# Research and Development of CAIBuilder: an Innovative Authoring Tool for Creating E-Courseware Easily

**Chailerd Pichitpornchai, M.D., Ph.D.**

*Department of Physiology, Faculty of Medicine Siriraj Hospital, Mahidol University, Bangkok 10700, Thailand*

## ABSTRACT

**Objective:** In recent years, educators have increasingly adopted e-learning. Unfortunately, most e-coursewares or Computer-Aided Instructions (CAI) could not always fulfill educators' needs. This results in a need to develop their own e-courseware by using available electronic authoring tools; however, there are difficulties in using these tools. The present study was intended to research on the difficulties in using authoring tools and to identify features needed by educators, the results would de used to develop an innovative *CAIBuilder* software.

**Methods:** Waterfall model, human-computer interaction and object action interface model have been employed in designing and implementing CAIBuilder. The software was written with ToolBook OpenScript.® The beta version was tested, bugs were fixed and additional features were coded. The final version was used to conduct two workshops for fifty-three non-programmer university educators. Pre-test and post-test questionnaires were distributed to collect demographic data, knowledge, concepts, skills, experience, and confidence on creating multimedia applications, including key features of CAIBuilder.

**Results:** The results were analyzed and presented in percentages, and the Wilcoxon signed rank test was used for comparison. The results demonstrated that over 85% of non-programmer educators found CAIBuilder was user-friendly, easy to learn, and also powerful. The Wilcoxon signed rank test at a significant level $p < 0.0001$ indicated that, after the CAIBuilder workshops, over 80% of participants had increased their knowledge and confidence in creating multimedia applications by themselves, creating computerized test items, navigational controls and program flows. *The Autonomic Nervous System* and *Physical Modalities* are two examples of interactive e-courseware implemented by using CAIBuilder. and they have been Currently, they are being used in the teaching of physiology and physiotherapy to medical students at the Faculty of Medicine, Siriraj Hospital.

**Conclusions**: CAIBuilder is an innovative electronic authoring tool of choice for non-programmer authors to create an interactive e-courseware of good quality with feedback in a reasonably quick time frame without programming.

**Keywords:** E-learning; CAI; Innovative authoring tool; Waterfall model; Object-action interface

E-learning is playing an increasingly impor tant role in educational systems, especially medi caleducation, whereas a shortage of medical doctors has been a major public health problem in Thailand.[1, 2] This problem brings about an emergence for medical schools to produce more medical graduates[3] while the number of medical staff is still unchanged or, even worse, reduced because of the national policy of laying off staff. At the same time, medical education has quietly expanded in terms of content overload.[4] E-learning via electronic courseware (e-courseware) or Computer Aided Instruction (CAI) can help the teaching staff transfer an enormous amount of knowledge to their students and save more of their time concentrating on research.[5] At present, multimedia teaching materials are available to some extents. However, they do not fulfill most of the needs of university educators.[6] To produce good CAIs relevant to curricula, at least four kinds of experts are required, including content or subject matter experts, instructional de signers, computer programmers and graphic designers.[7-9] In general, it can be assumed that every teaching staff member is a content expert. In Thailand, there are very few instructional designers; hence, the staff member is also presumably an instructional designer. There are only a very small fraction of the staff who are also capable of computer programming and graphic designing. The difficulty in co-operation and communication among many experts to produce a good CAI has been known as a major problem which causes delivery delay.[10, 11]

At present, electronic authoring programs or tools available in the market, such as Asymetrix ToolBook II Instructor®, Macromedia Authorware®, and Macromedia Director®, are either easier to use but less powerful or more difficult to use but more powerful.[6,9,12,13] Most academics, who are not programmers, have almost always found that an powerful software is quite difficult to start with and takes too much time to learn.[14] They also found that the easier-to-use software contains too little functionality or the software is not powerful when the software has been used for awhile. Sometimes the staff found that

the functionalities they would like to use to create their electronic lessons were not available. These findings bring up a need to research and develop an electronic authoring tool that fulfills the non-programmer academics' requirements. The software should be able to be learned easily with less effort on scripting or programming and yet providing powerful functionalities. Hence, in the present study, the innovation of an electronic authoring software, namely *CAIBuilder*, that can be used by most inexperienced e-courseware developers or non-programmer academics has been researched and developed.

The research and development methodology or software engineering was based on the Waterfall Model[15] which is a methodology of standard software engineering. It comprises (1) identifying users' requirements and problems, (2) analyzing the users' requirements and problems, (3) designing software, (4) coding and implementing software, (5) testing and debugging, and (6) documenting and maintaining the software. The research was divided into four parts consisting of the following (I) identifying the electronic authors' requirements and problems by interviewing experienced academics who used to develop multimedia lessons; (II) designing and implementing the software by using OpenScript® (a programming language of Asymetrix ToolBook II Instructor software); (III) testing and evaluating the software by conducting two workshops of *Producing CAI easily with CAIBuilder* and using pretest and post-test questionnaires as tools for data collection; and (IV) examples of developing real CAI lessons for classes.

It was assumed that users' equirements and pro-blems could be identified by interviewing experienced e-courseware developers. The hypotheses of the present study were that at least 80% or more of inexperienced e-courseware developers or non-programmer academics found that CAIBuilder could be learned and used easily; it could be used to create test items for evaluating learners easily; it provides an easy-to-use and user-friendly graphic user interface (GUI), and it contains powerful program functionalities.

## MATERIALS AND METHODS

### Part I: Identifying the electronic authors' requirements and problems

The first part of this study was not intended to explore and compare electronic authoring software extensively, but to formulate a bird's eyes view about experience and difficulties in using some commonly-used electronic authoring softwares, the ease of use of its GUI, and its powerfulness, in order to be used as a starting point in implementing a new authoring tool.

Eight experienced electronic authors were interviewed by two trained interviewers using a semi-open ended questionnaire. Every author had more than two years of experience in using at least one of the electronic authoring programs, including Asymetrix ToolBook II Instructor Version 8.0, Macromedia Director Version 8.0, and Macromedia Authorware Version 7.0, for creating CAI lessons in their teaching and learning classes. Four of the authors are university staff members and four are high school teachers. The interviewers are system analysts who have good verbal communication skills and are well trained to gather information and problems by using a semi-open ended questionnaire as a guideline. The questions in the questionnaire reflected the author's experience and difficulties in using authoring programs of their choice, especially the software's authoring concept or program flow, ease of use of the GUI, the powerfulness of the software, methods of creating objects and giving actions to each object, navigational controls and hyperlink, integrating multimedia objects, and creating test items for evaluations. Each interview took about one to two hours and notes were taken during the interview. The content of each interview was reviewed and verified by the corresponding interviewee, and then analyzed by using qualitative content analysis in order to identify requirements and problems in using existing authoring software, ease of use, and powerfulness of the software.

### Part II: Designing and implementing the software

According to the principles of user-centered design,[16] the result from part I was analyzed for designing GUI and the software functionalities. The software was developed by using OpenScript language[17] of the Asymetrix ToolBook II Instructor software.[18] The working environment of all windows and widgets were designed and made of ToolBook pages, ToolBook objects, and groups of objects whose front-end functionalities were written with OpenScript.

The design of the working area was minimized to the simplest design which was composed of a blank *working area* window and a *Tool window*. The *Tool window* contained two panels and a widget icon in-between. The lower panel of the *Tool window* contained an object icon template which would be used for creating objects in the working area. The upper panel of the *Tool window* contained utility icons providing modification to the objects created in the working area. A widget icon located between the upper and the lower panels provided a pop-up dialog box containing pre-defined objects with a ready-made functionality. The object-action interface model[19] was used. Users created interface objects on the working area and then assigned a sequence of actions to the objects, such as opening a dialog box all the way down to a series of detailed keystrokes and clicks. CAIBuilder interface was designed to work by pointing and clicking in order to select an object, and clicking on an icon tool in the upper panel of the *Tool window* to execute some commands or apply some characteristics on the selected objects. The program also provided a right click response of popping up a context-sensitive dialog box.

A very essential module of *Auto-script* was develo ped by writing a collection of most commonly-used scripts of actions presented in a window of *Action Property* dialog box containing a list of actions, triggers,

and parameters, which were ready to be assigned to any object. These processes resulted in a working environment which enabled non-programmer users to create highly interactive multimedia applications without programming. These designs were done according to Human-Computer Interaction (HCI) recommendations.[16,19-21] The beta version software was tested in a workshop consisting of twenty-five high school teachers who were neither programmers nor multimedia application developers. These teachers were equally divided into five groups to produce five prototypes of their first CAI lessons. The software feature improvements and software bugs were collected during the two-day workshop. The requested features were then developed and the software bugs were fixed.

### Part III: Testing and evaluating the software

Two workshops of *Producing CAI easily with CAIBuilder* were conducted to test and evaluate the final version software. Pre-test and post-test questionnaires were distributed before and after the workshop which consisted of fifty-three participants who were staff members of various universities. The pre-test and post-test questionnaires were mapped in pairs by markers on the questionnaires.

The pre-test questionnaire was designed to collect information and experience about the workshop's participants, including age, sex, teaching experience, experience in authoring electronic or multimedia lessons, electronic presentation, computer experience and skills, good multimedia application development concepts, instructional design concepts, navigational control, program flow control, and creating students' test items for evaluation.

The post-test questionnaire was designed to collect information regarding knowledge and experience from the workshop, including good multimedia application development concepts, instructional design concepts, navigational control, program flow control, and creating student's test items for evaluation. For this part of the research study, the data were analyzed and presented in percentages and Wilcoxon signed rank test was done to compare the corresponding pre-test questionnaire.

The post-test questionnaire was also designed to evaluate the CAIBuilder's features, including ease of use or user-friendly GUI, the software's functional ability, overall ease of use compared to existing electronic authoring tools (such as ToolBook, Authorware, or Director), the need for software improvements, and the quality of the training manual.

In order to avoid undetermined answers of middle choice, a 4-choice numeric rating scale[13] was used in both the pre-test and post-test questionnaires ranging from 1 (the least), 2 (a little), 3 (a lot), and 4 (the most). Most of the data were analyzed and presented in percentages.

### Part IV: Examples of developing real CAI lessons for classes

Two examples of developing real CAI lessons for classes at the Faculty of Medicine Siriraj Hospital, by participants from the workshops of *Producing CAI easily with CAIBuilder* were *Autonomic Nervous System* and *Physical Modalities*.

The *Autonomic Nervous System* was intended for second-year medical students learning physiology and it was designed to be used in two approaches: (1) as a tutorial containing electronic page turner containing a brief conceptual text and links to related multimedia; and (2) as a pre-test, evaluating the scores and feeding back by showing appropriate content, performing a post-test, and showing a report of the pre-test and post-test scores. The software was developed within approximately 100 man-hours without programming.

The *Physical Modalities* was intended for fifth-year medical students learning physiotherapy. It was developed based on a concept of tutoring students in electronic page turners containing a brief text content and interes- ting multimedia clips, including video clips, sketch pictures, and images of real equipment that made students familiar with and ready for the real-world working environment. The software also contained multiple choice questions for evaluation. The software, including producing multimedia clips, was created within 400 man-hours without programming. More than half of the development time was consumed by multimedia digitization and retouching.

## RESULTS

The CAIBuilder program was developed by utilizing a research and development process. The research part was done by interviewing eight experienced e-courseware developers in order to identify their software features needed to produce CAI by themselves and the needs of GUI (Part I). The features were identified and then used as criteria in developing the program by using the OpenScript language of the Asymetrix ToolBook II Instructor software (Part II). The beta version of the software was tested; its features were enhanced, and bugs were fixed. The final version of the software was then used to conduct two workshops (Part III) to test and evaluate the software including its manual. Part IV demonstrated two CAI lessons by using CAIBuilder.

### Part I: Identifying the electronic authors' requirements and problems

#### 1. Comparison among commonly-used electronic authoring programs

The electronic authoring programs that were commonly-used among the eight interviewees included Asymetrix ToolBook II Instructor, Macromedia Direc tor, and Macromedia Authorware. Four of the authors used ToolBook II Instructor and Authorware; two used ToolBook and Director; and two used Authorware and Director.

#### 1.1 Experience and difficulties in using authoring programs

The keywords observed from the experience and difficulties in using authoring programs of their choice

revealed *page-based concept*, *icon-flow concept*, *movie-based concept* or *time-based concept* and *the way each program created objects.*

The *page-based concept* from ToolBook gives a sense of turning pages from a physical book, a paradigm which is familiar to all academics. Everything the authors want to show on a particular page can be put on the page. The authors can create *title page*, *menu page*, *table of content page*, *content page*, and *index page*. The layout is like a real book. Some difficulties encountered were the *page* and *background* concepts where the objects put in the background can be shared to all pages that use the same background. Putting objects in a background object can actually save a lot of computer memory but this creates confusions for beginners. It is suggested that putting objects in the foreground or on the page is worth considering to reduce confusion for beginners. Creating objects by clicking and dragging could result in creating very small almost invisible objects on the page.

The *icon-flow concept* from Authorware gives a sense of dynamic flow of icons which is quite easy to follow, especially in a linear flow fashion. It is found that, for a linear navigation without too many branches of navigation, the *icon-flow concept* can be accomplished quite easily. However, when there are many decision branches, the *icon-flow concept* can cause some confu-sion; it is quite difficult to follow the program flow. The developers must use a complex manipulation of variables and functions as well as the complex logic expressed in the flowchart in order to accomplish the same interactive tasks as those written easily with scripting languages in ToolBook or Director. However, creating objects by dragging and dropping on the icon flow is considered easy and convenient.

The *movie-based concept* or *time-based concept* from Director gives a sense that the authors are the director conducting *casts* or *actors* acting on a stage in a movie. It is quite easy to kick off actors to act but it is not easy to stop these actors on the stage. Most of the time the authors need to write programs in *Lingo* language to customize each actor to the scene. It is suggested that for beginner authors, programming *Lingo* is not easy. However, it seems that creating animation in Director is a strong point. Creating objects by clicking and dragging could result in creating very small almost invisible objects.

### 1.2 Ease of use of the graphic user interface (GUI)

An overall observation of ease of use of the GUI among ToolBook, Authorware and Director, includes methods of creating objects, modifying objects' properties, using tool windows or palettes and their icons, intuitions of menu bars and helping facilities. It is observed that three out of the four of the authors who used ToolBook and Authorware revealed that using Authorware was easier than using ToolBook. Both authors who used ToolBook and Director revealed that using ToolBook was easier than using Director, and both authors who used Authorware and Director revealed that using Authorware was easier than using Director. Director has the highest number of pop-up windows and palettes which confuse beginners. ToolBook and Authorware have fewer pop-up windows and palettes which confuse beginners less. However, all authors mentioned that all the authoring programs provided too many windows or palettes. It can be concluded that Authorware is the easiest to use, and using Authorware and ToolBook is easier than Director.

### 1.3 The powerfulness of the software

An overall observation of the powerfulness of the software among ToolBook, Authorware and Director, was conducted in terms of Windows events and Windows messages handling, objects' functionality assigned by authors, navigational control and decision branching, repetitive looping, reusability of objects and codes, and programming logic. All of the four authors who used ToolBook and Authorware revealed that ToolBook was much more powerful than Authorware. Both authors who used ToolBook and Director revealed that ToolBook was more powerful than Director, and both authors who used Authorware and Director revealed that Director was more powerful than Authorware. It can be concluded that ToolBook was the most powerful authoring program, and that Director was more powerful than Authorware.

## 2. Key features or requirements of an ideal authoring software

### 2.1 Working paradigm

The authors wanted to have a working paradigm which was familiar to most academics which could be a *page-based concept* or a less familiar but acceptable one like an *icon-flow concept.* The working area of the window should be easy to manipulate, for example, defining the window's size or defining the characteristics of its frame and status bar. More than half of the interviewees required as few windows as possible; the interface objects should be intuitive and need little explanation for use. There should be a right mouse clicked context sensitive pop-up menu or dialog to define objects' properties or functionalities.

### 2.2 Methods of creating object and giving actions to each object

To create objects, all of the authors preferred the *drag and drop* or *click and drag* methods. However, the "click and drag" method in a very small area (the starting point is very close to the finishing point) may result in creating a very small object without being noticed. The *drag and drop* method that creates an object that is small but big enough to be noticed would be more preferable. The size and position of each object can then be resized or repositioned as required by either *click and drag* on the object or, to be precise, by assigning a property assignment dialog box. Other kinds of object properties (e.g., stroke color, fill color, and transparency) should be able to be assigned directly to the object or through a property dialog box.

Every author would like to have multiple methods of giving actions to each object, for example, an automatically assigned script by pointing and clicking from a list of commonly-used actions and filling in more information to fine-tune detailed features if needed, and additional

programming language can be assigned to each object for advanced authors when they need to assign more advanced features to objects.

### 2.3 Navigational controls and hyperlink, and integrating multimedia objects

Seven out of the eight authors would like to have ready-made objects containing some graphics and pre-assigned functions such as *first* button, *previous* button, *next button* and *last* button. The software should also provide sets of commonly-used objects, for example, a navigator set consisting of buttons navigating to the previous or the next pages; a set of multimedia control for playing, pausing and stopping a video, and a set of buttons controlling the playing and recording sound.

Each object should be made available to be assigned a *hyperlink* which takes users to other pages on the same book or file, as well as to other pages on other books or files with some transitional effects, and without programming if possible.

### 2.4 Creating test items for evaluations

All the authors would like to be able to create test items for evaluating students and scoring features. The test items should include true-false, multiple choice question, fill in the blank, and matching. Each test item should be made available to be assigned a weighting score. The software should be able to score test items individually or on demand, as well as scoring the whole test. The software should also be made available to reset individual test items or all the test items to their original state so they can be ready for the next students.

All the authors would like to have a response text area for displaying responses to each answer choice; if possible, each choice item should be able to respond to other formats, such as showing objects or playing multimedia files.

### 2.5 Open-ended features

An open-ended question was asked to all eight interviewees regarding additional features whether they need to have in an authoring software; it revealed the following recommendations.

Five out of the eight authors wanted to have a *password protected mechanism* in order to secure their work from unauthorized alterations.

Four out of the eight authors wanted to have *auto-save which* is a feature of saving their work automatically at a certain defined period of time.

Four out of the eight authors wanted to develop web-based applications, if possible.

Two out of the eight authors would like to have a course management system to collect students' test results and for follow up at a later time.

Two out of the eight authors would like to have a setup program to integrate the CAI and all of its necessary multimedia files to be installed into users' machines.

## Part II: Designing and implementing the software

The features identified by the interviews from Part I were analyzed and priorities of each feature were collected from the eight authors based on ease of use and necessity. The priorities are classified as *Mandatory*, which means very essential or highly needed, and *Desirable*, which means less essential and can be implemented in the next version. Features are described in sections 2.2, 2.3, 2.4, and some features in section 2.5 (the *password protected mechanism* and the *auto-save* were classified as *Mandatory*). Features described in section 2.1 and the rest of section 2.5 was classified as *Desirable*. It was found that most features the authors preferred to have most resemble ToolBook's working environment but it is necessary to customize it to a much simpler working environment and more user-friendly GUI. To reduce program development time, therefore, the CAIBuilder was developed by using OpenScript language in Asymetrix ToolBook II Instructor Version 8.0. The basic concept of software design was to build a simplified version of Asymetrix ToolBook II Instructor which can be used easily with high functionality but without writing scripts. When an author assigns an *Action* to an object by pointing, clicking, and filling in some parameters where necessary, the CAIBuilder will create an automatic script, namely *Auto-script*, for the *Builder*. However, the *Builder* is not confined to the limited set of *Auto-script* but he is free to enter more advanced OpenScript language to the object.

The design of the new authoring software would be *page-based concept* rather than *icon-flow concept*, because most academics are familiar with flipping through and reading paper-based books. Each newly created file is created from a pre-defined template file. There are two modes, namely, *Builder* mode and *User* mode. The *Builder* mode is used by an author who creates new lessons or books or files, and the *User* mode is used by students or users who use the lessons or books or files created by authors. Only in the *Builder* mode can an author assign a password for protecting the author's authority in altering book's content and program logic. In order to reduce the author's confusion in the *Builder* mode, the CAIBuilder provides a menu bar and only one *Tool Window* containing two major panels. The top panel of the *Tool Window* provides icon tools for changing object properties and assigning object functionality through assigning *Action* to the objects in the working area (see Fig 1) The lower panel of the *Tool Window* is used for creating new objects, by dragging an object from the lower panel and dropping it onto the working area. Between the two panels is situated a widget button (Fig 2) containing pre-defined objects with ready-made functionality, such as navigators, navigator panels, multimedia control objects, test item widgets, and scoring and reset widgets.

The concept of creating a new computer-aided instruction required three major steps including: (1) creating objects, (2) changing objects' properties, and (3) giving objects' functionality.

(1) Creating objects is done by dragging and dropping the object from the lower panel of the *Tool Window* or from the *Widget Window* onto the working area.

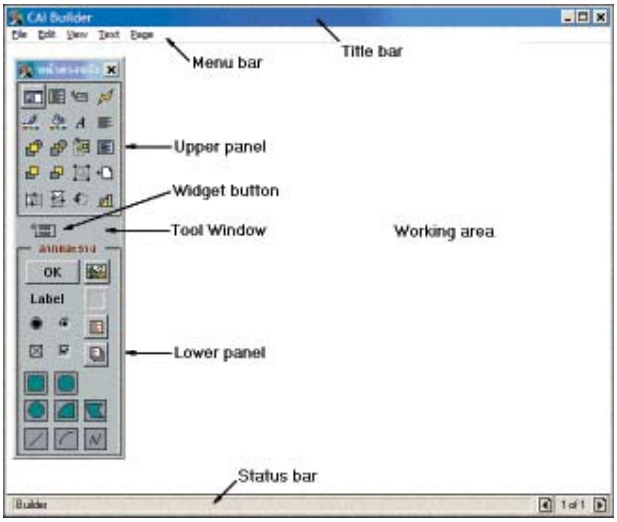


**Fig 1.** The newly created book or file containing 1 page. There are three bars including the very top *Title bar* and beneath it a *Menu bar*; a *Status bar* is located at the bottom. The *Working area* is the area where authors put their objects. The *Tool Window* contains an *Upper panel*, a *Lower panel*, and a Widget button located between the two panels.


**Fig 2.** *A Widget Window* containing a dropdown list of *Navigators (Individual button)*, *Navigation panel*, *Play sound*, *Record sound*, *Quiz : True/False*, *Quiz : Multiple Choice Question*, *Quiz : Fill in the blank*, *Quiz: Matching*, *Scoring a question and reset*, and *Scoring all pages*. These widgets can be created by dragging the objects and dropping them onto the working area.

(2) Changing objects' properties is done by altering properties' values in the context sensitive pop-up *Object Property* dialog boxes (see Fig 3 for a sample of the *Button Property* dialog box).

The properties of each object can be assigned either directly to the object from the interface palette or through an object property dialog box. The properties of each widget can be assigned only through a widget property dialog box.

(3) Giving objects a functionality is done by assignation through the *Object Action* dialog box (Fig 4) Nearly every kind of object in CAIBuilder can be assigned its *Trigger*, which means the event (i.e.,*buttonClick*, *mouseEnter*, and *mouseLeave*) starts the responsive action (such as *Exit from a program*, *Go to the first page*, *Go to the last page*, *Go to next page*, *Go to previous page*, *Go to page menu*, *Go to page... with transition effect*, show or hide objects, control multimedia clips (such as start playing, pause a clip, and stop a clip), and *Frame animation*. In some actions, there must be some parameters to be completed in order to perform correctly.


**Fig 3.** *A Button Property* dialog box shows various properties of a button which can be assigned.

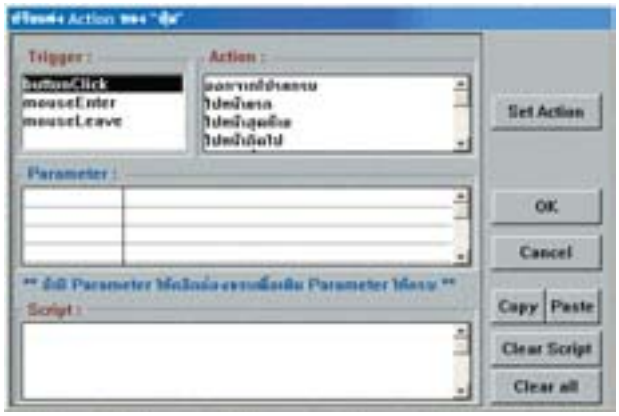


**Fig 4.** An *Action Property* dialog box shows *Trigger*, *Action*, *Parameter*, and *Script* of a button which can be assigned.

For the *book* object, there is a *Window property* dialog box (Fig 5) where the Builder can assign a caption text for the window's title bar, the type of window frame, the language of the menu bar, the visibility of the status bar, the size of the window, the color of hypertext, the time interval for the *auto-save* feature (a property of *save-on-close* of an application), a *Password* protection feature, and an *Action* button (Fig 5). The *Action* button brings up a *Book Action* dialog box whose triggers include *enterApplication* and *leaveApplication* messages. The *enterApplication* message is triggered when an application is evoked or started and it is used to prepare environments or variables ready for a new user. The *leaveApplication* message is triggered when an application is closed and it is used for cleaning up or flushing unsaved data or information.


**Fig 5.** A *Window Property* dialog box shows *Title bar caption*, *Window frame type*, *Menu language*, *Visibility of status bar*, *Window size*, *Hypertext color*, *Auto-save interval*, *Save-on-close feature*, a *Password* protection buttion, and an *Action* button.


**Fig 6.** A *Page Property* dialog box shows *Page name*, *Page number*, *Background name*, *Background color*, *Background image*, *The way a background image is shown*, and an *Action* button.

For the *Page* object, there is a Page *property* dialog box (Fig 6) where the Builder can assign a page name, the order of the page, a background name, a background color, a background graphic, the way the background graphic is shown, and an *Action* button. The *Action* button brings up a *Page Action* dialog box whose triggers include *enterPage* and *leavePage* messages. The *enterPage* message is triggered when the User is about to flip to a page and is used for preparing an appropriate environment or setting up variables of the entered page. The *leavePage* event is triggered when a User is about to close a page or leave to another page or about to close a book in order to clean up the environment or variables.

## Testing of the beta version of CAIBuilder

The twenty-five participants who were high school teachers had little previous experience in using any kind of authoring program except Microsoft PowerPoint. The participants were divided into five groups and each group was responsible for creating one prototype CAI. The first day of the workshop was to teach basic knowledge regarding the working environment of CAIBuilder, how to create new objects, and how to assign object's properties and actions. Each group would write a short storyboard of 5 - 10 pages for each lesson. By the end of the first day, each group had 5 - 10 pages of storyboard and some graphics required by the lessons which were prepared by a supporting graphic team in the evening.

The second day of the workshop was conducted so that the participants could put together and link texts, graphics and multimedia clips to build their application software with CAIBuilder. It was found that each group could finish their prototype CAI in time. There were no obvious obstacles in using CAIBuilder for beginner authors though there were some minor bugs which were all fixed in the final version. The GUI was intuitive and needed very little explanation. The functionality of the program was very fruitful.

## Part III: Testing and evaluating the software

### 1. Data gathered from the pre-test questionnaire revealed the following results.

### 1.1 Demographic data

The number of participants in the sample was fifty-three, with 22.6% male and 77.4 female. The major age groups of the participants were mostly 40-49 years old (41.5%) and 30-39 years old (35.8%); 13.2% were 20-29 years old and 9.5% were 50-59 years old. More than half of the participants (58.5%) had over six years teaching experience while 86.8% of the participants had been using computers for three years and more.

### 1.2 Knowledge, skill and experience in creating multimedia application and concepts

The findings are summarized in Table 1, which shows that over 75% of the participants had a lot of mouse skill; 62.2% had a little experience in creating electronic presentations and about one-third had a lot of such experience. Over 90% of the participants had little experience in creating multimedia application, using multimedia authoring programs, creating computerized test items, creating navigational controls and directing program flows. Over 90% of the participants also had little knowledge about multimedia application development concept and instructional design concept. There were 83% who had little confidence in creating multimedia applications by themselves; however, about 58.5% of the participants thought that CAIBuilder program was easy-to-use within two days.

### 2. Data gathered from the post-test questionnaire revealed the following results.

### 2.1 Knowledge, skills and experience in creating multimedia application and concepts

Table 2 shows that over 88% of the participants gained more knowledge about multimedia application development concepts and instruction design concepts (score = 3 and 4). Most of them (about 85-95%) gained confidence in creating computerized test items, creating navigational controls and directing program flows, and creating multimedia applications by themselves (score = 3 and 4).

Wilcoxon signed rank test which compares results of knowledge about multimedia application development concepts and instructional design concepts, confidence in creating computerized test items, creating navigational controls and program flows, and creating multimedia application by oneself (items 4, 5, 7, 8 and 9 of Table 1 compared with items 1-5 of Table 2, respectively) revealed a statistically significant increase from *the least*

**TABLE 1.** Summarized findings of the pre-test quesionnaires regarding participants' knowledge, skills, experience, and confidence on creating multimedia applications. The figures, representing the majority of answers, are displayed in dark gray; the second from the most majority figures are in light gray. (1 means 'the least', 2 means 'a little', 3 means 'a lot', and 4 means 'the most')

| | Knowledge, skill, experience, and confidence | 1 | 2 | 3 | 4 |
|---|---|---|---|---|---|
| 1. | Using mouse skill | 3.8 | 20.8 | 50.9 | 24.5 |
| 2. | Experience in creating electronic presentations | 9.4 | 52.8 | 35.8 | 1.9 |
| 3. | Experience in training to create multimedia applications | 54.7 | 35.8 | 7.5 | 1.9 |
| 4. | Knowledge about multimedia application development concepts | 50.9 | 43.4 | 5.7 | 0 |
| 5. | Knowledge about instructional design concepts | 67.9 | 30.2 | 1.9 | 0 |
| 6. | Experience in using ToolBook, Authorware or Director | 67.9 | 26.4 | 5.7 | 0 |
| 7. | Experience in creating computerized test items | 71.7 | 26.4 | 1.9 | 0 |
| 8. | Experience in creating navigation controls and program flows | 79.2 | 18.9 | 1.9 | 0 |
| 9. | Confidence in creating multimedia application by oneself | 39.6 | 43.4 | 15.1 | 1.9 |
| 10. | Expectatio of ease of use of the CAIBuilder program | 7.5 | 22.6 | 58.5 | 11.3 |

(score = 1) or *a little* (score = 2) to *a lot* (score = 3) or *the most* (score = 4) at $p < 0.0001$.

### 2.2 Key features of CAIBuilder program

CAIBuilder was easy (score = 3) to use for 90.6% of the participants; 7.5% found the program the easiest (score = 4) to use, and only 1.9% found it difficult (score = 2) to use. Of those who used ToolBook, Authorware, or Director, 78.6% found the CAIBuilder was easier to use (score = 3), and 21.4% found that CAIBuilder was much easier to use (score = 4).

Majority of the participants, 67.9% (score = 3) found that the GUI of CAIBuilder was easy, good and user-friendly, and 20.8% (score = 4) found it the easiest, best and most user-friendly authoring software. There were only 11.3% that found CAIBuilder a fair software (score = 2). Most of the participants (92.5%) found CAIBuilder a powerful authoring software (score = 3), and 5.7% found it the most powerful (score = 4). 86.8% of the participants found CAIBuilder was easy to learn (score = 3) for inexperienced e-courseware developers, and 9.4% found it the easiest to learn (score = 4). 56.6% of the participants found the CAIBuilder training manual of good quality (score = 3) and 39.6% found it very good (score = 4).

### 2.3 Others

Those who had some experience with ToolBook stated that CAIBuilder was much easier to use than ToolBook with nearly equally powerful features, such as creating interaction and feedback.

73.6% of the participants expressed their intention to come back to an intermediate or advanced CAIBuilder workshop, and 26.4% wanted to try using the software for a little longer.

Some suggestions were added to improve the software features or more provisions of a course management system to collect students' test results, more animation options, and more Auto-scripts.

## Part IV: Examples of developing real CAI lessons for classes

The *Autonomic Nervous System* was developed by using a pre-test of multiple choice questions. There were ten questions and each question contained five choices. The software was used for aiding in teaching medical physiology to second-year medical students at the Faculty of Medicine, Siriraj Hospital. The multiple choice questions needed one best answer from the students. After the students were done with the pre-test, the program marked the scores and identified the content which the students might misunderstand. The program gave feedback to the students by showing the corresponding content for the students to review, resulting in more understanding. After the students answered ten questions of the post-test, the marking scores were calculated and presented. The report of pre-test and post-test scores demonstrated that all of the students (about 190) got a better score on the post-test (above 80% correct answers) after the review.

The *Physical Modalities* was used by fifth-year medical students at the Faculty of Medicine Siriraj Hospital, before and after learning physiotherapy. The CAI lesson provided a brief content text and interesting multimedia. The content was relevant, brief and focused on the important points of the knowledge. Video clips were very useful in helping students visualize subjects receiving manipulation or therapy in different physical

**TABLE 2.** Summarized findings of the post-test quesionnaires regarding participants' knowledge and confidence in creating multimedia applications. (1 means 'the least', 2 means 'a little', 3 means 'a lot', and 4 means 'the most')

| | Knowledge, and confidence | 1 | 2 | 3 | 4 |
|---|---|---|---|---|---|
| 1. | Knowledge about multimedia application development concepts | 0 | 9.4 | 83 | 7.5 |
| 2. | Knowledge about instructional design concepts | 0 | 11.3 | 81.1 | 7.5 |
| 3. | Confidence in creating computerized test items | 0 | 5.7 | 77.4 | 17 |
| 4. | Confidence in creating navigational controls and program flows | 1.9 | 13.2 | 69.8 | 15.1 |
| 5. | Confidence in creating multimedia application by oneself | 1.9 | 5.7 | 81.1 | 11.3 |

modalities before the class, and the CAI lesson was also helpful for review after classes. The questions and immediate feedback of correct answers to students were helpful in enhancing students understanding of the content.

## DISCUSSION

Although this study did not extensively explore the experience and difficulties of electronic authors in using the commonly-used authoring programs, such as ToolBook, Authorware, and Director, it is worth noting that the sample group gave information mostly in accordance with the studies of Pichitpornchai[22] and Dalgarno.[6] In Dalgarno's study of analysis of the available authoring tools, he compared the ToolBook version 3.0, Authorware version 3.0, and Director version 4.04. Dalgarno found Authorware and ToolBook were the two easiest authoring tools on condition that the multimedia content created by using Authorware did not contain sophisticated branching logic. He found that ToolBook was the easiest authoring software when the content contained more interactive features and the non-programmer authors could develop without programming. Although the software versions studied by Dalgarno and the software versions studied in this research were different, the results were mostly in harmony because each software was consistent in its conceptual design, and with each new emerging version, the software was enhanced in its capability.

The present study revealed that the Authorware is the easiest to use and learn but it is the least powerful authoring tool when compared to the other two tools because it does not provide scripting language. Director is quite capable of animation control and color management but has fewer powerful Windows' events and messages handling; to control objects, the author needs to write a script. ToolBook can provide the most functionality with less effort when compared to Director, but it is found harder to start to learn when compared with Authorware. Computer-based learning materials with interaction and feedback can be created easily with ToolBook but rather complex variables and functions as well as developing algorithms expressed as complex flowcharts make Authorware more difficult to use than ToolBook. These findings were in accordance with Dalgarno's study in 1996 and a review of interactive multimedia authoring software by Jones (cited in Dalgarno, 1996) showed that ToolBook was the first choice of commercial e-courseware developers whereas Director was the second choice.

The *page* paradigm is among the most common mind-sets for most academics because all academics have experienced reading paper books. Therefore, implementing the CAIBuilder's working environment based on the *page* paradigm will presumably reduce the learning curve for most electronic authors.

The research and development strategy of CAIBuilder is considered to comply with standard software engineering.[15] It was started by identifying users' requirements and problems, analyzing and designing, coding and implementing, testing and debugging, and documenting and maintenance. The beta version of CAIBuilder was successfully tested by 25 inexperienced users who guaranteed its usability, learning curve, powerfulness, and bug gathering and fixing process.

In general, the behavior of these users who are familiar with Windows interface, such as *selecting an object* by pointing and clicking the object, *then applying some features or characteristics* by clicking on an icon in a tool bar or a menu item from a menu bar is normal. At some other times, the users' right-clicking action gives rise to a context sensitive right-clicked menu. By recognizing such familiarity to Windows' working environment and recognizing the diversity of users' behavior, CAIBuilder provided the same way of selecting an object and applying characteristics to the selected object. CAIBuilder also provided a right-clicked context sensitive pop-up window in an extremely consistent way, conforming to the golden rule of interface design[19] and the human computer interaction approach.[16]

The design of CAIBuilder, which contains a minimal pop-up window or palette, could reduce confusion in beginner developers. The values entered in the fields required in each dialog box were checked for errors and those errors were prevented. This error-free data entry feature has also been designed following the golden rule of interface design.[19] The functionality or actions of interface objects in the working area were assigned by filling in the context sensitive pop-up dialog boxes. This design reduced the authors' difficulties in using the right dialog box at the right time. Because the entire GUI designed in CAIBuilder was developed according to the author's requirements gathered from Part I of the study, it can be appropriately concluded that all GUI is suitable and convenient to use for most beginners of e-courseware developers. The CAIBuilder is also considered powerful because it imitates the powerfulness of ToolBook's scripting language but the CAIBuilder is easier to use because it has a customized auto-script and easy-to-use interface. The advanced authors of CAIBuilder have an option of advanced scripting in OpenScript language if they are experienced in the scripting language.

The final testing and evaluation of the CAIBuilder conducted in two workshops for fifty-three non-programmer electronic authors confirmed the usability, learning curve, and powerfulness of the software. When considering CAIBuilder key features by summing the scores of 3 and 4, representing *easy* and *the easiest*, a total of 96-98% of parti- cipants indicated that CAIBuilder was easy or the easiest to use compared with ToolBook, Authorware, or Director, and CAIBuilder was easy to learn for non-programmers. Sum scores of 3 and 4 showed that 98.2% of participants indicated that CAIBuilder was powerful (score = 3) and the most powerful (score = 4). As for the GUI of CAIBuilder, a total 88.7% of participants indicated that it was user-friendly (score = 3) and the most user-friendly (score = 4).

The comparison of knowledge about multimedia application development concepts and instructional design concepts before and after the workshops represented the effectiveness of the training course. In the workshop, as well as in some answers in the post-test open-ended questionnaire, the design of CAIBuilder which is intuitive to most participants partly helped participants understand more instructional design and GUI design.

After a two-day workshop among inexperienced e-courseware developers, the confidence in creating multimedia application by the participants themselves, the confidence in creating navigational controls and program flows, and the confidence in creating computerized test items all increased with a statistical significance. These results show somewhat the ease of use and powerfulness of using the CAIBuilder for creating multimedia lessons. When three two-day workshops of *Produce CAI Easily with ToolBook II Instructor* were conducted by the researcher at the Siriraj Annual Congress in the years 1999, 2000, and 2001, to about 100 participants, it was found that the learning curve of ToolBook was harder and over 80% of workshop participants had less confidence in such areas.

Two examples of real CAI lessons have been created with CAIBuilder and have been used by second-year medical students and fifth-year medical students at the Faculty of Medicine Siriraj Hospital. The features used in the two softwares were an electronic presentation such as an electronic page turner, as well as multiple choice questions used for pre-test and post-test evaluation. CAIBuilder can be used to create test items for evaluating learners easily. The softwares were developed with ease and in a reasonable time frame without programming skills whereas the multimedia clips consumed most of the total development time.

It is worth mentioning that the limitation of ToolBook and CAIBuilder is that they are only available for personal computers (PC). Virtually, Macintosh's market share has been small when compared to that of the PC. There are also increasing needs to provide computer-based learning materials across the Internet. A plug-in of *Neuron* can be used in conjunction with web browsers to show ToolBook-based or CAIBuilder-based multimedia applications. The present version of ToolBook version 8 or 9 can also generate HTML-based application but a limited number of features can be implemented when compared to the native ToolBook application.

## CONCLUSION

It is obvious at present that good CAIs will certainly help reduce the teaching and learning burden of teaching staff. Further more, good CAIs can make lessons more interesting and understandable. The shorter the learning curve is, the better the students can learn at their own pace. From the present study it can be concluded that the CAIBuilder has been developed according to research requirements and by using a standard software engineering methodology. It provides easy-to-use GUI, more powerful features with less effort, and additional options of scripting when the electronic authors are ready to use the scripting language. CAIBuilder can be a good software of choice for inexperienced e-courseware developers or non-programmer authors to create good quality and interactive CAI lessons with feedback in a fairly quick time frame.

## REFERENCES


1. ยุวรรณ์ดา พานทอง. รมว.สธ. ยอมรับปัญหาขาดแคลนแพทย์ส่งผลกระทบต่อการดำเนินโครงการ 30 บาท. สำนักข่าวกรมประชาสัมพันธ์ ส่วนข่าววิทยุ 24 กรกฎาคม พ.ศ. 2547 (cited 17 มกราคม พ.ศ. 2548); Available from: http://www.radionewsprd.com/previewnews.php?news_id=254707230082
2. สำนักงานสถิติแห่งชาติ. สถิติสุขภาพ 4.7 บุคลากรทางการแพทย์ ตาราง 1 จำนวนและอัตราส่วนบุคลากรทางการแพทย์ และสาธารณสุข พ.ศ. 2536 - 2544. สำนักงานสถิติแห่งชาติ พ.ศ. 2547 (cited 5 มกราคม พ.ศ. 2548); Available from: http://www.nso.go.th/nso/data/data23/stat_23/toc_4/4.7-1.xls
3. พรภัสสร ปิ่นสกุล. รองเลขาธิการคณะกรรมการการอุดมศึกษา ระบุ ประเทศไทยยังต้องการแพทย์เพิ่มในทุกสาขา. สำนักข่าวกรมประชาสัมพันธ์ ส่วนข่าววิทยุ 4 กรกฎาคม พ.ศ. 2547 (cited 17 มกราคม พ.ศ. 2548); Available from: http://www.radionewsprd.com/previewnews.php?news_id=254706270117
4. Pichitpornchai C. IT and Medical Education in South-East Asia: Possible International Cooperation on Medical Education. In: The First Forum on International Cooperation and Research of Medical Education. Yayoi Conference Hall, Tokyo University, Japan: The International Research Center for Medical Education; 2001.
5. พิเชฐ ดุรงคเวโรจน์, ชุตินันท์ แสงหิรัญ, ศุเรนทร์ ฐาปนางกูร, ชัยยุทธ ปัญญสวัสดิ์สุทธิ์, อมรา รสสุข. รายงานการวิจัย: นโยบายและยุทธศาสตร์การพัฒนาเทคโนโลยีสารสนเทศเพื่อการศึกษาของประเทศไทย. กรุงเทพมหานคร: สำนักงานคณะกรรมการการศึกษาแห่งชาติ; กรกฎาคม พ.ศ. 2543.
6. Dalgarno B. Authoring Your Own Learning Resources: An Analysis of the Tools Available. Proceedings of the Australian Computer Education Conference 1996 (cited 2005 January 4); Available from: http://farrer.riv.csu.edu.au/~dalgarno/ucwebpages/articles/acec/dalg.htm
7. Lee WW, Owens DL. Multimedia-Based Instructional Design: Computer-Based Training, Web-Base Training, Distance Broadcast Training. San Francisco, California: Jossey-Bass Pfeiffer; 2000.
8. Piskurich GM. Rapid Instructional Design: Learning ID Fast and Right. San Francisco, California: Jossey-Bass Pfeiffer; 2000.
9. Pichitpornchai C, Sapsomboon B. Producing Quality CAI: Theory To Practice. In: 42nd Siriraj Scientific Congress (Accredited Medicine) Faculty of Medicine Siriraj Hospital; 2002 March 4-8, 2002; Faculty of Medicine Siriraj Hospital, Mahidol University: Faculty of Medicine Siriraj Hospital; 2002:125-126.
10. Pichitpornchai C. E-Learning and Instructional Design. In: 43rd Siriraj Scientific Congress: From Advanced Medicine to the Standard of Care Faculty of Medicine Siriraj Hospital; 2003 March 3-7, 2003; Faculty of Medicine Siriraj Hospital, Mahidol University: Faculty of Medicine Siriraj Hospital; 2003:24.
11. พรพิไล เลิศวิชา. สื่อประสมเพื่อการศึกษา. ใน: มัลติมีเดียกับการปฏิรูปการเรียนการสอนในประเทศไทย; โรงแรมสยามซิตี้ กรุงเทพมหานคร; 24 กุมภาพันธ์ พ.ศ. 2540.
12. Pichitpornchai C, Sapsomboon B, Pausawasdi N. Educational Media: Tools And Technology. In: 42nd Siriraj Scientific Congress (Accredited Medicine) Faculty of Medicine Siriraj Hospital; 2002 March 4-8, 2002; Faculty of Medicine Siriraj Hospital, Mahidol University: Faculty of Medicine Siriraj Hospital; 2002:123-4.
13. บุปผชาติ ทัญหิกรณ์, สุกรี รอดโพธิ์ทอง, ชัยเลิศ พิชิตพรชัย, โสภาพรรณ แสงศัพท์. ความรู้เกี่ยวกับสื่อมัลติมีเดียเพื่อการศึกษา. กรุงเทพมหานคร: ศูนย์พัฒนาหนังสือ กรมวิชาการ กระทรวงศึกษาธิการ; พ.ศ. 2544.
14. Pichitpornchai C. Computer-Based Training (CBT) authoring with Multimedia ToolBook. In: Siriraj Scientific Congress: On the occasion of the 50th Anniversary (Golden Jubilee) Celebrations of His Majesty's Accession to the Throne; 1996 March 4-8, 1996; Faculty of Medicine Siriraj Hospital, Mahidol University: Faculty of Medicine Siriraj Hospital; 1996:31-2.
15. Meyers BC, Oberndorf P. A Framework for the Specification of Acquisition Models. (pdf file) 2001 December (cited 2004 January 5); Available from: http://www.sei.cmu.edu/publications/documents/01.reports/01tr004.html and ftp://ftp.sei.cmu.edu/pub/documents/01.reports/pdf/01tr004.pdf
16. Preece J, Rogers Y, Sharpm H, Benyon D, Holland S, Carey T. Human-Computer Interaction. Wokingam, England: Addison-Wesley Publishing Company; 1994.
17. Click2Learn.com. Programming in OpenScript ToolBook II Instructor version 8.0. Bellevue, Washington; 2000.

18. Click2Learn.com. User Guide ToolBook II Instructor version 8.0. Bellevue, Wash ington;2000.
19. Shneiderman B. Designing the User Interface: Strategies for Effective Human-Computer Interface. 3rd ed. ed. Reading, Massachusetts: Addison-Wesley; 1998.
20. Eberts RE. User Interface Design. Prentice Hall International Editions ed. Englewood Cliffs, New Jersey: Prentice Hall International Inc.; 1994.
21. ณอมพร เลาหจรัสแสง. หลักการออกแบบและการสร้างคอมพิวเตอร์ช่วยสอนด้วยโปรแกรม Multimedia ToolBook. กรุงเทพมหานคร: ศูนย์หนังสือจุฬาลงกรณ์มหาวิทยาลัย; พ.ศ. 2542.
22. Pichitpornchai C. Research report: A comparison study of commonly-used multi media authoring tools. Monash University, Melbourne, Australia; November 1995.

## บทคัดย่อ

# การวิจัยและพัฒนา CAIBuilder: นวัตกรรมของโปรแกรมสำหรับสร้างสื่ออิเล็กทรอนิกส์เพื่อการศึกษาได้โดยง่าย

**ชัยเลิศ พิชิตพรชัย พ.บ., ปร.ด.**
*ภาควิชาสรีรวิทยา, คณะแพทยศาสตร์ศิริราชพยาบาล, มหาวิทยาลัยมหิดล, กทม. 10700. ประเทศไทย*

ในช่วงหลายปีที่ผ่านมา นักการศึกษานิยมใช้การเรียนรู้ผ่านสื่ออิเล็กทรอนิกส์ช่วยในการสอนเพิ่มมากขึ้น แต่เป็นที่น่าเสียดายว่าสื่ออิเล็กทรอนิกส์เพื่อการศึกษาส่วนใหญ่ไม่สามารถสนองความต้องการของนักการศึกษาได้ทั้งหมด จึงเกิดความจำเป็นในการสร้างบทเรียนอิเล็กทรอนิกส์ด้วยตนเองโดยอาศัยโปรแกรมสร้างสื่ออิเล็กทรอนิกส์ที่มีในตลาด อย่างไรก็ตาม พบว่ามีความยากลำบากในการเรียนรู้และใช้งานโปรแกรมเหล่านั้น งานวิจัยนี้มีจุดประสงค์เพื่อค้นหาปัญหาการเรียนรู้และการใช้งานโปรแกรมสร้างสื่อการสอนอิเล็กทรอนิกส์ พร้อมทั้งสอบถามคุณลักษณะของโปรแกรมหรือความต้องการเพิ่มเติมและใช้ผลการวิจัยในเบื้องต้นนี้ในการพัฒนานวัตกรรมโปรแกรมชื่อ CAIBuilder โดยใช้กระบวนการทางวิศวกรรมซอฟต์แวร์ชื่อ waterfall model ประกอบกับการใช้ทฤษฎีส่วนต่อประสานระหว่างมนุษย์กับคอมพิวเตอร์ และแบบจำลองวัตถุกับภาคแสดงในการออกแบบและพัฒนา และเขียนด้วยภาษา ToolBook OpenScript® เมื่อได้ทดสอบซอฟต์แวร์ที่เป็นเบต้าเวอร์ชั่นเพื่อทดสอบการทำงาน แก้ไขข้อผิดพลาดการทำงาน และเขียนโปรแกรมเพิ่มเติมคุณลักษณะบางประการแล้ว จึงทดสอบซอฟต์แวร์เวอร์ชั่นสุดท้ายโดยการจัดการอบรมเชิงปฏิบัติการสองครั้งให้แก่นักการศึกษาหรืออาจารย์ในมหาวิทยาลัยที่ไม่ใช่นักเขียนโปรแกรมจำนวน 53 คน การวิจัยประกอบด้วยการประเมินโดยใช้แบบสอบถามก่อนและหลังการฝึกอบรมเพื่อเก็บข้อมูลทั่วไปเกี่ยวกับผู้เข้ารับการอบรม รวมทั้งความรู้ แนวคิด ทักษะ ประสบการณ์และความมั่นใจเกี่ยวกับการสร้างบทเรียนมัลติมีเดีย และข้อคิดเห็นเกี่ยวกับคุณลักษณะของโปรแกรม CAIBuilder วิเคราะห์ผลจากการสำรวจโดยรายงานเป็นร้อยละและใช้ Wilcoxon signed rank test ในกรณีที่มีการเปรียบเทียบ ผลการวิจัยพบว่ามากกว่าร้อยละ 85 ของผู้เข้ารับการอบรมพบว่าโปรแกรม CAIBuilder มีความเป็นมิตรกับผู้ใช้ ง่ายต่อการเรียนรู้และใช้งาน แต่มีความสามารถมาก ผลการวิเคราะห์ข้อมูลโดยใช้ Wilcoxon signed rank test พบว่า หลังจากผ่านการอบรม พบว่าผู้เข้ารับการอบรมมากกว่าร้อยละ 80% มีความรู้เพิ่มขึ้นและมีความมั่นใจเพิ่มขึ้นอย่างมีนัยสำคัญทางสถิติที่ $p < 0.0001$ ในด้านการสร้างบทเรียนมัลติมีเดียด้วยตนเอง การสร้างข้อสอบอิเล็กทรอนิกส์ การสร้างปุ่มควบคุมการทำงานและการควบคุมทิศทางการทำงานของโปรแกรม ตัวอย่างสื่อการศึกษาอิเล็กทรอนิกส์ที่พัฒนาขึ้นจากการใช้โปรแกรม CAIBuilder ได้แก่ เรื่อง *Autonomic Nervous System* และ *Physical Modalities* ซึ่งปัจจุบันใช้สำหรับการเรียนการสอนวิชาสรีรวิทยาและวิชากายภาพบำบัดตามลำดับ ให้แก่นักศึกษาแพทย์ที่คณะแพทยศาสตร์ศิริราชพยาบาล กล่าวโดยสรุป CAIBuilder เป็นนวัตกรรมที่เกิดจากการวิจัยและพัฒนา เป็นโปรแกรมที่นักการศึกษาที่ไม่ใช่นักเขียนโปรแกรม สามารถเรียนรู้ การใช้งานได้ง่ายและมีความสามารถสูง สามารถใช้ในการสร้างบทเรียนคอมพิวเตอร์ที่มีคุณภาพดี มีปฏิสัมพันธ์และตอบโต้กับนักศึกษาได้เป็นอย่างดี และสามารถสร้างได้ในเวลาที่รวดเร็วโดยไม่ต้องเขียนโปรแกรม