**Sisteme de editare colaborativa (SEC)**

Sunt sisteme ce intermediaza colaborarea intre participanti in timp-real. Ele permit mai multor utilizatori aflati in locatii diferite sa editeze acelasi document in acelasi timp.

**1. Caracteristicile unui SEC**

Articolul X defineste caracteristicile pe care trebuie sa le indeplineasca un SEC. Ele sunt:

1. **Raspuns rapid** –raspunsul la actiunile locale ale utilizatorului este rapid ( in cazul ideal, la fel de rapid ca si al aplicatiilor cu un singur utilizator ).

2. **Control distribuit –** utilizatorii initiaza operatiile de editare de pe noduri diferite, comunicarea se realizeaza prin retea.

3. **Editare libera –** utilizatorii pot edita liber orice parte a documentului in orice moment. Sistemul mediaza interactiunea dintre utilizatori si inlesneste rezolvarea conflictelor ce rezulta din actiunile concurente.

**2. Strategii de implementare ale unui SEC**

Pentru implementare unui SEC, sunt cunoscute urmatoarele abordari:

1. **Bazata pe jetoane** – un singur utilizator detine jetonul in baza caruia poate edita documentul partajat intr-o anumita fereastra de timp. Accesul la jeton poate fi controlat prin mecanisme interne, implementate in aplicatie, sau prin mecanisme externe, stabilite de utilizatori. Aceasta abordare nu este potrivita sistemelor ce necesita un grad inalt de paralelism si inhiba editarea libera.

**2. Bazata pe lacate** – un obiect (ex: in documentul text – o linie, un cuvant, o sectiune) este rezervat inainte de editare. Un singur utilizator poate edita obiectul intr-o anumita fereastra de timp. Implementarea bazata pe lacate nu permite editarea libera a documentului partajat si degradeaza timpul de raspuns prin operatiile aditionale de rezervare si eliberare a obiectelor.

**3. Bazata pe tranzactii** – mecanismul tranzactiilor este un domeniu bine cercetat si aplicat cu succes in sistemele de baze de date. Rezultate bune au fost obtinute si in sistemele non-real time. In sistemele real-time, mecanismul ACID nu permit editarea libera si degradeaza raspunsul sistemului din cauza complexitatii implementarii mecanismului ACID intr-un mediu distribuit.

**4. Bazata pe serializare** – operatiile sunt executate in aceasi ordine pe toate nodurile, astfel rezultatul final va fi identic. Implementari de acest gen folosesc cel mai adesea altgoritmi distribuiti bazati pe ceasuri logice sau bazati pe un coordonator central. In al doilea caz toate operatiile sunt trimise nodului central, care le distribuie secvential. Problemele prinicipale ale acestei abordari sunt existenta unui singur punct de esec si degradarea timpului de raspuns din cauza trecerii operatiilor prin nodul central.

**5. Bazate pe executia optimista** – operatiile sunt executate in momentul receptionarii lor. In cazul in care la primirea unei noi operatii se constata ca operatiile nu au fost executate in ordine pentru a repara rezultatul exista doua solutii: 1. Efectul operatiilor executate in ordine gresita este anulat, este executata ultima operatie primita, dupa care sunt executate celelalte operatii in ordine. 2. Ultima operatie primita este transformata, astfel incat prin aplicarea ei sa se obtina acelasi rezultat ca in cazul in care ar fi fost aplicata in ordine corecta.

6. Articolul X prezinta o noua abordare de implementare a unui SEC. Fata de strategiile prezentate mai sus, aceasta se distinge prin faptul ca reuseste sa acopere toate cele trei caracteristici prezentate la punctul 1. [Ce isi propune abordarea, cateva lucruri in plus ]Deoarece este cea mai noua si cuprinzatoare abordare, o sa detaliez fundamentele teoretice si aplicatiile acesteia.

**6. 1 Mecanismele de baza**

Sunt cele pe care se bazeaza toate celelalte scheme de control al concurentei. Toate schemele prezentate mai jos se aplica pentru cazul in care in sistem sunt N participanti, fiecare participant fiind identificat printr-un intreg de la 0 la N-1.

* + 1. **Executia operatiilor locale** – o operatie este executata pe un nod imediat dupa generare. Astfel se asigura un raspuns rapid al sistemului si executia in ordine a operatiilor locale. Pentru a stabilii timpul la care o operatie a fost executata se foloseste un sistem de ceasuri logice.
    2. **Ceasurie logice vectoriale Lamport –** fiecare nod pastreaza un vector cu N componente, cate o componenta pe nod. Pe baza acestor vectori se aplica o stampila de timp fiecarei operatii execute in sistem. Mai multe detalii despre modul de functionare al ceasurilor Lamport sunt in articolul Y.

**6.1.2.1 Teorema:** Fie doua operatii Oa si Ob, stampilate cu SVOa si respectiv cu SVOb, Oa ->Ob, daca si numai daca pentru orice i , SVOa[i] SVOb[i] si exista un j pentru care SVOa[j] SVOb[j]

**6.2** **Convergenta** – la inchiderea unei sesiuni de colaborare, replicile documentului partajat sunt identice. Convergenta este asigurata de urmatoarea schema:

**6.2.1** Definirea uei relatii de ordonare totala intre operatii, bazata pe stampila de timp si pe identificatorul nodului.

**6.2.2** Mentinerea pe fiecare nod a unui buffer cu operatiile executate**.**

**6.2.3** Cand o operatie primita nu este in ordine, toate operatiile deja executate, care ar fi trebuit sa ii urmeze sunt anulate, este executata respectiva operatie, si apoi reexecutate operatiile anulate.

Exemplu: **Initial:** Buffer = Oa | Ob | Oc | Od , SVOa < SVOb , SVOb < SVOc, SVOc < SVOd.

* Este primta operatia Oe cu SVOe > SVOb si SVOe < SVOc.
* Efectul operatiilor Oc si Od este anulat. **Rezultat:** Buffer = Oa | Ob
* Este aplicata Oe. **Rezultat:** Buffer = Oa | Ob | Oe
* Sunt aplicate in ordine Oc si Od. **Rezultat:** Buffer = Oa | Ob | Oe | Oc | Od

**6.3. Mentinerea cauzalitatii operatiilor** – pentru orice pereche de operatii Oa si Ob, daca Oa ->Ob, Oa este executat inaintea lui Ob pe toate statiile. Desi schema de convergenta ne asigura ca rezultatul final al unei sesiune de editare este consistent, ea nu asigura consistenta in timpul sesiunii. Mentinerea consistentei documentului pe tot parcursul procesului de editare este important pentru utilizator pentru ca acesta sa isi poata forma o idee despre ceea ce se intampla in sesiune. Intreaga schema este prezentata pe larg in articolul X la pagina 8.

**6.4** Mentinerea intentiei operatiilor (sau utilizatorului) - pentru orice pereche de operati Oa si Ob, daca Oa || Ob, executia lui Oa si Ob in oricare ordine va mentine intentiile operatiilor.

Un sistem de editare colaborativa este consistent daca mentine urmatoarele trei proprietati:

**Aplicatie, sistemul REDUCE**

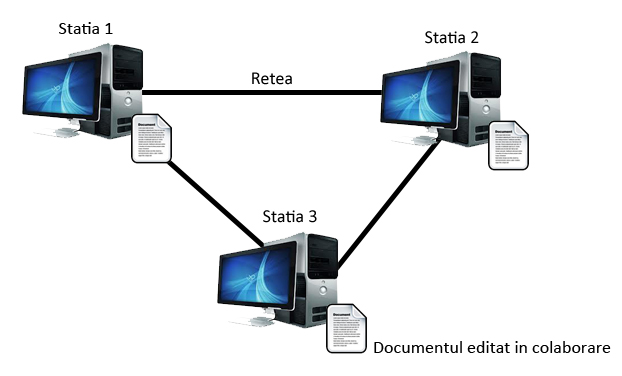
Mentinerea consistentei documentului editat este una din sarcinile cel mai dificil de realizat.

Sistemul REDUCE (REal-time Distributed Unconstrained Cooperative Editing) intruneste toate caracteristicile unui sistem de colaborare, caracteristici prezentate mai sus.

REDUCE foloseste ca mediu de comuicare Internet-ul si a fost dezvoltat in Java.

Detalii:

**Arhitectura sistemului REDUCE**

****

**Fig 2.1 – Arhitectura conceptuala a sistemului REDUCE**

Autorii propun o arhitectura in care documentul editat la comun este replicat la nivelul fiecarui punct de lucru. Intr-o astfel de arhitectura problema cea mai mare o reprezinta mentinerea consistentei documentului replicat.

Documentul editat in colaborare este replicat la nivelul fiecarei statii. peratiile sunt executate prima data la nivel local, rezultatul lor este vizibil imediat, dupa care sunt trimise la celelalte statii.

Prima problema care apare este inconsistenta documentului editat. In articolul X sunt prezentate

**Sisteme de editare grafica colaborativa (SEGC)**

Un tip particular de astfel de sistem este sistemul de editare a documentelor grafice bazat pe obiecte. Un astfel de document contine unul sau mai multe obiecte grafice. Fiecare obiect este reprezentat de un set de atribute, cum ar fi: tip, marime, pozitie, culoare, grup, etc. Operatiile de editare includ crearea, modificarea si distrugerea unor astfel de obiecte. Intr-un astfel de sistem conflictele apar atunci cand operatii concurente sunt generate in diferite locatii pentru a modifica acelasi obiect sau acelasi atribut al aceluiasi obiect. Granularitatea cotrolului depinde de implementarea sistemului. In cazul sistemului GRACE, modificarile se fac la nivel de atribut.

**Lacate in SEGC**

Executia unor operatii conflictuale poate duce la aparitia de inconsistente in cazul in care nu sunt rezolvate corespunzator. Pentru a prevenii conflictele care sunt mai dificil de rezolvat in sistemul GRACE este implementat un mecanism de lacate.

Tehnica bazata pe lacate a fost dezvoltata initial pentru a evita conflictele in sistemele de baze de date. Conceptul de lacat este usor de inteles de catre utilizatori, un obiect pe care s-a pus un lacat nu poate fi accesat decat de catre detinatorul lacatului.

Exista sisteme de editare grafica colaborative ce implementeaza aceasta tehnica, dintre ele le amintesc pe urmatoarele: Ensemble, Group Draw, GroupGraphics si GroupKit. In aceste sisteme inainte de orice operatie ce modifica un obiect trebuie obtinut un lacat exclusiv. Iau ca exemplu situatia in care un utilizator doreste sa mute un obiect in scena, pasii pe care trebuie sa ii urmeze sunt urmatorii: trimite o cerere de obtinere a unui lacat exclusiv pe obiectul dorit, asteapta primirea raspunsului, in caz afirmativ poate deplasa obiectul pe scena in pozitia dorita, la finalul operatiei de mutare elibereaza lacatul. Acest comportament garanteaza ca un singur utilizator va modifica un obiect la un anumit moment in timp, ceea ce previne aparitia conflictelor.

**Lacate in GRACE**

Sistemul GRACE pune la dispozitia utilizatorilor un mediu de colaborare ce permite aparitia conflictelor, insa sansele de conflict sunt reduse. Intr-un asemenea sistem impunerea obligatorie a unui mecanism pe baza de lacate ar fi ineficient, scazand gradul de responsivitatea al aplicatiei. In ciuda schemei de versionare multipla aplicata in sistem ce rezolva multe dintre problemele implicate de activitatea colaborativa exista zone in care lacatele sunt de folos. Lacatele sunt utilizate in situatia in care aparitia unor conflicte este iminenta. In aceste cazuri aplicarea de lacate exclusive va prevenii aparitia lor. In cazul sistemului GRACE consistenta este mentinuta prin aplicarea strategiei de versionare multipla, insa functionarea per ansamlu a sistemului este imbunatatita de folosirea lacatelor deoarece reduce numarul de conflicte ce trebuie rezolvate.

**Dificultati**

Una din dificultatile ce apare in cadrul sistemelor ce implementeaza mecanisme de lacate este maniera in care sistemul rezolva cererile concurente de obtinere a aceluiasi lacat.