International Journal of Information & Computation Technology.
ISSN 0974-2239 Volume 4, Number 9 (2014), pp. 931-936

http://www. irphouse.com

# Impact of Event Driven Programing Paradigm on Real World

**HarshitJuneja, Himanshu Yadav, Rajat Paruthi, Vikram Gupta**

*Student[1], Student[2], Student[3], Student[4]*
*ITM University, Gurgaon*

**Abstract**

The real world works according to events. People make unique choices according to the conditions they are put in. In this pap er, an event driven approach is talked about and the impacts it has had on various fields with its evolution. Three of which have been discussed in the paper. At the lower level, basic sequences are coded in elementary software objects called function blocks which prove their functionality when called or used. At the upper level, the execution of such blocks is carried out according to the desired sequences forced by a controller. The interaction of the user with the computer used to be limited to the sequential file, this all changed with the arrival of event driven programs the user can now engage the computer while it is running ot her tasks.
**Keywords –** event driven approach; function blocks; interaction etc.

## 1. Introduction

Event driven programming, as the name suggests is a programming paradigm in which the primary activity is reacting to the signals. These signals or rather inputs can be received from any source. They can be timers, human responses such as a basic left mouse click or can be made during the computational process in response to other signals.

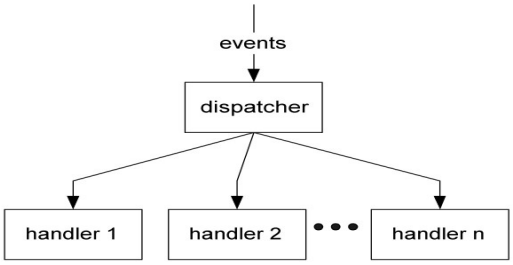
Modern graphical user interfaces like the desktop of our computer have been possible because of event driven paradigm, earlier it would just have been a few lines of code of instructions followed by the computer in a linear (procedural) fashion , thus leaving no scope for user interaction but with the advent of event driven programs there are various events (programs) for the user to choose from, giving the user a freedom to do what he/she wants and that alone. These programs have allowed us to truly push the

boundaries of programming by making it more interactive and responsive, the advantage of event driven paradigm is that it can be integrated into the existing paradigms to make a program much better and accessible.

Although it may appear to be similar to concurrency control, event driven programming is simply a tool that allows t he flow of control of program to change according to events that are external to the program itself.

## 2. Background and Related Work

The concept behind the development of Event Driven programming arose from the need to interact with the program while it is being executed which was not possible when batch processing was employed which essentially made a computer an assembly line where the sequential file is brought in and processed and finally the finished result or an error is generated. It started out in 1970's that Larry L. Constantine, a business application programmer developed structured systems. Today he is considered the human father of structured systems and IBM's Systems Research Institute the corporate father. Structured system uses dataflow diagrams (DFD's) to show the logical structure of a computer system. A dataflow diagram shows the logical functions that a system must perform, but it doesn't say anything about the design of the program that will perform those functions.




In order to write an event driven program the first thing we need to do is to write a series of sub-routines or methods called event handler routines. For e.g. - a single touch on the screen of a modern mobile phone triggers a routine that may open an app., save data to the memory space or delete something. Many modern programming environments provide the programmer with event templates that he/she needs to program in the events.

The second step is to bind the correct functions to the event handlers so that the correct/intended e vent takes place. Graphical editors combine the first two steps: double click the button you wish to program and the editor creates an empty event handler and opens a text window to let you type in the program.

The third step is to write a main loop which is basically a function that repeatedly checks if any event has been selected by the user and call the correct event handler to process that event. The continuous checking makes sure that no event is missed.

An event driven application is a computer program whose purpose is to respond accordingly to the actions generated by the system, or the user. In a computing context, we can identify any event that signifies its occurrence to the system hardware, or the software. Both system-generated events like loading of any program etc. and user-generated actions like mouse clicks etc. are included in events.

In Event-driven programming, the event-processing logic gets separated from the rest of the program's code. Batch processing stands in opposition of the event-driven approach. As event driven programming is a programing paradigm, therefore we can create event-driven apps in any language we want. The responsiveness, flexibility and the throughput can be improved with the help of event-driven processing depending upon the specific application.

*Peter Niblett* and *Dr. Opher Etzion* in their book *Event-Driven Processing in Action*, have described some purposes of event- driven applications which includes:

- All the event are around the application. The application will analyse and reacts to the events to which the sensors detects and reports.
- According to the situation (i.e. either good or bad), the application needs to identify and react to this. An event driven approach lets the application to respond in more timely fashion and in a more controlled way as the changes are more efficiently monitored as they happen whereas in a batch process, the monitoring process is done intermittently.
- The application might have large amount of data that might be needed as output for some application or to some human user. An event-driven approach can be used to distribute the analysis of taking the input data as events across multiple computing nodes.

With event-driven approach, an existing application can be extended in a pliable, non-invasive manner. To add some extra function, the original application can be instrumented with the help of adding some event procedures to it in spite of changing the original application and this functionality can be implemented by processing the events generated.

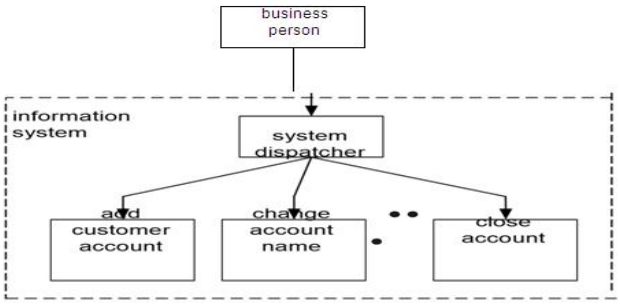
## 3. Successful Implementation of Event Driven Methodology

### *3.1 Industrial Automation Systems Based On PLCS*

Automation industry of today has the capability to implement applications involving widely distributed devices and a high reuse of software components thus generating a huge good quality production. A proper organisation of the inputs and outputs of function blocks and a supervisory control is very important for assembly lines in the automation industry to meet new challenges in this field. As a result, the application of event driven programing paradigm is of much more importance in industrial automation than it is in software engineering as industrial control system needs to be equipped with real time operating systems that guarantees real time constraints.

### *3.2 Business Integration Using Event Driven Programming*

The basic requirement of a business integration can also be fulfilled by an event driven systems which include event identification, event co-ordination, event driven updating of stateful objects and also event emission. In a business integration environment which basically includes business process automation, activity monitoring, marketing and even service mediation, transmission of events takes place typically via network channels.

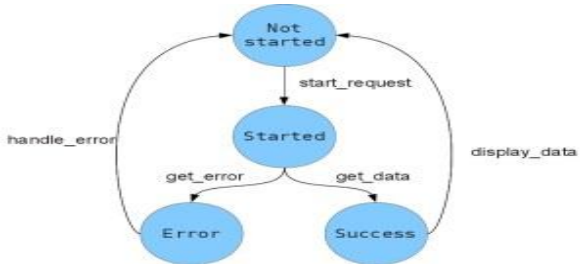



### *3.3 Power Management Using Event Driven Programming*

With the advancement in technology and graphical and computational processing of mobile consumer electronics such as smartphones and tablet PC's replacing conventional PC'S and other handheld devices. However the higher clock frequencies and ever increasing number of cores has increased the workload on the already strained battery life. Event driven programming along with processor hotplugging can be used to program a power management scheme to guarantee high responsiveness while at the same time optimising battery life. Since typical mobile systems place the framework layer on the OS to serve applications in an event driven manner. In this scheme we use an interactive event triggered b y the user for example a screen touch to be used to adequate level of processor performance providing the highest clock frequency in response to an interactive event while minimising battery loss by lowering clock frequency when the device is in an idle state. This system assisted event driven power management by the framework enhances the user's experience and energy efficiency of mobile consumer devices without compromising on performance.

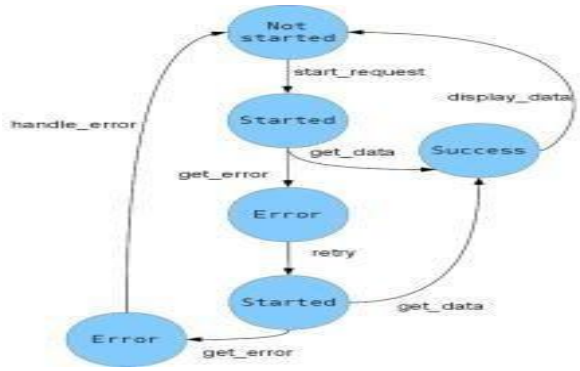
## 4. Challenges

Suppose we need to retrieve some data using a GET request, and handling a HTTP error. A simple state machine in generic event-driven framework will be implemented and the graph will be as shown:



Here, all states (excluding initial one) will correspond to a call-back. The framework will determine the transitions. The current state also need to be kept in track accurately in order to avoid starting more than one request at same time.

Now if we make a simple change to the program. If we need to make the requests try ag ain but only once. This time, some extra nodes are to be added for non-fatal error condition as more complications will be there this time in state machine.




In the event-driven code, if we encounter with an error, we have to keep track of that. Also, to perform right action, and keep it up to date, we need to check this flag at each call-back. Since, for multiple codes, same call-back is reused, so the code will not be shaped like the state machine. For testing, every possible path needs to be traced.

Suppose if simultaneous requests are to be allowed. This time, the code gets pretty fast. We here, also see many devastating changes in state graph by putting some small changes in the requirements. Practically, it is impossible to modify the code, a s many times, the state graph is kept implicit making it impossible to test the code reliably.

## 5. Conclusions

This paper covers all the important aspects of event driven programming paradigm and its implementation which proves its closeness to the real world. Thus making it a lucrative concept to integrate with the existing management schemes.

## References

[1] http://ieeexplore.ieee.org/xpl/articleDetails.jsp?tp=&arnumber=6490268&queryText%3DAn+Event-Driven+Power+Management+Scheme

[2] http://ieeexplore.ieee.org/xpl/articleDetails.jsp?tp=&arnumber=6686246&queryText%3DOn+Event-Driven+Business+Integration

[3] http://ieeexplore.ieee.org/xpl/articleDetails.jsp?tp=&arnumber=6571621&queryText%3DConsidering+Context+Events+in+Event-Based