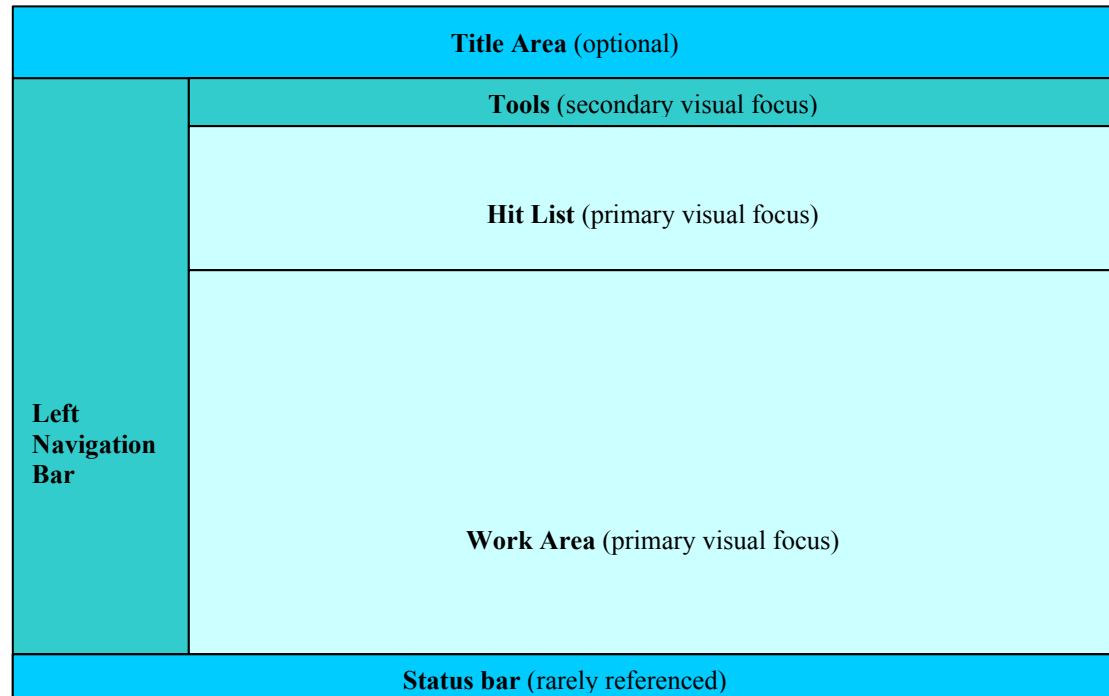


jACOB GUI Guidelines

GENERAL.....	4
CAPITALIZATION.....	5
TITLE-CAPITALIZATION.....	5
SENTENCE-STYLE.....	5
SYSTEM MESSAGES.....	6
GENERAL RULES.....	6
WORDING.....	6
EXAMPLES.....	6
LAYOUT	7
GENERAL LAYOUT STRATEGY.....	7
SPACING BETWEEN GROUPED CONTROLS.....	7
SPACING BETWEEN SINGLE CONTROLS.....	8
VISIBLE CONTROLS.....	9
BUTTON.....	9
COMMON.....	9
RULES.....	9
CHART.....	9
CHECKBOX.....	9
DATE.....	9
DROPDOWN LIST BOX.....	9
COMMON.....	9
RULES.....	9
INPUT FIELD.....	10
CHECK BOXES.....	10
COMMON.....	10
RULES.....	10
LABEL.....	10
COMMON.....	10
RULES.....	10
TABLE VIEW.....	10
COMMON.....	10
TEXT EDIT.....	10
FILE UPLOAD.....	10
DIALOGS.....	11
TREE DIALOG.....	11
MESSAGE DIALOG.....	11
EXAMPLE.....	11

General

The user interface for the **jACOB** Web Client consists of six parts: title area, left navigation bar, tools, hit list, work area, and status bar.



The visual emphasis on the different areas is determined by how often call center agents need them; the focus is on those areas where agents spend most of their time.

The **hit list** and **work area** are the primary visual focus. The **work area** is where agents carry out their main tasks. The left **navigation bar** and the **work area** tools are the secondary visual focus. The navigation bar allows agents to call up the tasks they need, and the work area tools provide the functions for those tasks.

The **status bars** and **title area** are rarely referenced. The title area could contain the name of the application, while the status bar generally displays the application status.

Capitalization

There are two systems of capitalization used in program strings. When to use which system is included in the rules for different situations.

Title-Capitalization

Title-style capitalization is, unfortunately, quite complicated. Just do your best to follow these simplified rules. The rest will be cleaned up in the proofreading process.

- Capitalize the first word no matter what is it
- Capitalize all major words
- Lowercase *the, an, a, and, but, for, or, nor, to, and as*
- Lowercase prepositions, such as *up, in, down, and of*
- Capitalize the last word

The following are some examples strings in title-style capitalization.

- Hostname and Name Server Configuration
- Do Not Use LDAP
- Reload All Patches from Server

The specific rule followed in the proofreading process is the Chicago Manual of Style rule 8.167.

Sentence-Style

Capitalize the first word. Only capitalize other words if they are proper nouns or names.

System Messages

General Rules

Use sentence-style capitalization for system messages.

Wording

1. Don't use technical jargon.
2. Don't use *Please*.
3. Don't use the word '*Error*'.
4. Speak the user language.
5. Use positive construction.
6. Use imperative.
7. Always use a verb in a sentence with a dot at the end of the line of the sentence.
8. No special characters like <>'-'*# in the sentence.
9. Do not use abbreviations like *e.g.*, *etc.*, *i.e.*
10. No article at the beginning of the sentence.
11. Do not shout at the user.

The term '*Error*' message is relevant only to programmer, it has no equivalent in our day-to-day conversation with other peoples. '*Error*' has a strong negative connotation. It adds nothing between the computer and the user. And can only decrease the spirit of cooperation between the user and computer.

Your application is designed to cooperate with the user, helping him or here to complete some task. Using exclamations points (!) in your messages, either in the message, in the title or as icon transforms your program from a cooperative partner to pompous, arrogant ass. Avoid them.

Examples

Bad: Personal number has not been entered

Good: Enter personal number.

Bad: Please enter the company code

Good: Enter the company code.

Bad: The data can't be saved.

Good: Data cannot be saved.

Bad: Error: You have entered an invalid date!

Good: Enter date in format DD.MM.YYYY

Bad: Would you exiting without saving?

Good: Would you save the data before exiting?

Layout

General Layout Strategy

This describes a general strategy for laying out jACOB applications and forms. Laying out a form is not just "throwing" controls on a page. Several aspects have to be considered, such as


- Flow of control - how the user progresses through a form when doing his or her work
- Dependencies - how elements on a form affect each other
- Togetherness - which elements on a form belong to each other, there may be closer and farther relations between elements
- Aesthetics and general Gestalt principles - how information can be effectively communicated visually

There are three steps in laying out - these can be done in the following sequence:
Determine the

1. Sequence of elements (vertical, horizontal)
2. Nesting of elements
3. Spacing between elements at different hierarchy levels.

The *sequence* takes care of the flow of control, dependencies, and information about which elements belong together - the latter in a more linear fashion. The *nesting* also takes care of dependencies and of togetherness -- but in a hierarchical or top-down fashion. The *spacing* takes care for aesthetics and the proper application of Gestalt principles (mostly togetherness).

Spacing between Grouped Controls

	0	1	2	3	4	5
	Group					
-	Label	personFirstname <TEXT>		Search 	Clear	
	Label	personName <TEXT>		New	Update	
1	Label	personNickname <TEXT>				
	Label	personStreet <TEXT>				
-	Label	personAny_longtext <LONGTEXT>				
2						
1						
3						

Spacing between Single Controls

Visible Controls

Button

Common

Use title-style capitalization for buttons.

Rules

1. Whenever a function can lead to lost data, provide the user with a warning.
2. Use an infinitive verb, such as *Cancel*, *Close*, *Delete*, or *Save* when it is clear what object is affected by the action. When it may not be clear, use the verb with the name of the object, for example, *Add to Shopping Cart* not *To Cart*.
3. Be as precise as possible. For example, write *Change Address* not *Process Address*.
4. Use ... at the end of command that open a pop-up or new dialog requiring user input.
5. Do not use ... on navigational buttons, such as *Next* or *Back*.
6. When the group of buttons allows the user to do different things with the same object, use the verb alone on each pushbutton. For example, if he is working with an invoice, write *Change* on one button, and *Display* and *Cancel* on the others instead of *Change Invoice*, *Display Invoice* and *Cancel Invoice*.
7. When the group of buttons allows the user to do the same thing with different objects, use the object alone. For example, if he or she is making changes to various objects, write *Invoice*, *Cancellation* and *Address* instead of *Change Invoice*, *Change Cancellation* and *Change Address*.

Chart

Checkbox

Date

Dropdown List Box

Common

Use title-style capitalization for the labels.

Rules

1. Always propose an appropriate default value in the list box field.
2. If there is no suitable default value, display *No object selected*. Do not leave the field empty, and do not use it for instructions.
3. When using a list box for navigation, set the system to act on the users choice when he or she presses **Go**. Don't use a UI hook for automatic navigation on selection change.
4. Dropdown list boxes are most appropriate for up to 20 alternatives.

Input Field

Check boxes

Common

Use title-style capitalization for labels.

Rules

1. Use checkboxes when the user needs to be able to select multiple items simultaneously, or select only one item, or select nothing.
2. Checkboxes are most appropriate for up to 7 - 9 items.
3. A horizontal arrangement is good for 2 to 4 checkboxes,
4. Vertical arrangement or a matrix is good for more.
5. Place checkboxes (or a single checkbox) that refer to one or more fields below, and left-aligned with, the field or fields. When there is enough space, you can place a single checkbox to the right of the field or fields. If it refers to multiple fields, place it to the right of the last field.
6. Place checkbox labels to the right of the checkboxes.
7. When other screen elements are dependent on the setting of a checkbox, place them below the checkbox, and use an unmarked checkbox for the default case.

Label

Common

Use title-style capitalization for labels.

Rules

1. Ending punctuation, such as : or ., should not be used.
2. Use words or short phrases that users easily understand.
3. Use nouns or phrases with nouns.
4. Use title case, for example *Shipping Address* not *Shipping address*.
5. Do not use punctuation at the end of the word or phrase.
6. Place a label to the left of the field it describes. If this is not possible, place it above, and left-aligned with, the field.
7. Remember that translations may require more space than the original language, and provide some extra spaces.
8. Consider providing a link to the glossary if it would help understand the field label, and if it is possible.

Table View

Common

Use title-style capitalization for column and row headers. Use sentence-style capitalization for entries. Do not use ending punctuation.

Text Edit

File Upload

Dialogs

Tree Dialog

The programmer pointed out that a tree soon fills up with identical icons differentiated only by the names that appears beneath. He suggest that a greater variety of icons be used because the “environment is a visual one”. He’s right in that tree full of identical icons is just a waste of space. He goes on to suggest that four different icons be used. But there will still be lots of identical icons lying around. He doesn’t take the next step and realize that the icons are totally unnecessary. In making graphics-based interfaces we must remember that text is also a visual cue, it is a very powerful one that is full of specific content, and one that we all know very well. Text is often the best visual cue possible; we are expert at visual differentiating one word from another (especially if they are not printed in all upper case letters) and can convey quite specific content.

TODO: Image of good and bad TreeDialog

Message Dialog

No messages are good messages. Dialog boxes need extra effort of the user. Both physically and psychologically. Reconceive the need of each message box.

Example

If you have a purchase order open and you want to do an item inquiry you receive the following dialog box: ”This function may not be used when Create/Update P.O. is running”. The correct user reaction is: Why not?

It is clear that the designer were not aware of real work flow of the user’s task. The general principal here is that almost any overly-structured work flow approach to system design risks impeding a user whose task or inspiration demands a different approach. In this case the interface moves from aiding the user to being a dictator. A computer should be servant, not a peer or boss.