Zhang Bowen, Zhang Rongliang

COLLABORATIVE SOFTWARE

This article is proposed to analyze the development, current use and potential use of collaborative software. It helps to understand in which particular situations, it might can applied.

Introduction

Collaborative software (or groupware) is application software designed to help people working on a common task to attain their goals, which means it allow separate users to make parallel operations towards the same project synchronously or asynchronously, while preserving every saved record by every user. This article contains five sections. In the following section, background and theory is briefly introduced. Section III classifies and evaluates the different categories of them, while section IV presents specific instances. Section V is the potential use of this Tech.

I. Backgoround

Thanks to division of labor, in the modern society we don't have to do everything individually in life, we only should focus on our own parts. And Collaborative software has become an indispensable tool in modern work and daily life. With the rise of remote work, complex projects, and the need for improved communication and transparency, collaborative software has become essential for teams to work together effectively. This process takes decades and the development is shown as follows:

1960s and 1970s: the collaborative software is developed to support communication and collaboration within research communities.

1980s: with the rise of PC and LANs, it became more widely available and popular.

1990s and 2000s: web-based collaborative software began to emerge.

Recent: cloud computing and mobile devices have expanded the capabilities and accessibility.

Overall, the development of collaborative software has been characterized by a continual evolution of features and capabilities, driven by advances in technology and changing user needs.

II. Theory

There are several theories and techniques that can be used to avoid shared resource conflicts in collaborative software. Here are a few examples:

Operational Transformation (OT): OT is a technique that allows multiple users to edit the same document or resource simultaneously without conflicts. It works by tracking each user's changes to the document and transforming those changes in real-time to ensure that they do not conflict with other users' changes. OT ensures that all users see the same version of the document, and that conflicts are resolved automatically as users make changes.

Conflict-free Replicated Data Types (CRDTs): CRDTs are a class of data structures that allow multiple users to edit the same resource without conflicts. CRDTs work by ensuring that all updates to the resource are commutative, meaning that they can be applied in any order without changing the result. This ensures that all users see the same version of the resource, even if they are working on different parts of it simultaneously.

Locking and Version Control: Locking and version control are techniques that allow users to prevent conflicts by controlling access to the shared resource. Locking allows users to "lock"specific parts of the resource, preventing other users from editing them. Version control allows users to track changes to the resource over time, making it easier to identify and resolve conflicts if they do arise.

Overall, the theory behind avoiding shared resource conflicts in collaborative software is based on the idea of ensuring that all users see the same version of the resource, regardless of who is editing it at any given time. Techniques such as operational transformation, conflict-free replicated data types, locking, and version control are used to ensure that conflicts are minimized and resolved quickly and efficiently when they do occur.

III. Instances

Collaborative software can be divided into different types based on their features and functionality. Some of the most common types of collaborative software include conferencing software, coordination software, and communication software. They can also be divided into synchronous (real-time) and asynchronous (non-real-time) software. Real-time collaborative software allows users to work together at the same time and includes audio-video communication systems and chat systems. Non-real-time software enables users to accomplish tasks together at different times and includes email, mailing lists, group calendars, etc.

Two examples about version control and real time collaborative software instances are discussed. The first one is Git which mainly used as a document version control, it can bring several benefits, including:

Version Control: Git provides a complete history of changes made to a document, allowing you to track changes over time, and easily revert to previous versions if needed.

Conflict Resolution: Git provides tools for resolving conflicts that may arise when multiple users are working on the same document simultaneously. This ensures that changes made by each user are correctly integrated into the final version of the document.

Backup and Restore: Git allows you to create backups of your documents, making it easy to restore documents to previous versions in case of accidental deletion or loss of data.

Transparency: Git provides transparency in document version control, enabling you to track who made changes to the document, when changes were made, and why changes were made. This can help maintain accountability and transparency in collaborative document projects.

Git can bring many benefits to document version control, including version control, collaboration, conflict resolution, backup and restore, and transparency. It is a powerful tool that can be used to manage documents efficiently and effectively.

The second one is Live share, one of the most commonly used scenarios for Visual Studio Live Share is "pair programming": two or more developers, working together on a shared task, with the goal of sharing knowledge, increasing team cohesion, and potentially, product quality. The exact look-and-feel of pair programming can differ significantly between teams and situations. Live Share enables a form of pair programming that allows developers to work on a shared goal, without removing their individual autonomy or environment preferences. This can lead to increased productivity, better code quality, and enhanced knowledge sharing.

IV. Potential use

In the future, collaborative software could be applied to remote surgery, enabling remote surgeons to collaborate with on-site medical teams in real-time. This could be particularly useful in situations where local medical expertise is limited, and remote specialists can provide valuable assistance.

Collaborative software could allow remote surgeons to access medical images and patient data in real-time, enabling them to provide guidance and advice to on-site medical teams. Collaborative software could also enable remote surgeons to collaborate with each other, sharing expertise and experience, and working together to solve complex medical problems. This could be particularly valuable in situations where rare or complex medical conditions require specialized expertise that may be available only in a limited number of locations.

To ensure the safety and effectiveness of remote surgical collaboration, advanced communication technologies, such as high-definition video and audio, and secure data transfer protocols, would need to be employed. Additionally, the software would need to be designed with strict security and privacy protocols to protect patient data and ensure compliance with regulatory requirements.

Overall, collaborative software might revolutionize the field of remote surgery, enabling remote surgeons to collaborate with on-site medical teams in real-time, improving patient outcomes, and expanding access to medical expertise.

V. Conclusion

Collaborative software can be applied in many fields to deal with complex and remote projects with high transparency, high efficiency, and more interaction. At the same time the collaborative software has been developing with a continual evolution of features and capabilities, driven by advances in technology and changing user needs.

Zhang Bowen, an undergraduate student of department of information technologies in automated systems, BSUIR, zhangbowen800@outlook.com

Zhang Rongliang, an undergraduate student of department of information technologies in automated systems, BSUIR, ZhangRL456@outlook.com

Scientific supervisor: Trofimovich Alexey Fyodorovich, Deputy Dean of FITU, Senior Lecturer of ITAS Department, BSUIR, trofimaf@bsuir.by