

Carlos Marin Burgos

HCDE 510

03/11/2011

# Variations Analysis Paper

---

## Description of the Base Document

For my variations project, I decided to use as a base document the source code of a computer program written in Java programming language. One of the reasons behind selecting a document written in Java consists of the understanding of the structure and content of programming programs due to my work experience. Also, I think that it would be interesting and challenging to transform the original document into different formats such as an *UML diagram*, an *IMAP document*, a *tweet* and a *hyperlink document*.

The base document includes a series of classes that divided the document in different sections. Each class is designed with a purpose in the program. For example, the class named `MainClass` serves a motor to run the application. In a way, the classes serve the function of headers in a document. Each class describes its content. As result, the reader can scan trough the document focusing in the classes to find any particularly functionality in the program with reading the entire document. In addition, methods and procedures support the classes with different behaviors such as drawing objects or getting dimension of the objects. The information includes in these methods and procedure can be defined as the content of the document.

Another characteristic of the document consist of its linear structure. Even though the classes can be read independently one from each other, the hieratical model that computer programs follow make necessary to read the document linear to be able to understand the flow of the content. The writing style found in the base document might differ for the classic SE model documents. It is obviously, due to is technical content, that this document is orientated to readers with a programming background. As a result, the writing style of the document consists of programming language command written in Java which makes this document difficult to non programmers to comprehend. Finally, the purpose of the base document is to serve as code reference to others programmers.

## First Variation: UML Diagram

For my first variation I decided to use a Unified Modeling Language (UML) diagram. This type of diagram used in computer science helps to visualize and document all the artifacts in an object-oriented program. As figure 1 shows, UML diagram consists of a series of cubical figures connected by arrows that allows the reader to visualize the structure of the computer program. Each square shown in the diagram represents a class from the base document. For instance, the base document includes a class named `MainClass` where a series of commands are declared inside. Also, another characteristic of UML consists of displaying the relationships

Shapes Diagram

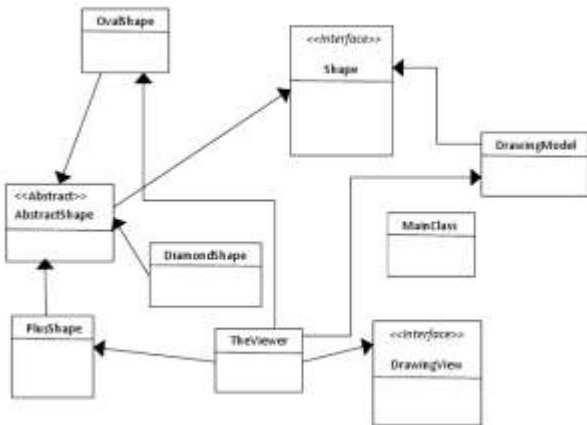


Figure 1. UML Diagram from the Shapes program

among the classes. The use of arrows indicates where the relationship exists between the different classes.

Presenting a program written in Java language on a document might present difficulties when trying to analyze the different relationships between the classes. One of the strengths of UML diagrams consists of the use of graphics to represent the relationships among the different classes in the program. According to the article “What Kinds of Writing Have a Future?” by Robert Horn, diagrams are non-linear like the structure of arguments and the necessity of boxes and arrows to show the

structure of the arguments on an argument map is necessary <sup>1</sup>. Therefore, displaying the relationships between the classes in a program using a written document can be very complex. As a result, UML diagrams use boxes and arrows to represent complex relationship between classes in the same way that arguments diagrams help to symbolize complicated claims on an argument.

On the other hand, one of the weaknesses in UML diagrams consist of the technological levels that the reader needs to have in order to understand what is presented in the diagram. Displaying a UML diagram to a reader without experience in programming might result in frustration due to the lack of knowledge on the subject. Also, the

In conclusion, UML diagrams help programmers to have an overlook of the artifacts that compose a program without the necessity of reading the source code written in Java. Also the base document does not lose any content when transforming into a UML diagram.

## Second Variation: IMAP Document

My second variation consists of creating an IMAP version of the base document.

Shapes Program											
Class:	MainClass.java										
Description:	Runs the program and creates a graphics view and put it in a window.										
Contains:	public class MainClass{}										
Class:	AbstractShape.java										
Description:	This class contain only instance variables that draws the object, all the "gets" and all the "sets"										
Contains:	Only instance variables that draws the object, all the "gets" and all the "sets"										
Class:	DiamondShape.java										
Description:	Constructs a new Diamond Shape with the given x, y coordinates, and size.										
Contains:	<table border="1"> <tr> <td>public Object clone()</td> <td>overwrites clone.</td> </tr> <tr> <td>public void drawShape(Graphics g)</td> <td>Draws the shape of the object</td> </tr> <tr> <td>public Color getColor()</td> <td>Gets the color of the object</td> </tr> <tr> <td>public int getHeight()</td> <td>Gets the height</td> </tr> <tr> <td>public int getWidth()</td> <td>Gets the width</td> </tr> </table>	public Object clone()	overwrites clone.	public void drawShape(Graphics g)	Draws the shape of the object	public Color getColor()	Gets the color of the object	public int getHeight()	Gets the height	public int getWidth()	Gets the width
public Object clone()	overwrites clone.										
public void drawShape(Graphics g)	Draws the shape of the object										
public Color getColor()	Gets the color of the object										
public int getHeight()	Gets the height										
public int getWidth()	Gets the width										

Figure 2. IMAP created from the base document.

According to Horn, some characteristics of structure documents such as IMAP documents included displaying only one chunk of functional information per information “block”, keep the document short, and label the document for easy scanning <sup>1</sup>. These characteristics can be applied to my base document without losing any content from the document in the process. It also serves as a structure document where the information presented in the base document can be easily found. The Shapes Program IMAP divides the base document into nine different key blocks.

These key blocks contain the name of each class, the description and the procedures and the

methods contained in the class.

One of the strengths of creating an IMAP variation of the base document consists of the facility for the reader to scan through the entire IMAP document when searching for any functionality in the program. Due to the block structure of IMAP documents, the information displayed in the text can be easily scanned. Another reason why this variation can be well applied to the base document consists of the use of headers. For example, if the base document contains several programs rather than just one, the difficulty of search content among the document would frustrate the reader. However, due to the use of headers in IMAP documents, the reader can search easily through the document using the headings as a search tool to find any content in the text.

On the other hand, one weakness found in this variation consists of losing the originally source code from the program and leaving only instead the name, description and content of the classes and procedures. Even though the structure of the base document stays when applying the variation, some content is missed during the process. As a result, the text transformed into an IMAP document serves as a reference to the classes and procedures rather than code documentation for the programmer.

### Third Variation: Twitter

The target of my third variation consists of being able to tweet the content of my base document to any twitter follower. Twitter consists of a social networking tool where users can send “tweets” as a unit of information. Every “tweet” must contain no more than one hundred and forty characters. As a result of this restriction, I face the challenge of putting the content of an eight-page document into a one hundred and forty-character display unit. The result consists of a chunk of one hundred and forty characters of non-sense Java code.



Figure 3. Twitter screenshot

The obvious restriction of the number of characters of this display unit makes this variation a failure when trying to deliver the content as per se of an eight-page document. However, I think that using a “tweet” with the summary of what the base document contains might be a good option. The same way that a structure document uses the header to describe the content of the chunk of information that follows. For instance, a “tweet” based on the document can say “MainClass.java runs the program and creates a graphics view and put it in a window.” The unit of information sent describes the content of the MainClass class using only eighty-two characters. Again, as in the IMAP document case, the source code is omitted in this variation but the description of its content can be delivered.

However, one possible way to deliver the entire content of the base document through Twitter consists of sending a description of the document next to a link where the entire document can be accessed. For example, the “tweet” can read “The shapes program at [www.shapesprogram.com](http://www.shapesprogram.com).” As a result, the entire content of the base document can be accessed through an intermediary tool that works as a bridge between two variants, Twitter and a SE document. I think that one of the strengths of Twitter as a variation consists of the power of diffusion to other recipients using a hyperlink as a medium.

### Fourth Variation: [Hypertext Document](#)

The last variation consists of creating a hypertext document from the base document that consists of a hierarchy structure that allows the reader to get an overview of the information path designed in the Shapes Program site. For instance, when the user clicks on the cross-reference

```

Shapes Program
Class: MainClass.java
Description: Runs the program and creates a graphics view and put it in a window

Class: AbstractShape.java
Description: This class contains only instance variables that draws the object, all the "gets" and all the "sets"
Description: This class contains only instance variables that draws the object, all the "gets" and all the "sets"
Contains: Only instance variables that draws the object, all the "gets" and all the "sets"

Class: DiamondShape.java
Description: Constructs a new Diamond Shape with the given x, y coordinates, and size.
Contains:
public Object clone()
public int getWidth() Gets the width
public int getX() Gets x coordinate
public int getY() Gets y coordinate
public void setColor(Color c) Sets the color
public void setHeight(int height) Changes the height and width separately have no effect in this case size is width and height.
public void setWidth(int width) Sets width of the object
public void setX(int x) Sets width of the object
public void setY(int y) Sets width of the objectSets Y coordinates of the object

```

Figure 4. [Hyperlink version of the Shapes program](#)

named `MainClass.java`, a new display unit appears showing all the content of that unit of data. In addition, due to the hierarchy structure of the base document, classes depend and heritages each other, the hypertext document represents, as David K. Farkas states in his article “The Linear-Hierarchical Model”, navigational choices instead of

structural markers forced on text that asks for linear reading <sup>2</sup>. The choices include classes and procedures for each class that forces the reading on the document to be linear.

One of the strengths of using a hypertext document as a variation consists of the enhancement of the navigation format from the original base document. For example, when using cross-references in the hypertext document allowing navigate across the multiple classes in a “linear” mode where the document starts and finishes in the same node following a linear pattern of navigation through all the classes. Consequently, the reader gets an overview of the information path rather than a document where the lack of navigational options makes the document difficult to follow when reading.

Another strength found when converting the base document into a hypertext document consists of providing a better organization to the document which makes the new version more searchable than the original. The original document presents a structure that consists of a series of classes without a hierarchical order that makes difficult to the reader to observe where the relationship between the classes takes place. For example, the class `OvalShape` extends `AbstractShape` class and implements the `Cloneable` object. If the user pretends to follow the

order previously describe starting in the OvaShape and following the AbstractShape and see how is implemented by the Cloneable needs to search every page due to the lack of organization of the document. The classes do not follow and order resulting in a search through the entire base document in order to follow the relationship between the classes. However, transforming the base document into a hypertext document facilitates the specific search of any relationship among the classes.

On the other hand, a weakness rises when the hypertext document increases its size and the relationships between the classes. For example, when clicking on the public Object [clone\(\)](#), the reader has the option of keep navigating in a succeeding series of display units reaching a point where the user can lose track of the current location. One option to solve this problem consists of creating a breadcrumb navigation system that allows the user to keep track of the current position. However, when the document increases its size and consequently the relationships among the classes from eight to two-thousand pages, the linear navigation system might evolve into a web-like structure. As a result, it can very hard to navigate through its content due to the lack of order of each node.

## Work Cited

1. Horn, E Robert. "What Kinds of Writing Have a Future?" October 22, 2001
2. Farkas, K David. "The Linear-Hierarchical Model." unpublished course material, 2010

# Variations Reflection

---

Before the course started I did not have a clear concept about what exactly is information design. Then, as the class was advancing I started to comprehend how important is to understand how to select the best display units to deliver efficient information to others in order to understand the message you are trying to distribute. This is information design.

For the variation project I tried to select a challenging base document because I did really want to test myself in the information design concept and see what I have learned during the quarter. These are some things that I have learned during the process of competition of the variation project.

First of all, one thing that I have learned in this project is to work alone in a project. This is my second quarter in the HCDE program and so far all the assignments have being organized to work in groups. However, for the variation project, the requirements states that the project consist of an individual assignment. As a result, it has being very challenging to work alone primarily because you have to administrate the time and task in order to success. One question that I would like to ask, is how would be possible to work in groups in an assignment like this? It could be interesting to see how the variations are selected as a group rather than individual.

Second, I have learned how wrong an information unit turns out when the format selected for delivering information is not appropriate. For example, one of my variations consisted of using Twitter as a format to deliver the content of my base document. The result was not very successful due to the limited of characters in the display unit. However, I did look for variants that could work while using Twitter as a bad example and I found some ways to deliver the content even when the format is not the correct one.

As a result, I did learn during the process of the Twitter variation how to modify or look for a solution when the format is not the appropriate to deliver good content. Consequently, I think that I have gained experience for future work assignments when a mandatory format is chosen by superiors and you are the responsible of delivering the information with the wrong tool.

Finally, one of the more important techniques learned during the execution of the variations project consist of the use of the Information Design concept as a tool. For example, concepts such as genre, medium and format, which are part of the information design concept, can be use as justification to prove what the best format to deliver content is.