Inventions on user configurable GUI- A TRIZ based analysis

Umakant Mishra
Inventions on user configurable GUI
A TRIZ based analysis

By- Umakant Mishra, Bangalore, India
http://umakantm.blogspot.in

Table of Contents

1. Introduction ..................................................................................................................2
  1.1 Need for user configurable GUI .............................................................................3
  1.2 A TRIZ based analysis of implementing user configurability ...............................3
  1.3 The Ideal Final Result ............................................................................................5
  1.4 Automatic configuration- a step beyond user configuration .................................6

2. Inventions on user configurable graphic interfaces .................................................6
  2.1 Visual interface between user and computer system .............................................6
  2.2 System and method for configuring a graphic interface .......................................7
  2.3 Method and apparatus for facilitating operator reconfiguration of a graphical user interface in a data processing system .........................................................8
  2.4 Method and apparatus for providing conditional cascading in a computer system graphical user interface .................................................................................9
  2.5 Menu bar Editor .....................................................................................................10
  2.6 Method and system for adding buttons to a toolbar ............................................11
  2.7 Programmable user interface elements .................................................................12
  2.8 Grouping of computer operations ..........................................................................13
  2.9 Method and system for the dynamic customization of graphical user interface elements .............................................................................................................13
  2.10 System and method for customizing appearance and behavior of graphical user interfaces ........................................................................................................14
  2.11 System and method for displaying data using graphical user interface control elements .........................................................................................................15
  2.12 Method, apparatus, system and computer program product for a user-configurable graphical user interface .................................................................16

3. Summary .....................................................................................................................17

Reference to patents: ....................................................................................................18
Other references: .........................................................................................................19
1. Introduction

The graphic user interface is a revolutionary development over the prior art technique of the command line interface. The graphic user interface displays various program elements (such as, commands, functions, internal operations etc.) through friendly graphic elements (such as, buttons, icons etc). This helps the user to do even complex operations through simple means like clicking buttons or dragging various graphic elements.

However, designing a user interface is not as simple as it appears to be. The developer has to keep various aspects of user requirement in mind while building the user interface. The typical aspects considered are,

⇒ What options to provide in menus and toolbars, what buttons and other GUI elements to be provided.

⇒ If there are a large number of options or GUI elements how to organize the options within the limited screen space. The graphic elements, although useful, should not obscure valuable screen space.

⇒ How to manage the advanced GUI options. The advanced options should not confuse the novice user.

⇒ Where to place the graphic elements (like, icons, buttons, toolbars etc.), whether at the top of window, or side of window or at any other place.

⇒ What picture to place on icons? What should be the size of the icons and other graphic elements? What should be the foreground and background color of various graphic elements?

⇒ What should be the order of displaying the elements? For example, the more frequently used item (or rather the more likely used item) may appear at the top a menu for ease of selection.

⇒ The GUI should facilitate easy of navigation and ease of searching. The user should not face difficulties in searching a desired function.

Thus we see that the GUI is built for the user. Hence, the GUI should ideally contain all the features that the user requires and should not contain any of the features that the user does not require. But this creates a complex situation, as the requirements are different for different users. An extremely well designed user interface may fail to meet some expectations of a specific category of users. Besides, the GUI meeting the requirements today may not meet the requirements tomorrow because of changes in user mood and expertise.
1.1 Need for user configurable GUI

Generally the menu items, button definitions etc. are all pre-coded and built into the system by the developer. Even the experienced developer cannot realistically anticipate what options will be required by the users at a future point of time. This leads to non-inclusion of some elements that is considered useful by the end user thereby resulting in user frustration.

If the developer is generous to include all possible functions that the end user might need in the future, that increases the number of elements and clutter the GUI screen thereby leading to user confusion.

Practically it is the user who knows what functions or GUI elements he requires. Hence it is better that the user himself decides on what options to retain or eliminate and configure the GUI accordingly.

The same holds true for the placement of GUI elements. An arbitrary placement of icons, buttons and toolbars may obscure valuable data on the screen. It is the end user who can best decide the placement of buttons and toolbars depending on the availability of space on his computer screen.

Similarly there are many other aspects of a GUI, such as size, appearance, color and other properties of GUI elements, which may be preferred to be set differently by different users. These considerations lead to the concept of user configurable GUI. **As it is ultimately the user who knows his requirement, it is better that the user himself configures the GUI according to his requirement.**

1.2 A TRIZ based analysis of implementing user configurability

Although “user configurability” is a powerful solution to address the monotonous and arbitrary display of GUI elements, the question arises how to implement user configurability. How can the user modify the picture, size, color, location, appearance and other aspects of the GUI elements? How can the user organize the available features for easier and faster access? How can the user add new functions to the software that is not provided by the developer?

**Possible solution:** The user should define and incorporate the additional functions those are required by him.

**Contradiction:** The user cannot modify the definition of menus, buttons, toolbars or other GUI elements, as he does not have the source code. The developer cannot provide the source code to the user, as that will reveal his code confidentiality and affect his business. Putting differently, the user knows exactly what he needs, but he does not have access to the source code, the developer has access to the source code, but he cannot anticipate the exact need of the user at a future period in time.
**Possible solution:** The developer provides the source code to the user as in case of GNU public license, so that the user can make necessary alterations.

**Contradiction:** Modifying source code would require sound knowledge on programming language and techniques. Although it is desirable that the user should be able to alter the GUI elements and their functionalities, it is not desirable for the user to learn the complicated programming languages.

**Possible solution:** The developer should provide all possible options in the menubar, toolbar and other places, so that the user will not feel the absence of any required functionality (**Principle-16: Partial or excessive action**).

**Contradiction:** Providing all possible functions will increase the number of menu items, command buttons, toolbar buttons and other GUI elements, which will cause user confusion. The GUI should contain maximum number of options to meet all possible user requirements, but simultaneously it should contain minimum number of options with a view to avoid user confusion.

**Possible solution:** Each graphic item in the GUI could be adequately described (may be using text) to avoid user confusion (**Principle-8: Counterweight**).

**Resulting problem:** Textual description of buttons will require more screen space and occupy a large area on the display device. If the options are too many, as it may happen in large applications, they may even occupy the whole display area.

**Possible solution:** The developer should pre-build all possible functionalities (**Principle-10: Prior action**) and provide a customization interface (**Principle-24: Intermediary**), so that the user can customize the user interface (**Principle-15: Dynamize**) without getting into complicated programming.

**Possible solution:** The developer should classify the pre-built functions into default and advanced (**Principle-1: Segmentation**). The default options can be displayed in the beginning and the advanced options can be displayed on request (**Principle-15: Dynamize**).

**Possible solution:** The user interface should have easy reconfiguration features so that the user can create buttons by (dragging and) dropping controls on to the toolbar or similar designated area (**Principle-15: Dynamize**) and similarly remove buttons by dragging them out of the GUI (**Principle-34: Discard and recover**).

**Possible solution:** The application should provide features to group or combine multiple toolbar buttons (**Principle-5: Merging**), define user functions as in MS Excel macros (**Principle-15: Dynamize, Principle-37: Expansion**), create programmable user interface elements as in Apple Mac (**Principle-15: Dynamize**) and create advanced buttons like “To be printed” button which can print the marked pages at a later period (**Principle-38: Enrich**).
**Resulting problem:** The same user may require different settings from time to time. Besides different users will prefer different settings. When the GUI is customized by one user, it becomes unusable by another user and the other user has to change all the settings all along again.

**Possible solution:** the customized settings of different users can be stored in separate configuration files (*Principle-10: Prior action*). The user can just select his configuration file and implement the settings (*Principle-3: Local quality*) without doing the whole exercise again and again.

**1.3 The Ideal Final Result**

Ideally the GUI should contain **all** those buttons (or menu items or other GUI elements) and **only** those buttons (or menu items or other GUI elements) that the user needs. Besides the GUI elements should not obscure the display of data, should take no time for searching and should create no confusion in any level of user (**Ideal Final Result**).

Although the IFR is difficult to achieve, it helps to draw a set of desired results in the line of IFR, which may be described as follows.

- The GUI should contain all the options required by the user.
- The GUI should not contain unnecessary items that may cause confusion.
- The (meaning and functionality of) GUI elements should be self-explanatory, creating no confusion even to the novice user.
- The GUI elements should occupy minimum screen space thereby leaving the valuable workspace to be used by other purposes.
- The user should be able to add or remove or alter the functionalities in the graphical user interface by simple drag and drop operations without writing complicated programs.
- The GUI elements should be self-exploring or the user should do minimum effort to locate a desired function.
- The GUI should be easy for selection and navigation.
- The user should be able to group similar functions (like toolbar buttons) for ease of searching.
- The user should be able to combine multiple functions to execute complicated operations in one go.
1.4 Automatic configuration- a step beyond user configuration

Although user customization is a solution to address the rigidity of a graphical user interface, it is not the ultimate solution. We can understand that the user interface needs to be customized for the user, but not necessarily by the user. Customizing user interface will certainly require certain amount of skill, time and effort that every user may not afford.

The problem can be explained in the form of a contradiction. “The user should be able to configure the user interface but he should not spend his valuable time and effort for doing the job. In other words, the interface needs to be customized for the user but the user should not bother doing the job” (Contradiction).

The above contradiction can be addressed by the auto configuration features (Principle-25: Self service). In auto-configurable GUI, the elements and features of GUI are automatically configured to meet the requirement of the user (may be based on usage pattern). This feature of the GUI is analyzed in a separate article “Inventions on auto-configurable GUI- A TRIZ based analysis” by the same author.

2. Inventions on user configurable graphic interfaces

2.1 Visual interface between user and computer system (US Patent 4692858)

Background problem

Generally the menu items are built into the menu system at the time of software development. As the source code is not available to the user, a typical user cannot modify the program to insert or alter items in the menu. Similarly the function associated to a particular mouse button is also fixed. But in some cases it may be desirable to vary the function of a mouse button in order to cater to specific needs of a user.

Solution provided by the invention

Redford et al. disclosed a method (Patent 4692858, assigned by Trillian Computer Corporation, issued Sep 1987) where the user predefines various tasks by using some code that are compiled and stored in the memory. Each of these tasks has specific names for easy identification.

When the user presses a key or a button the predefined task names are displayed on the screen in an overlay display window. The user selects a task by moving the cursor or mouse pointer and corresponding input signals are sent to the computer.
Inventions on user configurable GUI, by Umakant Mishra

TRIZ based analysis
The user cannot alter the functionalities of menu items, buttons etc. without changing the source code. But the developer cannot give the source code to the user, as thereby he will loose his code confidentiality (Contradiction).

According to the invention, the user will pre-code his commands and compile them to use them in future on pressing a specific key or mouse button (Principle-10: Prior Action).

The user-defined tasks will be displayed in an overlay window where the user can select the desired operation (Principle-17: Another dimension).

2.2 System and method for configuring a graphic interface (US Patent 5163130)

Background problem
In a graphical user interface, typically graphic elements are linked to single programming elements or objects. For example, a program may have buttons like “New File”, “Save File”, “Exit” etc., which are defined in the application. A drawing toolbox may have buttons for “line”, “box” etc. In this way, the buttons are linked to functions and not to variable fields. The program internally links the variables to the variable fields. This limits the ability of the user to configure an interface to achieve a desired result.

Solution provided by the invention
Patent 5163130 (Invented by Hullot, assigned by NeXT Computer Inc, Issued Nov 1992) provided a solution to the above problem. This graphic interface configuration system allows a user (or a programmer) to create a graphic interface for a computer program in which graphic elements in the interface are linked to variables or functions in any one of a number of programming elements.
TRIZ based analysis
The invention uses a GUI configuration utility to display the program elements of an application and also to display the variables and action portions (Principle-24: Intermediary).

The user selects the graphic elements and dynamically link to variables or functions as per his choice (Principle-15: Dynamize).

2.3 Method and apparatus for facilitating operator reconfiguration of a graphical user interface in a data processing system (US Patent 5384910)

Background problem
The conventional graphical user interfaces are rigid in structure and displays all the items in a menu simultaneously without leaving any role for the operator in designing the menu. This causes operator confusion and makes the software inconvenient and unpopular. There is a need to streamline the interface and eliminate the arbitrary display features in order to minimize operator confusion.

Solution provided by the invention
Patent 5384910 (invented by Torres, assigned by IBM, issued Jan 1995) uses a menu-formatted graphical user interface, which can be configured by the operator. The initial display configuration is set by the software manufacturer, which can later be modified in response to operator input.

The invention provides (1) adding a new menu field (2) removing a particular menu field and (3) automatically providing a visual representation of the particular menu fields. The end-user operator can rearrange the (1) textual content or (2) iconographic representation format or (3) both.

This menu-formatted graphical user interface is directly manipulated by the operator. The size of the GUI is automatically adjusted in response to the inclusion or removal of items made by the operator.
### TRIZ based analysis

The invention provides a menu formatted graphical user interface, which may be configured by the end user (or operator) for inclusion or exclusion of menu items (**Principle-24: Intermediary, Principle-15: Dynamize**).

The sizes of the visual components are automatically adjusted in response to the inclusion or deletion of items (**Principle-15: Dynamize, Principle-25: Self service**).

### 2.4 Method and apparatus for providing conditional cascading in a computer system graphical user interface (US Patent 5425140)

#### Background problem

In a cascading menu system, the user is presented with a main menu including a plurality of choices, each of the choices representing a basic function again having a plurality of choices. This advanced function of a menu interface is very desirable for more experienced GUI users but seen as burdening and confusing to the novice users.

#### Solution provided by the invention

Patent 5425140 (invented by Bloomfield, et al., assigned by International Business Machines Corporation, issued in Jun 1995) provided a means of shielding the users from advanced operations in a graphical user interface.

The invention includes an advanced function activator button adjacent to one of the user selectable action items. The advanced functions are to be displayed only if the advanced function activator option has been selected by the user.

This option enhances the ease of use of a graphical user interface by novice users, as the advanced functions are not displayed to them.
TRIZ based analysis
The GUI should display advanced functions for advanced users, but the display of advanced functions should not confuse the novice users (contradiction).

The invention isolates the advanced functions from the basic functions and does not display the advanced functions to a common user (Principle-2: Taking out).

The user can decide whether to view only basic functions or view the advanced functions by checking an advanced option check box (Principle-15: Dynamize).

The invention provides a method of dynamically classifying advanced items and basic items. For example, when the user selects a submenu item that is different from the default submenu item, the presently selected submenu item becomes the default item or behavior (Principle-15: Dynamize).

2.5 Menu bar Editor (US Patent 5530796)

Background problem
While developing the application, the developer must define and feed the menu items to the application. It is a tedious process to accomplish this by altering the properties of the menu object. There is a need to provide a simple user-friendly technique to design and define application program menu bars.

Solution provided by the invention
John Wang invented a technique (Patent 5530796, assigned by IBM, June 96) of direct editing of the menu bar items in the menu bar object through a GUI. While designing the application, the developer can drag and drop a menubar tool object on an application window on the same display screen. This forms a blank menu bar below the title bar of the application window. The developer enters the menubar items directly on the screen.
**TRIZ based analysis**

The invention provides the programmer a graphic user interface to directly build the menu bar on the application and edit the menu items inside the menu bar itself (*Principle-24: Intermediary*).

The method allows feeding the menu items in the same way as it would appear when displayed finally on the application screen (*Principle-26: Copy*).

**2.6 Method and system for adding buttons to a toolbar (US Patent 5644739)**

**Background problem**

The toolbars provide convenient alternatives to drop down menus. Some of the modern toolbars provide option to add and remove buttons from the toolbar, but through difficult and cumbersome methods. There is a need to provide an easy way of adding and removing buttons from a toolbar.

**Solution provided by the invention**

US Patent 5644739 (Invented by Elizabeth Moursund, Assigned by Microsoft Corporation, Jul 97) disclosed a method of intuitively adding a button or other type of control to a toolbar. According to the invention, dragging and dropping controls onto the toolbar can create new toolbar controls. The new control is bound to an operation of the object and can be executed by a mouse event.

**TRIZ based analysis**

The invention allows creating new buttons on the toolbar by dragging and dropping items on the toolbar (*Principle- Dynamize*).

The invention allows easy removal of buttons too. Thus the user can add and remove buttons as and when required (*Principle- Discard and Recover*).
2.7 Programmable user interface elements (US Patent 5898434)

Background problem
The Macintosh computer user interface employs a desktop metaphor that displays different icons and buttons on the desktop. Although a user can create and define a button by associating some script with the button, the user-defined button has limited functionality. For example, a user cannot create a "To be printed" button which will print the marked pages at a later period. The concept of macros in Microsoft Excel, although can perform certain calculation in the spreadsheet, has limited functionality.

Thus the icons, cells, stamps, buttons etc have limited functions and cannot be used to mark, find, organize or process data in documents within a computer system. There is a need to provide programmable user interface elements to meet the wide range of user requirement.

Solution provided by the invention
Small et al. discloses a user interface system (Patent 5898434, assigned by Apple Computers, Apr 99), which includes user interface elements for marking, finding, organizing and processing data within documents stored in the computer. The users can create their own elements, program the elements with their own desired functionality and modify the existing elements as per need. The elements can also be compounded together so as to cause a combination of tasks to be performed by simply activating one element.

TRIZ based analysis
The invention increases the programmability of the buttons so that the user can define the buttons to do more complicated functions (Principle-38: Enrich).
2.8 Grouping of computer operations (US Patent 5929851)

Background problem
The graphical user interfaces provide various graphical objects to access various functionalities of an application. It’s necessary to change the icons and tools in the interface according to the need of the user. Some applications provide mechanism for customizing toolbars and menu bars through drag and drop operations, but there are no mechanism for customizing other types of collective views such as grouping actions with icons and buttons.

Solution provided by the invention
Vanessa Donnelly invented a method of grouping actions (Patent 5929851, Assigned by IBM, Jul 99). According to the invention, the user can select an action in order to invoke a desired operation. The user is provided with an easy to use mechanism for grouping actions together as desired, so that his commonly used actions are provided in one place.

TRIZ based analysis
The invention provides customization of icons and buttons (Principle-15: Dynamize).

The user can group similar actions or buttons together, which will facilitate ease of finding and ease of execution (Principle-5: Merging).

2.9 Method and system for the dynamic customization of graphical user interface elements (US Patent 6069623)

Background problem
The conventional graphical user interfaces provide scroll bar to scroll a viewable object, menu options to perform a user selected action, a toolbar to access commonly used commands and so on. It is sometimes desirable to allow users to customize the way in which the graphical user interface operates, as the developers cannot realistically anticipate at the time of program development.
Solution provided by the invention
Patent 6069623 (invented by Brooks, assigned by IBM, issued May 2000) provides a method for customizing the graphical user interface by the end user. The invention provides two types of selection mechanism, viz. normal selection and special selection. The normal selection can be done by conventional mouse click and the special selection can be initiated by a special key combination or mouse input.

The normal selection is meant to initiate a normal course of action, such as selecting a menu item to perform the function of that menu item. The alternative selection or special selection is meant to customize a menu item; such as to modify the caption, picture, location or visibility associated with a selected menu item or other graphical user interface elements.

TRIZ based analysis
The invention allows customizing graphical user interface elements displayed within a graphic interface by the end user (Principle-15: Dynamize).

The invention allows two types of selecting menus or buttons, one for normal course of action and the other for customizing the user interface elements (Principle-35: Parameter change).

2.10 System and method for customizing appearance and behavior of graphical user interfaces (US Patent 6104391)

Background problem
With the advantage of multitasking every user wants to run multiple applications each having their windows opened on the desktop. But the GUI of each different application has its own appearance and behavior. There may be dissimilarity in appearance and behavior between applications, which can be annoying and confusing to a user.

It would be desirable for the application developers and application users to have additional flexibility to have greater control over the appearance and behavior of the desktop objects.

Solution provided by the invention
Patent 6104391 (invented by Jr. Johnston et al., Assigned by Apple Computer Inc, issued Aug 2000) provides an appearance management layer that gives users (both developers and end users) the ability to customize the appearance and behavior of the desktop.
The invention provides a layer between all the applications and the graphic subsystem, which actually writes to the display. In this way, a level of abstraction is provided between the client and the system so that the user switches between themes to customize the visual appearance and behavior even at runtime.

**TRIZ based analysis**

The user can customize an application(s) (Principle-15: Dynamize) to bring consistency of look and feel between different GUI elements of the application or between elements of different applications (Principle-6: Universality).

Sets of objects can be grouped into themes to provide a user with a distinct overall impression of the interface (Principle-5: Merging).

The invention provides a layer between the applications and the graphic subsystems. The customization is done at this layer (Principle-24: Intermediary).

### 2.11 System and method for displaying data using graphical user interface control elements (US Patent 6237004)

**Background problem**

The graphic interface usually includes graphic symbols and icons to represent objects or elements in the system. The graphic symbols or widgets are associated with program codes to process certain desired behavior. This means, in order to change the type of widget used, for example, from a checkbox to a button, the application code must be changed, recompiled or rebuilt. It would be desirable to change and modify the widgets without recompiling the program.

**Solution provided by the invention**

Dodson et al. invented a method (Patent 6237004, assigned by IBM, may 2001) of dynamically changing the GUI control element including the graphic symbol and control code without changing or recompiling the source code. The invention uses a data driven model wherein the GUI control element is determined...
dynamically, based on information contained in a database. The application calls a generic GUI manager, which determines which widget to use for the particular data available at that point. This can change the graphic symbols and data items for the GUI.

The user can modify the widget library through the GUI manager to make changes either in the graphic symbols or in their behaviors or both.

TRIZ based analysis
The widget specifications are kept outside the program code. Hence changing of widget specifications doesn’t require changing of program code (Principle-2: Taking out).

The graphic symbols and control codes are managed by the Generic GUI Manager. The executable program has to call the Generic GUI manager to display widget data (Principle-24: Intermediary).

The user can change the behavior of graphic symbols by making changes in the widget library (Principle-15: Dynamize).

2.12 Method, apparatus, system and computer program product for a user-configurable graphical user interface (US Patent 6275225)

Background problem
The GUI becomes more and more complex in complex applications. Currently, some applications provide certain features of customizing the GUI such as linking icons to functions. But the options are very limited. It would be nice to allow the user to tailor his GUI depending on his requirements.
Solution provided by the invention
Rangarajan et al. disclosed a method of customizing a GUI to meet the user requirements (Patent 6275225, assigned by Sun Microsystems, Aug 2001). According to the invention, the GUI customization tool allows the user to customize the GUI according to his need, which can be saved in different configuration files. The stored configuration files are used to retrieve specific GUI configurations for future use.

TRIZ based analysis
The invention allows the user to customize his GUI (Principle- Dynamize).

The user can save the customized configurations in configuration files and use the saved configurations when necessary (Principle-10: Prior Action).

3. Summary

It is necessary to customize a GUI, as the developer cannot anticipate the exact requirements of a future user at the time of development. As a user is the final decider of his requirement, it is better that the user himself customizes the GUI the way as he needs (Using TRIZ resources, Principle-25: Self service).

However, there are different problems in implementing user configuration. For example, the user may not have access to the source code to modify the program; the user may not be technically competent to incorporate the desired changes; the GUI configuration may require abundant time and effort; the configuration done by one user may not meet the expectations of another user and so on. The article describes and analyses various inventions, which tackle the problems relating to user configurability in different ways. Some interesting solutions are as below.
The user-defined tasks will be displayed in an overlay window.

The user can select the graphic elements and link them to variables through a GUI configuration utility.

The user may be allowed to include or exclude options from a pre-built set of GUI elements.

The functions may be classified into basic and advanced. The user may be allowed to view or hide the advanced functions.

The user (or programmer) may edit menu items in the menu bar itself.

The user may create or remove buttons by dragging and dropping items on the toolbar or similar designated area.

The user can program the buttons for complicated functions.

The user may be allowed to group similar actions or buttons together.

The widget specifications (such as buttons, checkboxes etc.) are kept outside the program code, so that the user can change the widget specifications without needing to change the program.

The user can save different configurations in separate configuration files and use the saved configurations as per requirement.

User configurability, although one of the possibilities of GUI configuration, is not the best solution as per an IFR analysis. Ideally the GUI should be configured automatically according to the need of the user without the user bothering how to do it (Ideal Final Result). This leads to the concept of automatic GUI configuration (Principle-25: Self service), which we will discuss in the next article.

Reference to patents:


**Other references:**


