Analyze a quality attribute

The quality attribute we have chosen to analyze is Usability since it has shown itself to be one of the cheapest and easiest ways to improve the perceived quality of a system. And since we are all very new to SignalR we felt that analyzing the usability of the system is of most interest to us.

SignalR for ASP.Net Core is open source which means that anyone can start using it instantly, and for free, which is really nice.

Right at the start we faced some difficulties with getting started with SignalR. The documentation is severely lacking, and the getting started guides are either hard to find, or nowhere to be found. The best official getting started guide we could found was the one that came along with the announcement of SignalR for ASP.NET Core [10].

Due to the fact that SignalR for ASP.NET Core is still in the alpha phase of its release there are a lot of questions that has to be answered and issues that has to be resolved.

SignalR for ASP.NET Core is actually very simple to learn and use if you are already familiar with ASP.NET Core. If you however are new to both ASP.NET Core and javascript and HTML coding, you will face some difficulties with SignalR. This is purely due to the fact that the documentation is nowhere to be found.

Usability comprises the following areas:

- Learning system features
- Using a system efficiently
- Minimizing the impact of errors
- Adapting the system to user needs
- Increasing confidence and satisfaction

If we analyze SignalR for ASP.NET Core with these areas in mind, we quickly find that the first 3 areas at least are heavily affected by the lack of documentation and guides. The library is not hard to use, nor is it hard to use efficiently, just as long as you are familiar with ASP.NET Core and JS/HTML coding from before.

The impact of errors might also be bloated by the fact that you might have trouble knowing if you actually made an error, or if you encountered a bug. This is due to the nature of an alpha version of a project, and the volatility of these kind of projects. The lack of documentation makes it hard to know if you simply made a mistake, since there is no place to double check, or if you just faced a bug.

Adapting the system to user needs is not really anything to discuss, this is simply a library for developers that has been made as an Open Source project, so the system in itself is already adapted and tailormade for the users, which in this case is ASP.NET Core developers.

The confidence and satisfaction is another story, the library makes adding real time functionality to your applications incredibly simple, which increases your confidence and can be extremely satisfactory. The lack of documentation on the other hand might prove to be extremely frustrating and aggravating for the user.

Architectural support is not a problem in SignalR for ASP.NET Core, cancelling commands, undoing commands and functions like these are working fine. Since SignalR for ASP.NET Core is simply a library for ASP.Net Core developers, this makes simple usability factors like User Interface or how intuitive a system is, impossible to analyze. This is why we have to focus more on how easy it is to get started with this library, learn it and use it efficiently. Unfortunately a lot of this is purely dependent on the documentation which is, as stated before, severely lacking.

My personal conclusion is that this library is everything it says, it makes adding real-time web functionality to your applications extremely simple, much more simple then doing it without this library. The library is also quite straightforward and easy to use, if you are familiar with ASP.NET Core and JS/HTML coding. For new users on the other hand, it’s very hard to get started because you have close to no documentation.