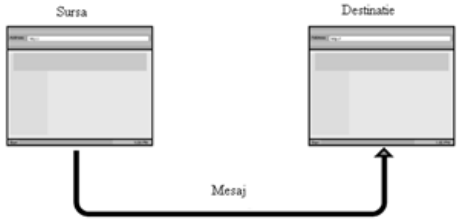
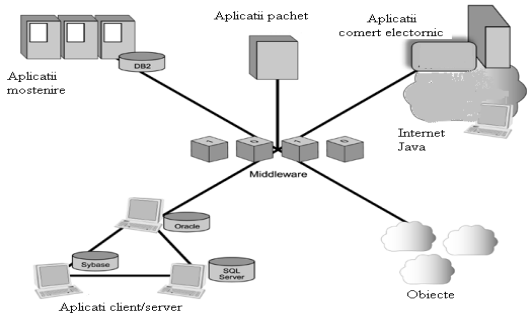
Tema P.S.S.C.

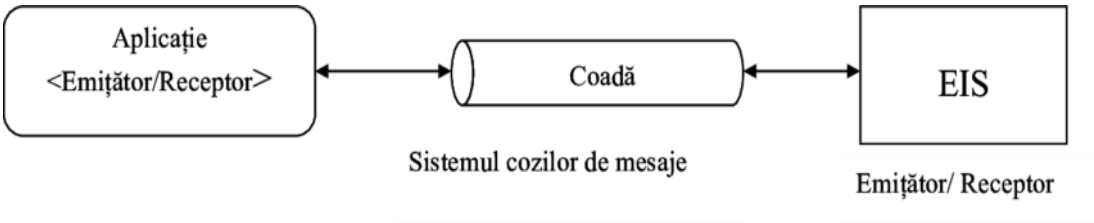
- **Middleware** este un mecanism care permite unei entităţi (bază de date sau aplicaţie) să comunice cu o altă entitate (sau cu mai multe entităţi).
- Există două modele de middleware: logic şi fizic.
- **Modelul logic** descrie cum are loc transferul de date conceptual. Configuraţiile asociate modelului logic sunt:
- unu la unu;



- mulţi la mulţi;



- sincron versus asincron;
- **Modelul fizic** descrie metodele precum şi tehnologia folosite pentru transferul de informaţie. Modelului fizic îi sunt asociate modele bazate pe mesaje.



Folosirea cozilor de mesaje pentru integrarea EIS ilustrează comunicarea bazata pe coadă de mesaje. Acest tip de comunicare, care mai este numită şi mesagerie punct – la – punct, presupune existenţa unei aplicaţii ce trimite mesaje într-o coadă de mesaje. În comunicarea bazată pe coadă de mesaje, o astfel de coadă independentă faţă de aplicaţiile emiţător şi receptor se comportă ca un buffer de mesaje între aplicaţiile comunicante . Aplicaţia emiţător trimite un mesaj în această coadă, iar aplicaţia receptor primeşte mesajele din aceeaşi coadă.

Cum se desfasoara comunicatia asincrona?

Tehnologiile middleware bazate pe cozi mesaje sunt de cele mai multe ori implementate ca și un server care poate gestiona mai mulți clienți conectați simultan. Astfel pentru a se realiza decuplarea expeditorului și a destinatarului sunt folosite una sau mai multe cozi în care expeditorii pot pune mesaje, respectiv din care destinatari pot citi mesaje. Un astfel de server poate să gestioneze mai mult cozi simultan, respectiv mai multe mesaje transmise simultan prin intermediul firelor de execuție organizate in pool-uri de fire de execuție. Un singur proces poate trimite mesaje către mai multe cozi, respectiv fiecare coadă poate fi interogată de unul sau mai mulți destinatari. Identificare cozilor se face pe baza numelor.

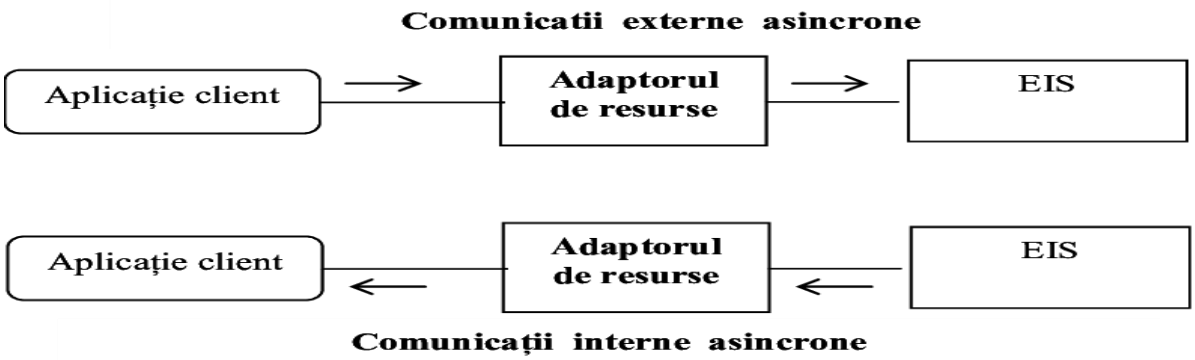
• In comunicatia asincrona operatia **send** nu este cu blocare. Astfel ca expeditorul isi va putea continua executia imediat ce mesajul a fost copiat intr-un tampon local pentru a fi transmis.

• Operatia **receive** poate avea variantele cu blocare sau fara blocare.

• In varianta fara blocare procesul destinatar va avea la dispozitie un tampon pentru a receptiona mesajul. El va trebui sa fie informat (engl.notified) cand acest tampon a primit un nou mesaj, fie prin tehnica polling, fie printr-o intrerupere.

• Intr-un sistem care suporta fire de executie, cum este Java, este mai raspandita varianta apelului receive cu blocare. Un fir separat poate executa operatia receive, in timp ce celelalte fire ale aplicatiei vor ramane active.

• Desi apelul receive fara blocare pare mai eficient, el aduce o complexitate suplimentara in procesul destinatar, care trebuie sa poata prelua mesajul in afara fluxului sau curent de control.



În comunicarea asincronă o aplicaţie apelează funcţia dirijată pentru a crea un nou articol de contabilitate a clienţilor în cadrul EIS. Aplicaţia face apelul dirijat şi imediat se întoarce şi continuă propria procesare. Funcţia dirijata este trimisă la EIS. Acesta prelucrează funcţia şi întoarce ca răspuns aplicaţiei o oarecare informaţie ca o invocare asincronă separată. Adaptorul de resurse transmite apelul asincron de la EIS la aplicaţie. Un fapt important de ţinut minte este acela că aplicaţia nu îşi suspendă execuţia cât timp funcţia dirijată rulează în cadrul EIS. Astfel aplicaţia îşi continuă activitatea şi primeşte la un moment dat notificări cu privire la rezultatele invocării funcţiei dirijate. Aşadar, un EIS poate invoca sau apela asincron o aplicaţie.