Patel et al (2020) present an issue of a lack of expertise and “capacity to address accessibility” in software and such, a large quantity of software is published with no consideration or time given to software accessibility. This paper investigates the challenges technology professionals face in including accessibility and design in the development process. The findings from the study conclusively found that the majority of participants in the survey had not learned anything substantial and one of the few that did mentioned “Like if I were to think about how I applied what I learned from… [my institution] about accessibility I don’t personally feel I’d be confident in applying what I did learn in those courses like in the real world.” Showing that the already small size of people taught accessibility do not have the confidence to put it into practice. The assumption can then be made that most tools in the game development space suffer from the same lack of accessibility considerations and so action needs to be taken to consider them.

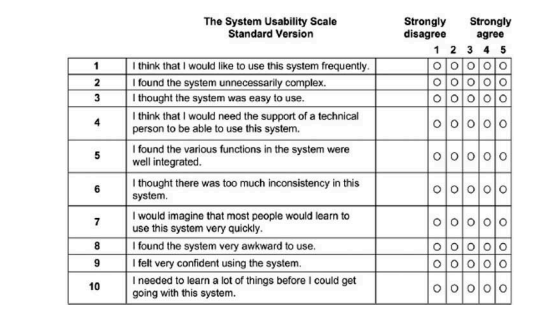
Weber et al (2019) study argues that the practice of model driven development which uses tools to “support and empower developers” are actively held back by the lack of usability as it acts as a obstacle for the adoption by novices and experts alike. Weber states a tool can only be successful “by being usable, supportive of their users’ goals, and by facilitating learning can they fulfil their purpose.” And follows up by addressing an issue within the usability research and open-source space where they fall short regarding usability. He says this lack potentially leads to a reduction in the productivity and/or comprehension of the tool and in its worst case “results in users simply not using the tool, which not only means a lot of development effort gone to waste, but also stops users from benefiting from the gains of MDE, even if they are aware of the theoretical benefits”. The study concluded the existence of “an abundance of usability issues” where they tackled all of them with the feedback of qualified test participants. They mentioned that their efforts were met with appreciation and engagement in the development and improvement of the accessibility.

Putnam et al (2012) surveyed UX/HCI professionals on the consideration they put into including or considering accessibility into their work. Results showed that 70% of the participants had some sort of consideration for accessibility whilst 19% of the participants said they should consider it and the remaining 11% had no consideration. This data seems rather inspiring but not every project, especially ones that are made by smaller teams have dedicated UX/HCI professionals and such the interface is left up to the developers which can lead to an improper implementation of accessibility features and the development of the tool without consideration for them and an attempt to add them to a mostly finished project in contrary to the better practice of implementing and designing for accessibility alongside the development. Putnam concluded that “considering accessibility and diverse users in HCI/UX professions is (1) a good decision on moral, financial and legal grounds and (2) well supported by organizations like the W3C.” and that many existing software’s are not accessible. They finish with emphasizing the importance of educational programs in IT that prepare students to consider and advocate for inclusive design.

Avouris’s (2001) workshop on software useability goes into the core aspects of usability in software and explains standards that must be met, these consist of “Understandability, Learnability, Operability, Attractiveness and Compliance”. Following from this is a discussion on measuring usability, a popular method is splitting a users feedback and performance into three categories, effectiveness, efficiency and satisfaction. Effectiveness looks into the amount of achieved goals, how many users successfully completed tasks they set out for and average accuracy of the completed tasks, Efficiency is measured by time to complete tasks and an average tasks completed by unit minute and finally satisfaction measured by a rating scale, frequency of discretion and frequency of complaints this factors of usability can be seen within the standardized questionnaire PSSUQ an alternative to SUS though SUS still focuses on the same three areas, effectiveness efficiency and satisfaction.

SUS is a standardised questionnaire developed by John Brooke in 1986 and was developed to take quick measurements of how people perceived the usability of computer systems, Brooke says in his 2013 retrospective that it was proven to be extremely simple and reliable for usability evaluations and was widely used as a standard even though it was never formally standardized. He then talks about how to get non bias responses he changed the tone of the statements from negative to positive “respondents read each statement and make an effort to think whether they agreed or disagreed with it.” A benefit of SUS compared to some other questionnaires for my purpose is that it has been around for nearly 3 decades and so has been studied and analysed on its performance and how to analyse the data. Brooke brings up a paper analysing around 3500 SUS results done by Bangor, Kortum and Miller (2009) which resulted in a decided score rating system (diagram below).   
A close-up of a score

Description automatically generated



CSUQ

A computer system quality survey

Description automatically generated

In a comparative analysis between game development engines by Christopoulou and Xinogalos (2017) they identify the growing broadness of the population who use game development engines saying, “people involved in the design and development of serious games come from various fields. For example, pedagogists and domain experts with limited, if any, coding skills are involved in designing and developing serious games”. this shows a need for development software and tools that are accessible for people without former programming knowledge, this is talked about in their conclusion where overall unity is a more accessible and beginner friendly software but requires C# knowledge whereas Unreal engine 4 does not require any programming knowledge as it uses visual scripting but overall is a more complex tool and is far more hardware demanding then unity and other tools. In their analysis of the different game development engines accessibility it talks about how Unity “seems to be the most usable game engine, providing the most free tutorials, examples and assets, while its community is very large”. This is brought down by unities lack of technical support and the fact its closed source compared to other mainstream game development engines such as Unreal Engine and Godot which can make developing tools for the engine harder though the source code for the paid version of unity is open. They ultimately concluded there is no best engine as they both have their pros and cons which is the consensus of any and all papers doing comparative analysis between engines.

A paper by Toftedahl and Engstrom(2019) initially set out to evaluate existing localization features within game engines but identify the complexitiy and inconnection between game engines and tools which made researching specific functions within the engine difficult leading them to change the direction of the paper to a “taxonomy of game engins and the tools that drives game production.