Event Driven Design

Making applications more robust with an EDD approach

11/08/2023

Williams Uchena Olufemi Teddy-Jimoh

Event Driven Design

Abstract

This documentation proposes the concept of Event Driven Design (EDD) with reference to the Object-Oriented Programming model paradigm but not limited exclusively to. The messaging system presented by an event-driven architecture (EDA) could arguably be implemented in hardware-functionalities for increased robustness of a system, whilst the EDD may be implemented on a macro-scale throughout a software program through encompassing principles applied via syntactical implementations.

Main Findings

The overall concept of EDD is to integrate a system of transmitted messages via events. These events are triggered upon some event publisher that raises the event, and an event sink that consumes the event [1]. This is the conventional basics of event-driven architecture. With EDD, I propose that a system can be built using an event-driven paradigm that involves actions being raised that fire discrete functionality. Events are encompassed by two main nodes: Event Publishers and Subscribers [1]. Event Publishers raise events, and Event Subscribers consume events. Without this coupling, no action should be taken. What makes EDD a novel approach is the concept of *grouping* events.

Using the .NET C# programming language as a reference from here on, I shall be using C# concepts to demonstrate the EDD model.

Please consider the following scenario:

[Event Publisher, p1] ---------->> [Event Subscriber, t1] ------->> [Updated Action, n1]

In this simple case, an event publisher is set to push a notification to a listening event subscriber, which then updates some action. This workflow is fairly coupled. Now, consider the following example:

[Event Publisher, p1] ---------->> [Event Subscriber, t1] ------->> [Updated Action, n1]

[Event Publisher, p2] ---------->> [Event Subscriber, t2] ------->> [Updated Action, n2]

[Event Publisher, p3] ---------->> [Event Subscriber, t3] ------->> [Updated Action, n3]

[Event Publisher, p4] ---------->> [Event Subscriber, t4] ------->> [Updated Action, n4]

Multiple event publishers are coupled respectively to event subscribers, with separate update-calls being fired. Next, please consider the following example:

[Event Publisher Collection < p1, p2, p3>] ----->> [Event Subscriber, t1] ----->> [Updated Action, n1]

A set of discrete, parallel event routes can be simplified into a single route. This can be achieved if Event Publishers are grouped in a collection and accessed via index. This raises an event for the listening subscriber, but the difference is that now that we have grouped our Event Publishers, we have created a loose coupling. This comes in handy if we want to slightly alter outputs based on factors pertaining towards a target Event Publisher's properties stored in the collection, accessible via an index. Furthermore, if the 'Updated Action' can observe pertaining properties to the event raised, then only one notification is necessary for a broad range of output contingencies.

Further Applications

As communication protocols such as SPI, LAN and HTTPS describe principles for setting communications across respective network topologies, an EDD could be considered as a foundation behind some novel communication protocol. The real-time aspect of an event-driven system [2] makes it considerably suitable for real-time applications, such as push-notification features, or some signaling routine such as a traffic system.

Conclusions

The implementation of EDD could have uses outside just programming. However, within the context of programming the advantages range from a more simplified codebase, to increased robustness and more optimal implementation logic with a reasonable amount of coupling pertaining towards an almost self-documenting attribution between nodes within a system.

References

[1] https://learn.microsoft.com/en-us/dotnet/csharp/programming-guide/events/#events-overview
[2] https://www.infoq.com/articles/realtime-event-driven-ecosystem/