified to assess the performance of the layered-combination design of the three-dimensional functional data and the aspect elements:

- RQ1: Does the separation of concerns, data, layers (sections), and composition help users to use the application easily?
- RQ2: Does the composition of crosscutting concerns through adaptive buttons help users to access data effectively?
- RQ3: Does the design of a variety of functionalities and aspects for an all-in-one touch screen help users to control and manage their experience comfortably?

Experimental Design

This study was designed using the three components related to Figures 6 and 7. We used a tutorial guild to tip the participants as to how to use our mobile application. The first component was to insert an income transaction into the InsertAccount component. The screen showed how the end user could access and insert a subcategory of an income record. The subdimensions belonging to the particular dimension could be called to access and select the subdimensions. The second component was the display of the total balance in the DisplayTotal component (Figure 6). The screen showed how the end user could access and see categories and subcategories of income records. The subdimensions belonging to the one dimension layering could be reported by end user clicking. The last component was the CompareStatement component (Figure 7). The comparison screens of the financial statements were assigned for the users to access the three record categories.

To evaluate end user satisfaction on mobile usability, a questionnaire with 19 items and an open-ended question to elicit opinions and suggestions for improving the mobile application was administered. Research questions from the System Usability Scale (SUS) questionnaire (Brooke, 1996), the controllability and management of mobile interactions (Hussain, Hashim, Nordin, & Tahir, 2013; Tonder & Wesson, 2012), and the questionnaires of Cui and Honkala (2013) were employed as part of the questionnaire in this study. We had grouped all items into three aspects: 1) Perceived Usefulness (5 items), 2) Perceived Ease of Use (5 items), 3) and Perceived Controllability and Management (9 items). Each statement was rated on a Likert scale from 1 (strongly disagree) to 5 (strongly agree). The questionnaire consisted of the following questions:

Perceived Usefulness (PU)

Q1. I felt confident using the application.
Q2. I found the application unnecessarily complex.
Q3. I found that the various functions in this application were well integrated.
Q4. I would imagine that most people would learn to use this application very quickly.
Q5. I think I would like to use this application frequently.

Perceived Ease of Use (PEU)

Q6. I thought there was inconsistency in this application.
Q7. I think that I would not need a technical person to support or advise me on using this application.
Q8. It was easy to track any financial information.
Q9. I could get to know my financial statements better than the existing application.
Q10. I thought the overall application was easy to use.

Perceived Controllability and Management (PCM)

Q11. I felt comfortable inserting category names of records for income, expenditure, and liabilities.
Q12. I felt comfortable inserting any transaction into my financial accounts.
Q13. I thought the application separated my categories clearly.
Q14. I think I can use the buttons to manage my financial records quickly.
Q15. I think the sections on the screen are separated with good looks.
Q16. I think the purpose of displaying the income, expenditure and liabilities button on the same screen is to make it easy to use the functions.
Q17. I think the purpose of separating periods of time (day, week, month, and year) is to make it convenient to view my reports.
Q18. I think it is easy to understand how the application works. The task of comparing my financial statement(s) by clicking buttons and then the resulting report(s) which can easily be changed.
Q19. I felt comfortable and could easily use the variety of functions that was designed for the all-in-one touch screen.

The following items were identified to match the research questions. The RQ1 consisted of: Q11, Q12, Q13, Q15, and Q17. The RQ2 consisted of: Q6, Q8, Q9, Q14, and Q18. The RQ3 consisted of: Q2, Q3, Q10, Q16, and Q19. We assumed that the minimum value of an expectation to validate was 75% for usefulness and satisfaction. The 75% was established as a final acceptance benchmark for most usability values (Veral & Macias, 2019).

Participants

We ran the experimental study for 100 participants (50 female and 50 male), aged between 18 and 30 who had experience in using any personal finance and/or accounting applications for more than three months. All the participants who volunteered to review the application usage task were from our university.

Procedure and Tasks Performed

We gave the participants our prototype application and allowed them to use the personal finance application freely. All users were requested to respond to the post-task questionnaire in order to find out their perceived usage. After the participants had accessed the tutorial guild, we allowed them to insert income categories and subcategories (T1). We also told them to insert more categories for expenditure and liabilities (T2). After that, the participants were asked to insert their usual transactions (income, expenditure, and liabilities records) (T3). Then the participants clicked the balance page to see the reports of each recording (T4) and following that, clicked the status page to view a comparison of their financial statements (T5). The buttons enabled them to see the different views. In the final stage, we administered the questionnaires to the participants to evaluate the end user usability test of our prototype conceptual design.

Results

In order to assess the end user usability, answers to the research questions were analyzed. The items were grouped as explained in the experimental design. The comparison of the research questions based on mean, minimum value (min), maximum value (max), and standard deviation (SD) is shown in Table 2. The minimum value of the expectation was 75%.The average value of RQ1 obtained for the purpose of separating views was 89.12%. The average value of RQ2 obtained for the purpose of dynamic compositions was 88.88%. The average value of RQ3 obtained for the purpose of all-in-one screen was 89.56%. Therefore, all research questions were more than the minimum value of expectations.

Based on the demographic information gathered from the participants who completed the questionnaires, 26% of the participants had experience in using personal finance applications from six to more than 12 months. All participants were between 18–30 years old. The overall mean of the topic rating scales for end user usability is presented in Figure 8. The result was based on a five-point Likert scale from 1 to 5.

Table 2.

End User Results Based on Research Questions.

		Mean	Percentage	SD	Min	Max
RQ1	Q11	4.41	88.20	0.5522	3	5
	Q12	4.32	86.40	0.4899	3	5
	Q13	4.45	89.00	0.5573	3	5
	Q15	4.58	91.60	0.5160	3	5
	Q17	4.52	90.40	0.5409	3	5
	Total	4.46	89.12	0.5373	3	5
RQ2	Q6	4.52	90.40	0.5218	3	5
	Q8	4.45	89.00	0.5389	3	5
	Q9	4.37	87.40	0.4852	3	5
	Q14	4.32	86.40	0.5101	3	5
	Q18	4.56	91.20	0.5187	3	5
	Total	4.44	88.88	0.5210	3	5
RQ3	Q2	4.46	89.20	0.5759	3	5
	Q3	4.28	85.60	0.5333	3	5
	Q10	4.44	88.80	0.5187	3	5
	Q16	4.68	93.60	0.4899	3	5
	Q19	4.53	90.60	0.5766	3	5
	Total	4.48	89.56	0.5533	3	5

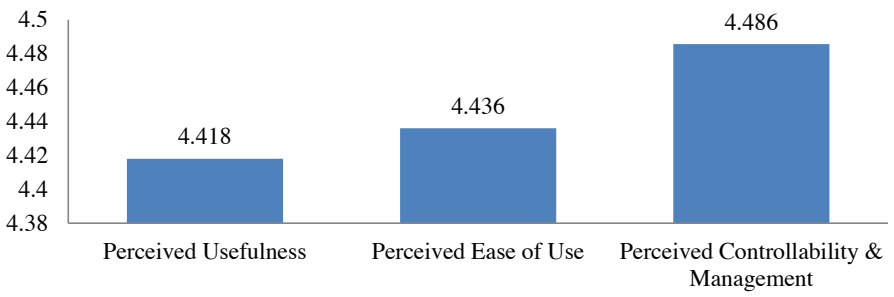

Figure 8. Overall mean of topic rating scales.

Based on some of the participants' comments on the open-ended question, it was found that the visual design of the all-in-one screen for reporting financial performance using charts was helpful. Besides, it was easy to view the comparison of the participants' records in a more comfortable and systematic manner. Ease of account reporting and financial comparison were most commented on by the participants, especially using the dynamic buttons. They could recognize the reason for using the different colours for the buttons.

Independent samples t-test was used to compare the means of two independent groups.We assigned the gender of the participants. The results showed statistically significant differences among Q9 ($F = 14.337$, $p <= 0.01$), Q15 ($F = 18.831$, $p <= 0.01$), and Q19 ($F = 12.887$, $p <= 0.01$). Two-way ANOVA test was applied to compare the means of three or more independent groups.We used the experiences of the application usage. The results showed statistically significant differences among Q1 ($F = 19.028$, $p <= 0.01$), Q7 ($F = 12.468$, $p <= 0.01$), Q8 ($F = 7.692$, $p <= 0.01$), Q16 ($F = 18.826$, $p <= 0.01$), and Q17 ($F = 11.578$, $p <= 0.01$).

DISCUSSION

We sorted the information requirements into sets of multidimensional data as proposed by Akanmu and Jamaluddin (2016). A comprehensive data dimension was reliably elicited from the dataset in the database tables. The multidimensional layering was specified for supporting data operations of the personal finance mobile application. By dividing the information recording into three dimensions, the functional data was defined as a correlative relationship of sets of data. An infinite series of a set of dimensions and subdimensions was proposed. The multidimensional relationships have been claimed to support software design using layers (Boukraa, Boussaid, Bentayeb, and Zegour, 2013). Thus, any new concerns or information requirements could be proposed incrementally and added in the dimensions. Our dimensional design was described along one dimension (Batory & Geraci, 1997; Batory, Liu, & Sarvela, 2003). The composition defined the set of all representations of refinement programs from {f1, …., fn} combined to a program p from {p1, …, pn}. Besides, more dimensions supporting the degree of abstractions were related to our design. The principles of the execution design allowed us to analyze and design any software parts extensively and adaptively. The decomposition of the requirement designs and system functions on the architecture view of cloud computing had supported our idea (Surendro, Supriana, & Supriana, 2016). We proposed that the software system be composed separately supporting a fine granularity and the composition be designed flexibly in the informative multidimensional layering.

In the early stages of the implementation level, an Aspect-Oriented Architecture Design (AOAD) (Rashid, Moreira, & Araujo, 2003; Moreira, Araujo, & Whittle, 2006; Sanchez, Moreira, Fuentes, Araujo, & Magno, 2010) an approach method and technique of separating concerns to improve on system quality attributes known as non-functional requirements was applied. The non-functional requirements can show developers how a software system should be, not what it should do with regards to functional requirements. However, our initial research on the separation of concerns seemed to support all stages of the software development life cycle. Proof of the design concept based on Rukhiran and Netinant (2017b) is shown in Figure 9.

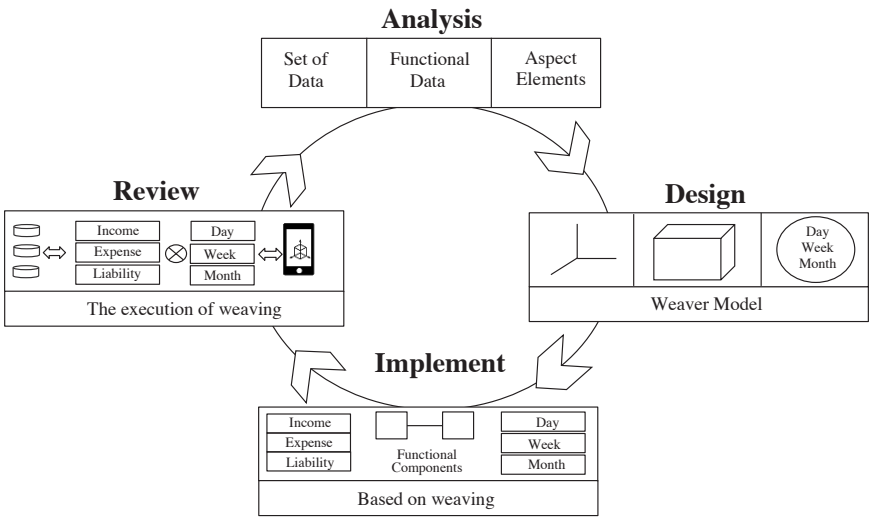

Figure 9. Software development stages using separation of concerns for prototype model.

In supporting our separation of UI designs, we arranged the graphical user interface into three sections. The top section is a selection of the functional data. There are many stages involved in display layering of one dimension, comparison between layering of two dimensions, and comparison among layering of three dimensions. The execution of weaving for displaying data compositions is based on end user selections by way of buttons in supporting the main dynamic design of Richard and John (2010).

We have experimented with the end user usability of the personal finance mobile application in the evaluation of this paper,. By applying the

questionnaire from many kinds of research, the topics were in agreement with the Technology Acceptance Model (TAM) (Davis & Venkatesh, 1996; Ibrahim & Al-Rawashdeh, 2014). The TAM integrates the determinants of perceived ease of use and perceived usefulness to assess and predict user acceptance. Moreover, we have designed a specific brief of perceived controllability to prove our conceptual design. The results from our research questions have passed the minimum value of expectations, which is consistently related to Veral and Macias (2019). Although there are different methods and techniques to evaluate user perception (Nooraishaya, Ahmad, & Ali, 2018), the approach in assessment of the visual composition of the end user perspective in this study was based on mobile application design and performance (Alalwan, 2020).

CONCLUSION AND FUTURE WORK

One of the key challenges in a mobile application is the focus on improving software quality attributes to support a variety of mobile performances. Breaking down a software system into smaller pieces is one solution to allow us to define the fine granularity for achievable data and reusable functions. We divided the application specification into two segments. The multidimensional layering of the personal finance information was assigned to support data acquisition and manipulation. The aspect element was assigned to operate functional methodology. The informative three-dimensional layering enabled support in the development of the personal finance application to achieve flexible and adaptable designs. We assumed the case study of an execution rule on the layering of three dimensions and aspect elements (Figure 3). The operational semantics analyzed these concerns using components. From the three-dimensional information, layers, functional data, and aspect elements, we developed the three-dimensional user interface composition model. The Three-layer User Interface Composition Model (TUICM) (Figure 5) which responded to UI gave a better performance in terms of relationships of UI, components, and data access layers. By dividing the display component to support our analyzed approach, we illustrated a cooperative UI design to keep a clear separation of the different layers (Figure 6–7). The results of the end user usability showed the multi-layered approach which enabled adapting of data operations and reporting.

Thus, this article has drawn a variety of peopleware in software development. Firstly, a system analyst can understand the relational separation of personal finance information through a collection of methods for analyzing functions and data via a three-dimensional model. The principles of the model can be applied to any software business. Secondly, it enables a developer to adapt our design for clean codes and less inheritance. AOSD is one of the approach

techniques that can avoid scattered (duplication) and tangled (dependency) codes. Our software design is an open software architecture that can be transformed and implemented to support any programming language. Finally, an end user should be able to use the well-organized software design to interact with data or access any information with no more than the basic rule of three mouse clicks.

To push forward the personal finance software design to the next stage, we will focus on an architecture constraint design of components. We intend to design the components for the personal finance software to support functional specifications and adaptations. An execution flow diagram of the components will be provided to connect all components and information flow in the software system. By improving on the separation of concerns in suitable UI designs, we plan to focus on various kinds of end user interface designs such as for the elderly and farmers. The use of technology by the elderly has a beneficial influence on enhancing their quality of life in an increasing ageing population in our country. In addition, farming is a major occupation which contributes to the Thai economy. Most existing software do not support the particular financial business statements of farmers. Therefore, this house bookkeeping software design project could take traditional Thai farmers forward to become smart farmers.

ACKNOWLEDGMENT

This work was supported by the National Broadcasting and Telecommunication Commission (grant number BT2-15/1-61, from 2019–2021), Rangsit University and Rajamangala University of Technology Tawan-ok, Thailand. This research project was funded in the amount of $100,000USD.

REFERENCES

Alalwan, A. A. (2020). Mobile food ordering apps: An empirical study of the factors affecting customer e-satisfaction and continued intention to reuse. *International Journal of Information Management*, *50*, 28-44. doi: 10.1016/j.ijinfomgt.2019.04.008

AI-Hudhud, G. (2015). Aspect-oriented design for team learning management system. *Computer in Human Behavior, 51*(PB), 627-631. doi: 10.1016/j.chb.2015.01.032

Ardito, C., Costabile, F. M., Desolda, G., Lanzilotti, R., Matera, M., Piccinno, A., & Picozzi, M. (2015). User-driven visual composition of service-based interactive spaces. *Journal of Visual Languages and Computing, 25*(4), 278-296. doi: 10.1016/j.jvlc.2014.01.003.

Akanmu, S. A., & Jamaluddin, Z. (2016). Designing information visualization for higher education institutions: A pre-design study. *Journal of Information and Communication Technology, 15*(1), 145-163.

Barricelli, R. B., Cassano, F., Fogli, D., & Piccinno, A. (2019). End-user development, end-user programming and end-user software engineering: A systematic mapping study. *Journal of System and Software, 149*, 101-137. doi:10.1016/j.jss.2018.11.041

Batory, D., & Geraci, B. (1997). Composition validation and subjectivity in GenVoca generators. *IEEE Transaction on Software Engineering, 23*(2), 67-82. doi: 10.1109/32.585497

Batory, D., Liu, J., & Sarvela, J. N. (2003). Refinements and multi-dimensional separation of concerns. *ACM SIGSOFT Software Engineering Notes, 28*(5), 48-57. doi: 10.1145/949952.940079

Boukraa, D., Boussaid, O., Bentayeb, F., & Zegour, D. (2013). A layered multidimensional model of complex objects. *Proceeding of the 25th International Conference on Advanced Information Systems Engineering* (pp. 498-513). Valencia, Spain: Springer-Verlag.

Brooke, J. (1996). SUS – A quick and dirty usability scale. *Usability Evaluation in Industry, 189*, 4-7.

Butting, A., Eikermann, R., Kautz, O., Rumpe, B., & Wortmann, A. (2019). Systematic composition of independent language features. *The Journal of Systems and Software, 152*, 50-69. doi: 10.1016/j.jss.2019.02.026

Cui, Y., & Honkala, M. (2013). A novel mobile device user interface with integrated social networking services. *International Journal of Human Computer Studies, 71*(9), 919-932. doi: 10.1016/j.ijhcs.2013.03.004

D'Andrea, R. (1999). Software for modeling, analysis, and control design for multidimensional systems. *Proceedings of the IEEE International Symposium on Computer Aided Control System Design* (pp. 24-27). Hawai, USA: IEEE Company Society.

Davis, D. F., & Venkatesh, V. (1996). A critical assessment of potential measurement biases in the technology acceptance model: Three experiments. *International Journal of Human Computer Studies, 45*(1), 19-45. doi: 10.1006/ijhc.1996.0040

Dennis, A., Wixom, H. B., & Roth, M. R. (2012). *System Analysis & Design*. New York, NY: John Wiley & Sons, Inc.

Desolda, G., Ardito, C., Costabile, F. M., & Matera, M. (2017). End-user composition of interactive applications through actionable UI components. *Journal of Visual Languages and Computing, 42*, 46-59. doi: 10.1016/j.jvlc.2017.08.004

Diaz, M., Romero, S., Rubio, B., Soler, E., & Troya, M. J. (2005). An aspect-oriented framework for scientific component development. *Proceedings of the 13th Euromicro Conference on Parallel* (pp. 290-296). Washington, USA: IEEE Company Society. doi:10.1109/EMPDP.2005.11

Eeles, J. (2001). *Layering Strategies*. San Jose, CA: Rational Software Corporation.

Gibbs, I., Dascalu, S., & Harris, C. F. (2015). A separation-based UI architecture with a DSL for role specialization. *Journal of System and Software, 101*, 69-85. doi: 10.1016/j.jss.2014.11.039

Hays, J. R. (2014). *User interface design for online social media*. California Polytechnic State University, California, USA.

Hinderks, A., Schrepp, M., Mayo, J. D. F., Escalona, J. M., & Thomaschewski, J. (2019). Developing a UX KPI based on the user experience questionnaire. *Computer Standards & Interfaces, 65*, 38- 44. doi:10.1016/j.csi.2019.01.007

Hoffman, K., & Eugster, P. (2008). Towards reusable components with aspects: An empirical study on modularity and obliviousness. *Proceedings of the 30th International Conference on Software Engineering* (pp. 91-100). Leipzig, Germany: ACM. doi: 10.1145/1368088.1368102

Hussain, A., Hashim, N. L., Nordin, N., & Tahir, H. M. (2013). A metric-based evaluation model for applications on mobile phones. *Journal of Information and Communication Technology, 12*(1), 55-71.

Ibrahim, H., & Al-Rawashdeh, T. A. (2014). Acceptance of web-based training system among public sector employees. *Journal of Information and Communication Technology, 13*, 87-107.

Jallow, A. K., Demian, P., Anumba, C. J., & Baldwin, A. N. (2017). An enterprise architecture framework for electronic requirements information management. *International Journal of Information Management, 37*(5), 455-472. doi: 10.1016/j.ijinfomgt.2017.04.005

Jelinek, J., & Slavik, P. (2004). GUI generation from annotated source code. *Proceedings of the 3rd Annual Conference on Task Models and Diagrams* (pp. 129-136). Prague, Czech Republic: ACM. doi: 10.1145/1045446.1045470

Kennard, R., & Leaney, J. (2010). Towards a general purpose architecture for UI generation. *Journal of System and Software, 83*(10), 1896-1906. doi:10.1016/j.jss.2010.05.079

Kiczales, G., Lamping, J., Mendhekar, A., Maeda, C., Lopes, C., Loingtier, J., & Irwin, J. (1997). Aspect-oriented programming. *Proceedings of the 11th European Conference on Object-Oriented Programming* (pp. 220-242). Finland: Springer-Verlag. doi: 10.1007/BFb0053381

Kumar, A., Kumar, A., & Iyyappan, M. (2016). Applying separation of concern for developing softwares using aspect oriented programming concepts. *Proceedings of the International Conference on Computational Modeling and Security* (pp. 906-914). Bengaluru, India: Elsevier B.V. doi: 10.1016/j.procs.2016.05.281

Latizina, M., & Beringer, J. (2012). Transformative user experience: Beyond packaged design. *Interactions, 19*(2), 30-33. doi: 10.1145/2090150.2090159

Leach, R. J. (2016). *Introduction to software engineering* (2nd ed.). Boca Raton, Florida: CRC Press.

Lin-lin, Z., Shi, Y. You-cong, N., Jing, W., Kai, Z., & Peng, Y. (2008) Towards multi-dimensional separating of NFRs in software architecture. *Proceedings of the International Conference on Computer Science and Software Engineering* (pp. 104-107). Hubei, China: IEEE Company Society. doi: 10.1109/CSSE.2008.1201

Mirbel, I., & Rivieres, V. (2003). Conciliating user interface and business domain analysis and design. *Proceedings of the International Conference on Object-Oriented Information Systems* (pp. 383-399). Geneva, Switzerland: Springer-Verlag. doi: 10.1007/978-3-540-45242-3_40

Moreira, A., Araujo, J., & Whittle, J. (2006). Modeling volatile concerns as aspects. *Proceedings of the 18th International Conference on Advanced Information Systems Engineering* (pp. 544-558). Luxembourg, USA: Springer-Verlag. doi: 10.1007/11767138_36

Muck, T. R., & Frohlich, A. A. (2014) Aspect-oriented RTL HW design using system C. *Microprocessors and Microsystems, 38*, 113-123. doi: 10.1016/j.micpro.2013.12.002

Netinant, P., & Elrad, T. (2016). Separation of concerns in designing mobile software. *Rangsit Journal of Arts and Sciences, (6)*1, 89-96. doi: 10.14456/rjas.2016.8

Nooraishaya, W., Ahmad, W., & Ali, N. M. (2018). The impact of persuasive technology on user emotional experience and user experience over time. *Journal of Information and Communication Technology, 17*(4), 601-628.

Oxford English Dictionary. (2011). Oxford: Oxford University Press.

Panunzion, M., & Vardanega, T. (2014a). A component-based process with separation of concerns for the development of embedded real-time software systems. *The Journal of Systems and Software, 96*, 105-121. doi: 10.1016/j.jss.2014.05.076

Panunzion, M., & Vardanega. T. (2014b). An architectural approach with separation of concerns to address extra-functional requirements in the development of embedded real-time software systems. *Journal of Systems Architecture, 60*(9), 770-781. doi: 10.1016/j.sysarc.2014.06.001

Pedersen, T. B., & Jensen, C.S. (1999). Multidimensional data modeling for complex data. *Proceedings of the 15th International Conference on Data Engineering* (pp. 336-345). Sydney, Australia: IEEE Computer Society.

Pekilis, R. B. (2002). *Multi-dimensional separation of concerns*. Technical Research Report. University of Waterloo.

Raheman S. R., Maringanti, H. B., & Rath, A. K. (2018). Aspect oriented programs: Issues and perspective. *Journal of Electrical Systems and Information Technology, 5*, 562-575. doi: 10.1016/j.jesit.2017.06.003

Rashid, A., Moreira, A., & Araujo, J. (2003). Modularization and composition of aspectual requirements. *Proceedings of the 2nd International Conference on Aspect-Oriented Software Development* (pp.11–20). Boston, MA: ACM. doi: 10.1145/643603.643605

Richard, K., & John, L. (2010). Toward a general purpose architecture for UI generation. *The Journal of Systems and Software, 83*(10), 1896-1906. doi: 10.1016/j.jss.2010.05.079

Rosenmuller, M., Siegmund, N., Thum, T., & Saake, G. (2011). Multi-dimensional variability modeling. *Proceedings of the 5th Workshop on Variability Modeling of Software-Intensive Systems* (pp. 11-20). Namur, Belgium: ACM. doi: 10.1145/1944892.1944894

Rukhiran, M., & Netinant, P. (2017a). The concept design of house bookkeeping software using Aspect-oriented approach. *Proceedings of the 2017 International Conference on Information Technology* (pp. 232-236). Singapore, Singapore: ACM. doi: 10.1145/3176653.3176667

Rukhiran, M., & Netinant, P. (2017b). Aspect-oriented approach for supporting house bookkeeping software design. *Proceedings of the 2017 International Conference on Software and e-Business* (pp. 49-54). Hong Kong: ACM. doi: 10.1145/3178212.3178217

Sadowski, C., & Zimmermann, T. (2019). *Rethinking productivity in software engineering*. Berkeley, California: Apress Open.

Sanchez, P., Moreira, A., Fuentes, L., Araujo, J., & Magno, J. (2010). Model-driven development for early aspects. *Information Software Technology, 52*(3), 249-273. doi: 10.1016/j.infsof.2009. 09.001

Silveira, F. F., Cunha, A. M., & Lisboa, M. L. (2014) A state-based testing method or detecting aspect composition faults. *Proceedings of the 14th International Conference on Computational Science and Its Applications* (pp. 418-433). Guimaraes, Portugal: Springer-Verlag. doi: 10.13140/2.1. 2306.1762

Sommerville, I. (2014). *Software Engineering* (10th ed.). Boston, Massachusetts: Pearson Education, Inc.

Surendro, K., Supriana, A., & Supriana, I. (2016). Requirements Engineering for Cloud Computing Adaptive Model. *Journal of Information and Communication Technology, 15*(2), 1-17.

Tanter, E., Figueroa, I., & Tabaerau, N. (2014). Execution levels for aspect-oriented programming: Design, semantics, implementations and applications. *Science of Computer Programming, 80*, 311-342. doi: 10.1016/j.scico.2013.09.002

Tarr, P., Ossher, H., Harrison, W., & Sutton, M. S. (1999). N degrees of separation: Multi-dimensional separation of concerns. *Proceedings of the 21st International Conference on Software Engineering* (pp. 107-119). Los Angeles, California: ACM. doi: 10.1145/302405.302457

Tonder, P. B., & Wesson L. J. (2012). Improving the controllability of tilt interaction for mobile map-based applications. *International Journal of Human Computer Studies, 70*(12), 920-935. doi: 10.1016/j.ijhcs.2012.08.001

Usoro, A. (2013). Effective document and data management. *International Journal of Information Management, 33*(4), 702-705. doi: 10.1016/j.ijinfomgt.2013.04.004

Veral, R., & Macias, A. J. (2019). Supporting user-perceived usability benchmarking through a developed quantitative metric. *International Journal of Human Computer Studies, 122*, 184-195. doi: 10.1016/j.ijhcs.2018.09.012

Zhang, G., & Rong, M., 2009. A framework for dynamic evolution based on reflective aspect-oriented software architecture. *Proceedings of the 4th International Conference on Computer Sciences and Convergence Information Technology* (pp. 7-10). Seoul, South Korea: IEEE. doi: 10.1109/ICCIT.2009.102