***End user Software Engineering***

* EUD = END USER DEVELOPMENT
* EUSE = END USER SOFTWARE ENGINEERING

Mansi

Introduction:

It all started with end-user programming.

End-user programming enables end users to create their own programs. Researchers and developers have been working on empowering end users to do this for a number of years, and they have succeeded: today, end users create numerous programs.

Bhavin

The “programming environments” used by end users include spreadsheet systems, web authoring tools, and graphical languages for creating educational simulations.

Uma

Some ways in which end users create these programs include writing and editing formulas, dragging and dropping objects onto a logical workspace, connecting objects in a diagram, or demonstrating intended logic to the system.

Chaitanya

research based on U.S. Bureau of Census and Bureau of Labour data shows that there are about 3 million professional programmers in the United States—but over 12 million more people who say they do programming at work, and over 50million who use spreadsheets and databases. the number of end-user programmers in the U.S. alone probably falls somewhere between 12 million and 50 million people—several times the number of professional programmers.

Mansi

Clearly then, end-user programming empowers—it has already empowered millions of end users to create their own software.

Bhavin

Unfortunately, there is a down side: the software they are creating with this new power is riddled with errors. In fact, evidence abounds of the pervasiveness of errors in software end users create. These errors can have significant impact. For example, one school faced a £30,000 shortfall because values in a budget spreadsheet had not been added up correctly.

Uma

Even when the errors in end-user-created software are non-catastrophic, however, their effects can matter. Web applications created by small-business owners to promote their businesses do just the opposite if they contain bad links or pages that display incorrectly, resulting in loss of revenue and credibility.

Chaitanya

Yeah. And The problem with end-user programming is that end users’ programs are all too often turning out to be of too low quality for the purposes for which they were created.

Mansi

Absolutely, but End-user software engineering does not mimic the traditional approaches of segregated support for each element of the software engineering life cycle, nor does it ask the user to think in those terms.

Bhavin

Instead, it employs a feedback loop supported by behind-the-scenes reasoning, with which the system and user collaborate to monitor dependability as the end user’s program evolves.

Uma

And This approach helps guard against the introduction of faults in the user’s program and if faults have already been introduced, helps the user detect and locate them. Because spreadsheet languages are the most widely used end-user programming languages to date.

Chaitanya

And here is the point right, as what we can say that the End-user software engineering (EUSE) is there as "end-user programming involving systematic and disciplined activities that address software quality issues" Attention to quality is important for EUP because poorly-written software can cause data loss, security breaches, financial loss, or even physical harm, even when the software is created by end-user developers.

Mansi

The software qualities relevant to EUSE are the same as those of interest to professional developers who sell their products. These qualities include reliability, performance, maintainability, reusability, privacy, and security.

Bhavin

Some qualities, such as maintainability and reusability, only become apparent after a program has been written and in operation for some time. Thus, EUSE combines the goal of EUP, which focuses on enabling end users to create software, with the concern for quality of that software across its entire lifecycle.

Uma

This lifecycle includes requirements, design, verification, debugging and code reuse (in addition to actual implementation, which has already been described above in the context of EUP tools).

Bhavin

Guys can we discuss the core software engineering concepts here?

Chaitanya

Yeah sure. likewise, you can consider the concept of

**Requirements and design**

Requirements describe what a program should do, and design refers to determining how a program should do it. For example, a requirement might be that a program should be able to sort a list of mailing addresses, and its design might detail the sorting algorithm to be used.

Mansi

That was really illustrative example you have given Chaitanya.

I would like to add some goals of it in EUD that it includes personalizing the way that an application or computer behaves, automating time-consuming tasks, performing computations that are hard to do accurately by hand, or communicating information.

Bhavin

Getting these requirements right is a critical aspect from the perspective of EUSE, because of its emphasis on quality. Professional developers are expected to investigate, document, and refine requirements before they start to design or code an application.

Uma

So as u say we can consider the example like

they might try to identify inconsistencies in requirements by applying one of several painstaking techniques (e.g., stakeholder review or formal modelling). In contrast, end users often live in their domain every day and know it very well, so they often already have an idea about what the requirements, and do not do any extra work to arrive at them, document them, or check them.

Chaitanya

And here EUSE comes into the picture to help the end user to overcome the advantages working professionals have over them regarding the quality of the product.

Uma

Yes, Chaitanya you got the point right.

Mansi

Then we also gave the concept of

**Verification and validation**

Verification and/or validation (V&V) cover activities attempting to make sure that a program does what it is supposed to do. Like testing is the most common approach for V&V (even among professional developers). One of the first works supporting V&V in EUD was to help users evaluate whether their programs contained bugs by encouraging end users to test strategically. Perhaps the most developed end-user testing approach is "What You See Is What You Test" which guides users through the process of systematically testing spreadsheets.

Bhavin

We can also say as an alternative approach for finding errors in programs is for the programming tool to automatically look for errors on the basis of types, dimensions, or units, etc.

Mansi

Yeah

Uma

Right Manasi and Bhavin, then as like u said for the validation, we also have the debugging.

Let me tell you more on it.

After a programming error is detected, the next step is to remove it by debugging. Some of the debugging techniques used by professional developers have been adapted for use in EUP tools. In addition to inserting "print" statements that display the value of variables as a program executes, end user developers can step through instructions one at a time, watching for incorrect operations.

Chaitanya

Correct Uma and

Assertions represent another important traditional technique that has been adapted for use in EUP: a user can insert a conditional into the code, and the program will call attention to that point if the conditional evaluates to false at execution.

Several EUP tools provide tight integration between testing and debugging. That is, they take care or they help the end user to get over the process of debugging.

Mansi

Yes, guys We also have reuse there

Like After code is written, reuse can speed the creation of later programs. Supporting reuse of end-user programs is challenging because end-user developers rarely have the opportunity or training required to design highly reusable programs. Another challenge is that end-user developers can make mistakes when creating programs or other files for tailoring applications, and reusing these can propagate errors across an organization.

Bhavin

So, Therefore, even though systems such as repositories or file servers can make it easy for end-user developers to post programs for others to reuse, it can be extremely difficult and time-consuming for other developers to evaluate the reusability of these programs. To help reduce the difficulty of reusing programs, models of what makes end-user programs reusable are now being developed in the hopes of helping users to search repositories for reusable programs related to the user's particular interests.

Uma

So, guys ok for the software engineering aspect for the end user but what u think about its future.

Chaitanya

**Future evolution**

While recent years have seen a considerable increase in novel proposals to address EUD issues, there is still a lot of work to be done to realise its full potential, and the continuous on-going technological evolution poses new challenges and opportunities to EUD but by Using the EUSE approach we can find the solution for most of the challenges using the scientific approach.

Mansi

Totally agree Chaitanya EUD with the help of the EDSW seem to have to bright things in near future.

Chaitanya

CONCLUSION.