Informatikai rendszerek üzenetalapú integrációs mintái

Óravázlat

Szerkesztette:

Dr. Nehéz Károly
egyetemi docens

Miskolci Egyetem
Informatikai Intézet
2021

Tartalomjegyzék

Tartalomjegyzék 2
Bevezetés 3
Az integráció nehézségei 4
Hogyan segíthet egy Integrációs-minta 6
Integráció stílusok 7
Az alkalmazás integráció előfeltételei 7
Alkalmazások függősége 7
Behatások a meglévő rendszerekben 8
A felhasznált technológiák kiválasztása 8
Adat formátum 8
Az adatok frissessége 8
Adat vagy funkció 9
Távoli kommunikáció 9
Integrációs lehetőségek 10
Fájl átvitel 10
Osztott adatbázis 10
Távoli eljáráshívás 10
Üzenetváltás 10
Fájl átvitel 11
Osztott Adatbázis 12
Távoli eljáráshívás 13
Üzenetváltás 14
Üzenetkezelő rendszerek 15
ÜKR főbb alkotóelemei 15
Csatornák / üzenetsorok 15
Üzenetek 16
Csövek és szűrők 16
Útvonalválasztók 17
Átalakítók 17
Végpontok 18
Üzenet csatornák 19
Pont-pont csatorna 19
Publikálás-feliratkozás csatorna 20
Rögzített adattípusú csatorna 21
Érvénytelen üzenet csatorna 22
Halott levél csatorna 23
Garantáltan kézbesítő csatorna 24
A WGRUS példa-vállalat 26
Elvárt eredmények 28
A megvalósítás 28
A bejövő kérések kiszolgálása 30
A karbantartási műveletek 31

Bevezetés

Egy átlagos nagyvállalat informatikai rendszere általában több száz, de akár több ezer egyedi fejlesztésű szoftvert, külső cégtől származó szoftvermegoldásokat, régi, örökölt rendszer komponenseket (legacy components/systems) tartalmaz, valójában ezekből épül fel. Gyakran többféle operációs rendszert használnak, telephelyeik néha földrajzilag is távol helyezkednek el egymástól. Nem ritka az sem, hogy több száz weboldalt, több vállalatirányítási rendszert és számtalan kisebb ERP szoftvert (pl. raktárnyilvántartó) is használnak párhuzamosan.

Általában elmondható, hogy üzleti alkalmazásokat készíteni nehéz. Létrehozni egy hatalmas alkalmazást, ami a teljes üzleti funkcionalitást lefedi és változásait is folyamatosan követi, majdnem lehetetlen. Ameddig nagy ERP rendszerkészítők egyre összetettebb szoftvereket hoztak létre, addig még a legnagyobb szoftverek (mint például az SAP, Oracle, Peoplesoft) sem fedik le egy nagyobb termelő vállalt teljes funkcionalitását, információáramlását.

Minden funkció ellátására optimális megoldást igyekeznek választani a cégvezetők, az egyes részfeladatokat teljesen különböző szoftver-termékek látják el, gyakran különböző operációs rendszer platformon, egyre újabb megoldásokat alkalmazva, az integrálást segítő kapcsolódási pontok nélkül. (A legtöbb szoftver gyártó a saját termékét egy független önálló egységként értelmezi, nem tesz a könnyebb integrálhatóságért, hiszen ez gyakran nem is érdeke). Bár ez a tendencia talán javult az utóbbi években.

Az integráció nehézségei

„Enterprise integráció” alatt definíció szerint: **több különböző platformon futó alkalmazás összekapcsolását értjük.**
EAI (Enterprise Application Integration) szoftver gyártók általában eszközöket szolgáltatnak a legtöbb platform és a legtöbb programozási nyelv összekapcsolásához, valamint, a legelterjedtebb üzleti alkalmazások saját programból való eléréséhez szükséges interfészeket. Ezek az eszközök azonban az integráció problémakörének csak egy kis részét fedik le, mivel ez messze túlmutat az üzleti és a technikai kihívásokon.

- Egy vállalati szintű szoftverintegrációs projekt véghezvitele nagy változtatást kényszeríthet ki a vállalati működésben is, mivel egy nagyvállalat informatikai rendszere az egyes részlegeken belül önállóan fejlődik és változik (például a számlázás részleg informatikai eszközei teljesen függetlenek a felhasználói visszajelzésekkel foglalkozó részleg eszközeitől), ezért az integráció nagy megkötést jelent, hiszen az összekapcsolás után már nem lehet egyedileg módosítani az összetevőket, mert minden rendszer egy új, nagy egész része, a módosítások befolyással vannak minden más alrendszerre is.

- Az integrációs folyamat megvalósítása nagy anyagi kockázattal járhat, mert a meglévő, jól működő rendszereknek akár az időleges összeomlása is hihetetlen nagy anyagi veszteségeket jelenthet, ezért az új rendszer bevezetése rendkívül nehéz, általában csak lépésről lépésre érdemes megvalósítani. (megjegyzés: az örökölt rendszerek „legacy systems” is azért alakulnak ki, mert nem vállalja fel a senki a cseréjük kockázatát)

- Az integrációt végző csapat általában kis létszámú, és gyakran kevés módosítási lehetősége van az integrálandó szoftverre, mivel az más gyártótól származik vagy olyan „örökölt rendszer”, melynek sokszor már a forráskódja sem elérhető, ezért senki nem ismeri a működését, ezért szinte lehetetlen vagy irreálisan drága lenne változtatni rajta. Gyakran visszatérő probléma, hogy bizonyos programokat egyszerűbb lenne újraírni, csak, hogy jobban illeszkedjenek a rendszerbe, de ez nem lehetséges technikai, vagy cégpolitikai okokból: egyszerűen nem tudják vagy nem akarják

megfizetni az újra-kifejlesztés költségét kifizetni egy olyan rendszernek (ületi komponensnek) ami jelenleg kifogástalanul működik, csak éppen ősrégi vagy csak ősrégi operációs rendszeren futtatható. A régi rendszer használatóból eredő komoly üzleti kockázatokat pedig nem ismerik fel időben.

- Annak ellenére, hogy bizonyos integrációt segítő rendszerek vagy technológiák (mint például az XML, XSL és a Web szolgáltatások) nagyon elterjedtek, mégis gyakran ezek okozzák a problémák jelentős részét. Az szoftverrendszereknek ugyanis gyakorta jelennek meg újabb és újabb verziói, különböző kiegészítések és extra eszközök, amelyek gyakran nem kompatibilisek (vagy nem örökké kompatibilisek) egymással. Hasonló a helyzet az úgynevezett „szabványos" eszközökkel is, mint például a minden platformon és programozási nyelven elérhető XML-RPC technológiával, ami a gyakorlatban csak akkor működik megfelelően, ha csak egy bizonyos gyártó implementációit használjuk, mert a különböző gyártók „szabványos" implementációi gyakran nem kompatibilisek egymással.

- Az XML alapú kommunikáció körüli félreértések is nagy problémák forrásai. Sokan az XML –t egy „önmagát definiáló adatszerkezetnek" értik, részben igaz is. A félreértés azonban abban rejlik, hogy az XML szó-szerint csak(!) az adat szerkezetét, annak nyelvtanát adja meg, az értelmezését, vagyis a jelentését nem. Olyan ez mintha definiálnánk egy közös nyelvet, amelynek a szavait mindenki érti. Ez a helyes kommunikáció alapja, de a nagyobb probléma az, hogy a szavak átvitt értelmét is egyeztetni kell, ami gyakran nehéz lehet. A szó „account" jelentése a különböző rendszereken más és más lehet így gyakran nagyon részletes magyarázat szükséges egy-egy adat jelentésének megértéséhez.

- A probléma nagyvállalti természetéből következően egy ilyen rendszerben a hibák felderítése, kijavítása is sokkal nehezebb, mint egy hagyományos rendszerben mivel nagy szakmai felkészültséget, tapasztalatot igényel.

- Kifejleszteni és üzemeltetni egy vállalti informatikai rendszert, nagyon sokféle IT szaktudást igényel, ami gyakran nem áll rendelkezésre az üzemeltetőknél, ezért a

nagy, integrációs átállás okozta problémák koordinálása nagyon nagy vezetői tapasztalatot és szaktudást igényel.

Mindent egybevéve, a vállalati szoftverek (enterprise) integrációja rendkívül fontos feladat, a legmodernebb üzleti megoldások alkalmazása is elképzelhetetlen nélküle, azonban ez rendkívül megnehezíti az informatikai üzemeltetők/részlegek életét, nagy kihívást jelent még a legtapasztaltabb fejlesztés-vezetők számára is. Ezért fontos, hogy ha rendszereket kell integrálnunk, akkor lehetőleg olyan megoldásokat válasszunk, ami a legnagyobb eséllyel megvalósítható és működtethető.

Hogyan segíthet egy Integrációs-minta

Nincs egyszerű válasz az integráció körüli kérdésekre. Azonban megfigyelhetünk olyan szakembereket, akik sokkal eredményesebbek, mint mások. Ezek a fejlesztők többnyire korábbi tapasztalataikat igyekeznek ráilleszteni az aktuális feladatra, vagyis a megoldandó feladatot olyan alap-problémákra osztják fel, amelyeket már ismernek és korábban sikeresen leküzdöttek. A továbbiakban ilyen alapproblémákat és azok megoldásait mutatjuk be egyszerű példákon keresztül.

Integráció stílusok

Az integráció feladata, hogy véglegesnek gyártott eszközöket egyesítsünk olyan új célok és funkcionalitások elérésére, amelyek csak az egyes eszközök együttes használatával érhetőek el. Ez a fejezet bemutatja az összes gyakorlatban használt integrációs eljárást, egy áttekintést adva arról, hogy milyen alapeszközöket tudunk felhasználni a problémák leküzdéséhez.

Az alkalmazás integráció előfeltételei

Mitől lesz jó egyintegrációs megoldás? Minden eset más és más, szinte kivétel nélkül egyedi megoldásokra van szükség, ezért teljességgel kizárható egy „mindenre jó” minta létezése. Azonban mielőtt megnéznénk a lehetőségeinket, gondoljuk végig, hogy valóban szükség van e integrációra. Ha a feladat megoldható egy önálló alkalmazás kifejlesztésével, akkor ez mindenképpen egyszerűbb megoldás, mint összekapcsolni a meglévő rendszereket. Az integráció azonban gyakran elkerülhetetlen, ha a cég alkalmazottai, üzletfelei és ügyfele felé egy egységes egyesített funkcionalitás rendszert akarunk biztosítani. Ezért, ha már biztosak vagyunk az integráció szükségességében, van néhány stratégiai döntés, amit meg kell hoznunk:

Alkalmazások függősége

Az alkalmazások kezdetben függetlenek, egymás zavarása nélkül változhatnak, fejlődhetnek, azonban az összekapcsolásuk egy nagy rendszerré, komoly függőséget okoz, többé-kevésbé megszünteti az egyes alkalmazások önállóságát. Ez rugalmatlanná, sőt továbbfejleszthetetlenné teheti a teljes rendszert ezért már az első lépések megtétele előtt mérlegelnünk kell a kialakuló függőségek nagyságát, jövőbeli hatásukat.
Ez a gyakorlatban azt jelenti, hogy a programokat összekötő interfészeket a lehető legáltalánosabbá kell tervezni, teret hagyva az implementáció jövőbeni változásainak.

Behatások a meglévő rendszerekben

A már meglévő eszközöket a lehető legkisebb módosítás elvégzésével kell alkalmassá tenni a rendszerben történő működésre, elkerülendő, hogy a jól működő eszközt elrontsuk. Ez a megközelítési mód a programozás hatékonyságát is növeli, azonban gyakran nehéz előre meghatározni azt az irányt, amely a fejlesztői munka folyamatában a leghatékonyabbnak bizonyul. A legkisebb behatással járó integrációs pontokat előzetes vizsgálatokkal kell felmérni.

A felhasznált technológiák kiválasztása

Az integrációhoz nagyon sok eszköz áll rendelkezésre, mielőtt választanánk mérlegelni kell ezek árát, a szükséges betanítási időket, az esetleges szállítótól való függést. Bizonyos esetekben egyszerűbbnek olcsóbbnak vagy hatékonyabbnak tűnhet bizonyos eszközöket házilag előállítani, mert így nem kell megvásárolni és megtanulni egy másik szoftvergyártó termékét, azonban ez gyakran a költségesebb út, mert a tapasztalatok szerint könnyen a probléma alulbecslésének hibájába eshetünk.

Adat formátum

Mindenképpen szükséges egy egységes adat formátum a sikeres kommunikációhoz. A probléma ezzel az, hogy a legtöbb összetevő belső kommunikációs, és adatfeldolgozási formátumai nem megfelelőek a mindenütt történő használatra, így általában egy teljesen új formátumra van szükség. Ez az új adat-formátum adapterek segítségével könnyen lefordítható kell, hogy legyen az egyes összetevők belső adat-formátumaira. További problémát jelenthet a közös adat-formátum fejlődésének, későbbi átalakításának nehézsége, ezért már a tervezésnél figyelembe kell venni, hogy lehetőleg semmilyen irányba ne tegyünk a szükségesnél több megkötést.

Az adatok frissessége

Nagy adatmennyiségek megosztása mindenképpen időigényes feladat, amely idő alatt az egyes alkalmazások esetleg elévült adatokat használhatnak. Ha viszont a kommunikációt kis adatcsomagok cseréjére alapozzuk, akkor a hálózati kommunikáció miatt jelentkezhetnek

veszteségek. A kommunikáció során cserélt adatok mennyiségét, és az adatcseréhez szükséges időket mindenképpen előre meg kell számolni mert ha csak a rendszer kifejlesztése után észleljük hibás működésként azt, hogy a rendszer nagy terhelés esetén hibás, elévült adatok alapján dolgozik, akkor már átalakítani a teljes kommunikációs rendszert nagyon költséges lehet.

Adat vagy funkció

A legtöbb integrációs rendszer magában foglal megoldásokat, amelyekkel nem csak adatokat, de funkciókat is megoszthatunk. Ez növeli a rendszer absztrakció szintjét, de mivel a távoli eljáráshívások megvalósítása nagyon hasonlít a helyi függvényhívásokra, így ez gyakran félrevezető lehet.

Távoli kommunikáció

Egy lokális kommunikáció során nem sokat kell törődnünk esetleges tűzfalakkal, vagy az adatok titkosságának védelmével. Távoli gépek között történő adatcsere viszont lehallgatható, a tűzfalak megakadályozhatják az átvitelt.

A távoli kommunikáció mindenképpen hosszabb ideig tart, és sokkal gyakrabban sérülhet a kommunikáció, így a használata külön figyelmet igényel. A távoli rendszerrel való kommunikációban csak az aszinkron kommunikáció működhet, mivel nem garantálhatjuk, hogy minden gép be van kapcsolva az üzenetküldés pillanatában.

Az ilyen bonyolult rendszerek tervezése, fejlesztése és hibajavítása nehéz, és külön szaktudást, tapasztalatokat igényel.

Integrációs lehetőségek

Az egyes alaptípusok történetileg egymástól függetlenül, folyamatosan fejlődtek.

Fájl átvitel

Minden alkalmazás, a megosztandó adatait fájlokba szervezve tárolja le, továbbá eléri a többiek megosztott fájljait, és felhasználja azok tartalmát.

Osztott adatbázis

Egy közös, minden alkalmazás által elérhető közös adatbázisban vannak letárolva az adatok, amelyeken minden, a kommunikációban részt vevő összetevő (komponenes) írhat és olvashat.

Távoli eljáráshívás

Minden alkalmazás megosztja bizonyos függvényeit, hogy azokat más alkalmazások is elérjék/alkalmazhassák. Ezt a megoldást nem csak adatcserére, de a megosztott funkciók révén, bizonyos tevékenységek, pl.: programrészletek megosztására is használhatjuk.

Üzenetváltás

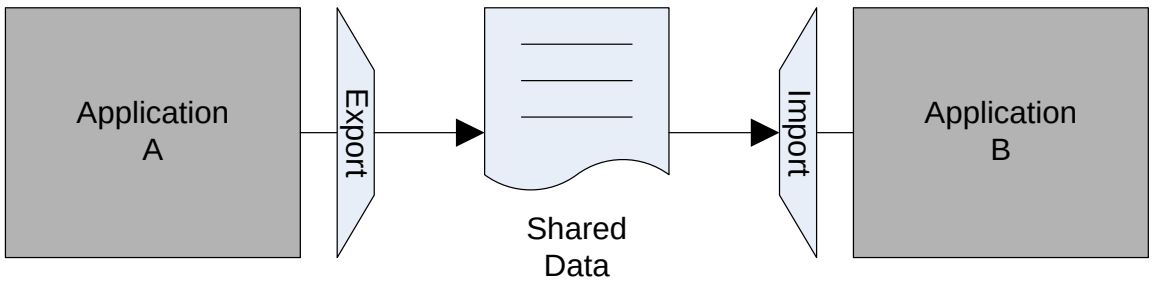
Minden alkalmazás egy közös üzenetküldő rendszerre csatlakozva üzeneteket küld és fogad. Az üzenetváltás lehet adatcsere, de valamilyen feladat elvégzésére irányuló kérelem is.

A következő fejezetek részletesen bemutatnak egy-egy minta példát minden alaptípusra. Nem helyes az a kérdés, hogy egy adott helyzetben melyik minta illeszthető leginkább az adott problémára, mert általában minták kombinálásával, az alapesetek egyidejű használatával érhetjük el a legjobb eredményt.

Fájl átvitel

A fájl átvitel legfontosabb előnye az összes többi eljárással szemben a viszonylagos egyszerűsége. Bizonyos esetekben ez az egyetlen rendelkezésre álló megoldás, hogy sikeresen összeköthessünk alkalmazásokat. A fájlok, és azok hálózaton történő továbbítása minden platformon létezik, ez a megoldás nem igényel speciális szoftvereket, maximum elérési engedélyeket.

A fájl átvitellel történő adatcsere folyamata rendkívül egyszerű, az adatokat a forrásalkalmazásból exportáljuk egy fájlba, amit aztán tovább küldhetünk vagy megosztott tároló helyen tárolhatunk, esetleg bármilyen más úton eljuttathatjuk a célalkalmazásig, ahol már csak az adatok importálását kell megoldanunk.



1. ábra: Fájl alapú integráció

A fájl alapú megoldás, egyszerűsége miatt rendkívül kedvelt és talán a legszélesebb körben is alkalmazható, azonban a hosszú távú felhasználására csak ritkán kerül sor, mert sajnos a legtöbb integrációs kérdésben a kommunikációhoz szükséges idő kulcsfontosságú kérdés, a fájl átvitel alapú integráció egy lassú megoldás. Nem maga az input-output folyamat problémás, hanem a fájlrendszerbeli módosítások egyes mappák periodikus megfigyelése. Továbbá a megoldás másik nagy kihívása, hogy ezeket az adatfájlokat mikor hozzuk létre / generáljuk le, és mikor dolgozzuk fel.

Az esetleges párhuzamos működés inkonzisztens adatokat eredményezhet, vagy az operációs rendszer tiltja a hozzáférést, párhuzamos működés nélkül viszont a teljes alkalmazás várakozik a művelet elvégzéséig, vagyis nagy terhelésű rendszerekben a használata nem hatékony, nehezen felderíthető lassulásokat is eredményezhet.

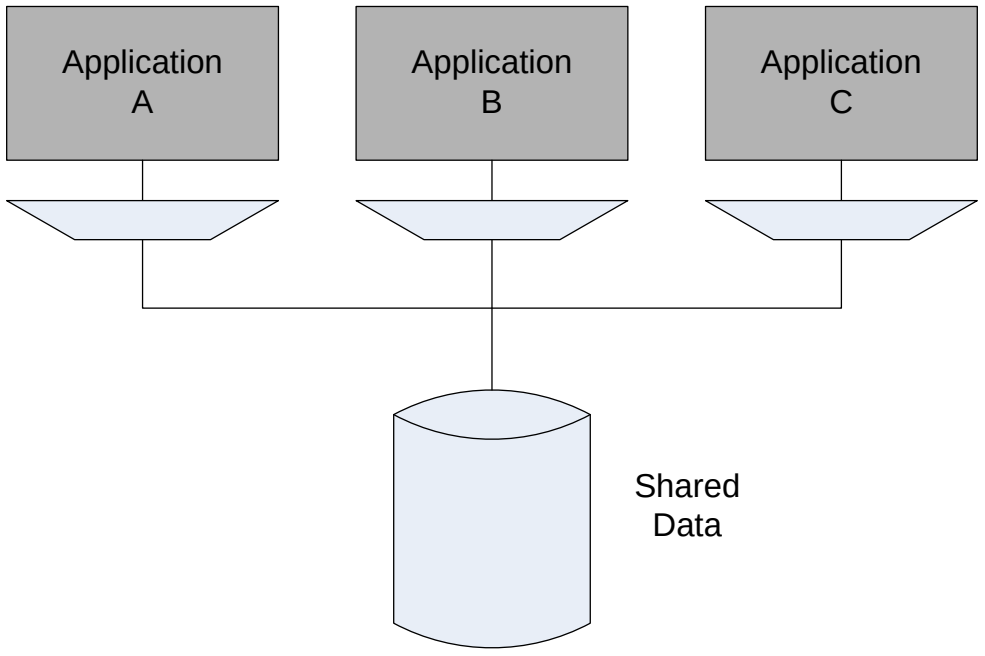
Összegezve: a fájl megosztás egyszerű megoldás, de éppen az egyszerűsége miatt csak kis mértékben skálázható, rugalmatlan eszköz.

Osztott Adatbázis

A fájl megosztásnál is kézenfekvőbb adatmegosztási eljárás az adatbázis kezelő rendszerek párhuzamos használata.

A működése a következő: létrehozunk egy közös adatbázist, amiben a rendszert alkotó minden alkalmazás adatokat kereshet, menthet, vagy módosíthat. Bár a megoldás triviálisnak tűnik, mégis sok nem várt problémát eredményezhet. A legnagyobb előnye, de ezzel egyben a legnagyobb hátránya is - a fájl megosztáshoz hasonlóan, -, hogy közvetlen adatkapcsolatot teremt az alkalmazások között. Ebben az esetben is kihívást jelent a változások figyelése, nyomon követése, sok komponens esetén az egyes elemek folyamatosan lekérdezik az adat változásokat, a módszer nem tudja értesíteni az egyes komponenseket adatmódosításkor.

Azon túl, hogy a módszer az alkalmazások egymástól való függőségét növeli, a teljes rendszert centralizálja, sőt akár egy komponens elégtelen teljesítménye, vagy hibás működése azonnal hatással van a teljes rendszerre, annak minden elemére.



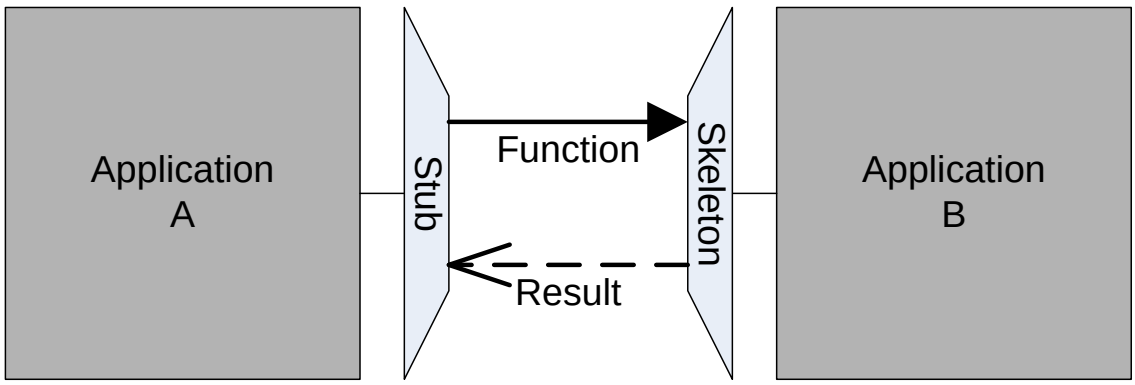
2. Integráció osztott adatbázison keresztül

Távoli eljáráshívás

A távoli eljáráshívás abból a szempontból jobb az egyszerű adatcserét megvalósító megoldásoknál, hogy itt lehetőség van az adatoktól, és az adatszerkezetektől független feladatok elvégzésére is. Például egy felhasználó címének módosítása több regisztrációs automatizmust is beindíthat a hátérben.

Az egységes, mindenki által elérhető adatok karbantartása rendkívül nehéz, mert a legapróbb adatszerkezet változtatáshoz is több alkalmazást kell egyszerre módosítani. A távoli eljárás elrejti a belső adatszerkezetet a többi alkalmazás elől, ezért a karbantarthatósága ebből a szempontból lényegesen jobb.

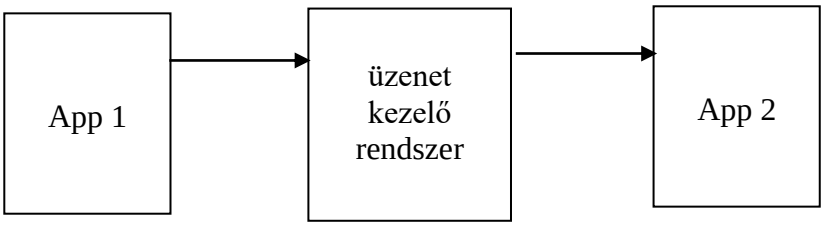
Az adatok köré szervezett távoli eljárások leegyszerűsítik a szemantikai eltérések kezelését is. Több interfészt hozhatunk létre egy adat elérésére, így megoldható, hogy az egyes alkalmazások különböző módon lássák az adatot. Bár sok előnye van, a komponenseket nagyon szorosan kapcsolja egymáshoz, minden összetevőnek ismernie kell, hogy a többi összetevő hol van és hogyan működik. Ez az egyes alkalmazások egyedi fejlődését nagyban akadályozhatja.



3. Távoli eljárás alapú integráció

Üzenetváltás

Megvalósíthatóság szempontjából korábban egy üzenetváltásokon alapuló rendszer igényelte a legtöbb fejlesztői munkát, de mégis széles körben elterjedt, mert ez a megoldás biztosítja a legnagyobb rugalmasságot, és magában foglalhatja az összes többi integrációs technológiát, így az integráció során nem sok kis, egyedi megoldást, hanem egy nagy üzenetváltáson alapuló rendszert fejleszthetünk ki.

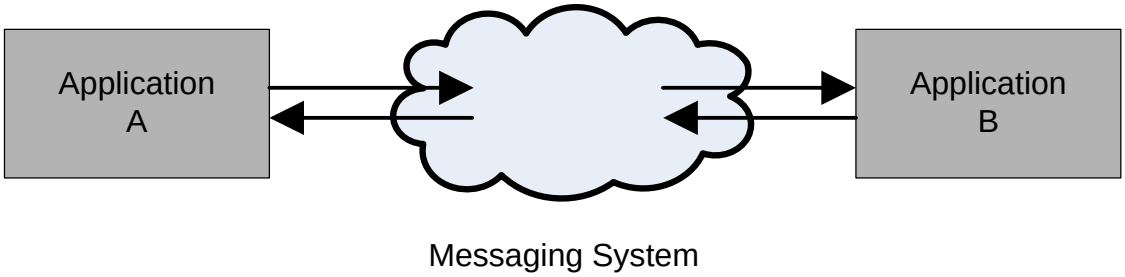


4 ábra: üzenet alapú integráció

Üzenetkezelő rendszerek

Az üzenetkezelő rendszerek (ÜKR), az alkalmazásokat lazán-kötötté teszik azáltal, hogy a kommunikáció aszinkron. Ez a szinkron működés ellentétje, ami azt jelenti, hogy egy üzenet elküldése után egyáltalán nem várakozunk a válaszra.

ÜKR alkalmazásával nő a rendszer stabilitása is, mivel a kommunikáló eszközöknek nem kell egyszerre rendelkezésre állnia (az üzenetküldő rendszer mindaddig újra küldi az üzenetet, amíg a címzett meg nem kapja azt). Mivel egy független üzenetküldő rendszer gondoskodik az adatok megérkezéséről, így az egyes alkalmazásoknak nem kell törődniük azzal, hogy hogyan jut el az üzenet a címzetthez, ami megkönnyíti az alkalmazások írását.



5. ábra üzenet kezelő rendszerek

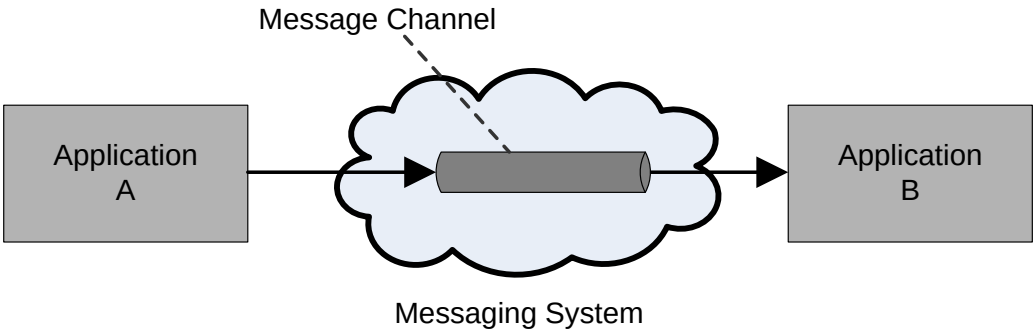
ÜKR főbb alkotóelemei

A következőkben a technológia legfontosabb eszközeit, kifejezéseit ismerheti meg, amely átfogó képet ad a teljes technológiáról, a teljesség igénye nélkül.

Csatornák / üzenetsorok

Az üzenetek küldésére üzenetküldő csatornák vagy üzenetsorok állnak rendelkezésre. Ezeket, az sorokat egyedileg hozhatjuk létre minimálisan konfigurálható névvel és típussal rendelkeznek. A feladatuk az, hogy adatokat közvetítsenek az ÜKR és az alkalmazások között.

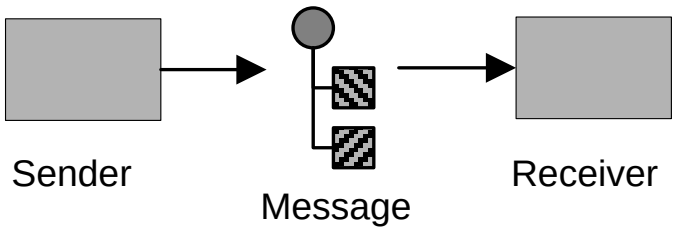
Általában minden frissen telepített ÜKR csatornák nélkül jön létre, az integrációt megvalósító csatornákat nekünk kell létrehoznunk, és konfigurálnunk.



6. ábra: üzenet csatorna

Üzenetek

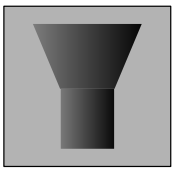
Atomi adat egység, ami elküldhető az üzenetsorokra. Ha egy alkalmazás adatot akar küldeni, akkor az adatot be kell csomagolja egy üzenetbe, és így küldheti el. Az egyes ÜKR implementációk ezt elrejtik a fejlesztők elől, elegendő a nyers adatokat kezelni.



7. ábra: üzenetek jelölése

Csövek és szűrők

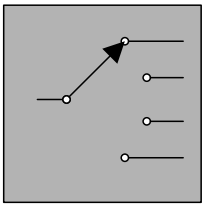
Bizonyos esetekben, ha nem megoldható a közvetlen adatküldés, mert például a küldő adatformátuma más, mint a fogadó állomásé, ekkor speciális csatornákra van szükség, amelyek átalakító, szűrő (esetenként monitorozó, felügyelő) eszközöket tartalmaznak.



Message
Filter

Útvonalválasztók

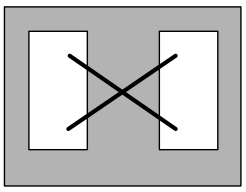
Azokban az esetekben, amikor a küldő nem tudja (vagy nem tudhatja), hogy pontosan kinek akar üzenetet küldeni útvonalválasztókat használunk. A leggyakoribb eset amikor erre szükség van az a terheléselosztást végző útvonalválasztó használata. Ebben az esetben csak a cél alkalmazás típusát tudja az üzenet küldője, de azt nem, hogy a lehetséges célállomások közül melyik a legkevésbé terhelt.



Message
Router

Átalakítók

Nagymértékben hasonlítanak a szűrőkre, azonban mégis külön csoportba soroljuk őket, mert nagyon gyakran használt eszközök. A feladatuk az, hogy az üzenetek formátumát megváltoztassák. Mivel az üzenetküldő rendszerben egy egyedi formátum az érvényes ezért minden alkalmazásnak, amelyik saját belső formátummal rendelkezik szüksége van két átalakítóra, hogy a ki és bemenő üzeneteket az egységes és a helyi formátum között konvertálja.

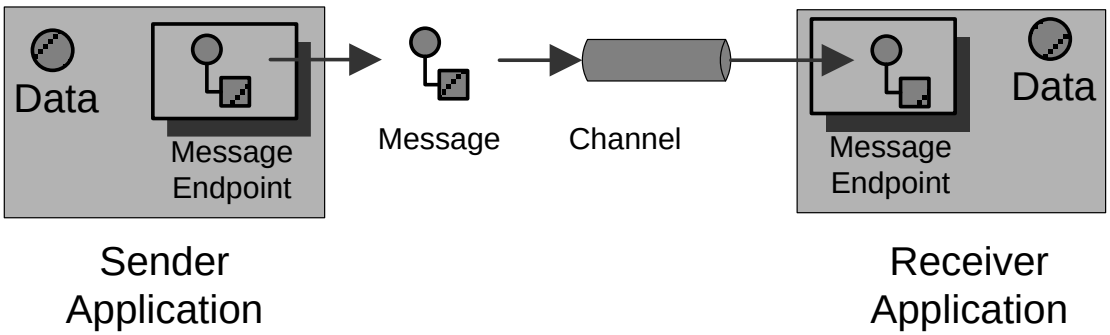


Message
Translator

Végpontok

Legfőképp az örökölt régi rendszerek integrációnál jelent problémát, hogy hogyan kössük össze az adott alkalmazást az üzenetküldő rendszerrel. Ilyenkor mindig egy speciális eszköz szükséges, ami lehet az alkalmazás egy új modulja, vagy akár önálló, új alkalmazás is, ami ezt a kommunikációt megvalósítja. Ilyen, az üzenetküldő rendszerrel közvetlenül kommunikáló eszközt nevezzük végpontnak.

Általában a megvalósítás szempontjából jól elkülönül attól az alkalmazástól, amelyiket az üzenetküldő rendszerbe kapcsolja, mégis a teljes rendszer szempontjából az alkalmazást és a benne kiépített végpontot tekinthetjük egy egységnek, mivel szorosan összefüggenek.



Üzenet csatornák

Az üzenet csatornák legfontosabb tulajdonsága, hogy a küldőnek nem kell (valójában nem is lehet) megneveznie a konkrét fogadó komponenseket, hanem pusztán egy üzenetsort választ, ahová az adatot beküldi. A sorokra/csatornákra való feliratkozó más komponensek viselkedésére a küldőknek nincs befolyása, nem is kérdezhető le, hogy valójában hová kerül az elküldött üzenete, és kik fogják megkapni.

Ez elsőre különleges megkötésnek tűnik, de mégis nagy rugalmasságot biztosít, mert így a rendszer működése közben például lecserélhetünk komponenseket. Természetesen ez a rugalmasság vissza is üthet: a küldő nem lehet biztos abban, hogy az üzenetét megkapja e valaha valaki. (a probléma kivédésére az ÜKR egyéb megoldásokat biztosít)

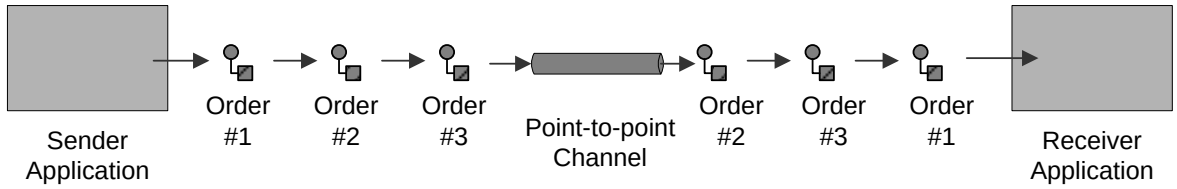
Ez adja az alapját annak az új szemléletnek, amit az üzenetküldő rendszer fejlesztésekor fel kell használnunk. Az üzenetek kézbesítése már nem az egyes alkalmazások feladata, hanem az üzenetküldő rendszeré. Van lehetőségünk monitorozni azt, hogy minden csatorna helyesen működik e (a küldés, a fogadás, és ha szükséges, akkor a válasz üzenet küldése megtörténik e)? Az egyes alkalmazásoknak már csak az üzenet tartalmával kell foglalkozniuk, továbbítás részleteit a rendszer megoldja.

Egy ilyen ÜKR beépítésének legsarkalatosabb pontja a csatornák hálózatának kiépítése. A következőkben az alap csatorna típusokat vesszük végig, mint a csatornahálózat építőköveit.

Pont-pont csatorna

A csatornák, természetüknél fogva (mivel önálló egységei a rendszernek), képesek lehetnek terhelés elosztásra, vagy monitorozásra is. A pont-pont csatornák - mint ahogyan a nevéből sejthető - egy küldőt kapcsolnak össze egy fogadóval. Ez a gyakorlatban azt jelenti, hogy minden a csatornába beküldött üzenet egy és csakis egy végponton fog kilépni. Ha több fogadó állomást állítunk be, a fejlettebb rendszerek nem csak véletlenszerűen, az egyik fogadónak juttatják el az üzenetet, hanem az egyes fogadók terheltségét figyelve, mindig a legkevésbé terheltnek adják a soron következő üzenetet.

Ez a megközelítés jól skálázhatóvá teszi a rendszert, mivel az egyes fogadók, lehetnek különböző gépeken is. Mivel a teljes rendszert csatornákra építve állítjuk össze, így a már kész rendszert megfigyelve, ott tudunk növelni a teljesítményen, ahol erre szükség van. Ez a lehetőség rendkívül nagy segítség lehet olyan nagy rendszereknél, ahol előre nehéz kiszámítani a pontos erőforrás igényeket.
Érdekesség, hogy az üzenetek sorrendje nem garantált, előfordulhat, hogy az egyes üzenetek „megelőzik" egymást, a később küldött fog előbb kilépni a csatornából. Ezt a problémát figyelembe kell venni a csatornahálózat tervezésekor.



8. ábra: pont-pont csatorna

Publikálás-feliratkozás csatorna

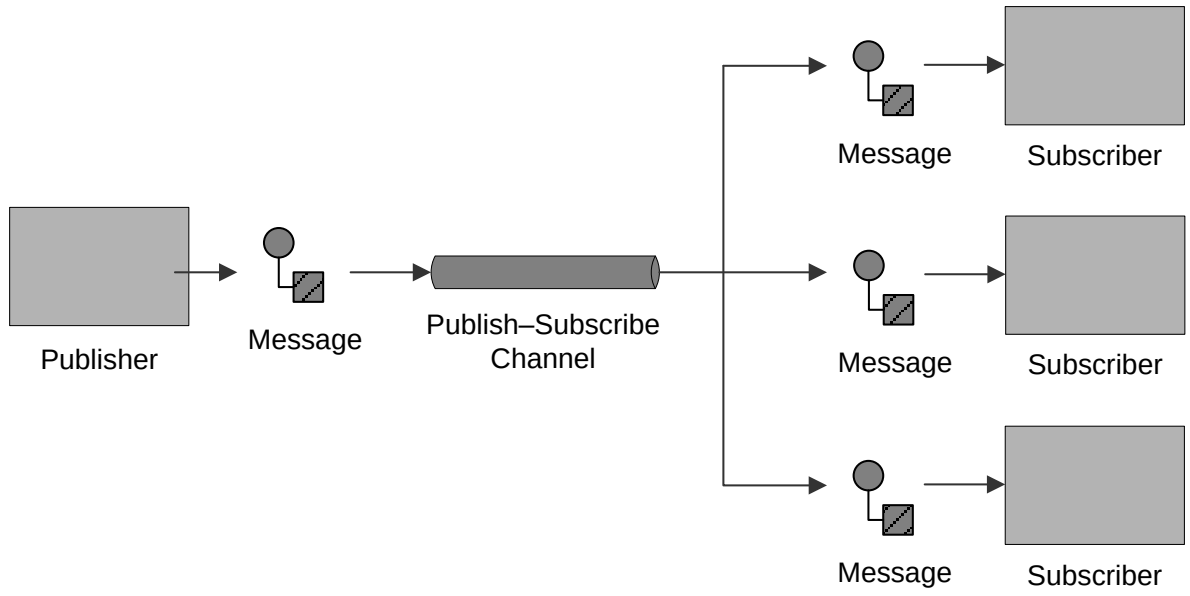
Mint azt az egyszerű pont-pont csatornánál láthattuk, lehetőségünk van arra, hogy egy csatornának több fogadója is legyen. A publikálás-feliratkozás elvű csatorna éppen ezt a lehetőséget aknázza ki. A működése a következő: amelyik alkalmazás regisztrálja magát (feliratkozik egy adott témájú üzenetre), küldhet üzenetet, amit minden regisztrált (feliratkozott) alkalmazás megkap.

Egy példa az ilyen jellegű csatornákra a felhasználó adatainak módosulását jelentő csatorna. Általában a felhasználó karbantarthatja az adatait egy Webalkalmazáson keresztül, de mobiltelefonról is módosíthatja az adatait. Mindkét esetben a változást egy alkalmazás kezeli és jelenti a többiek felé. Az egyik esetben a Web szerver, a másikban a felhasználói-kapcsolattartó központ szervere kell, hogy eljuttassa a rendszer többi részeibe az információt. Mivel a változást több egymástól független rendszernek is tudtára kell adni (pl. számlázó rendszer), ezért célszerű egy publikálás-feliratkozás elvű csatorna létrehozása és használata.

Ha egy alkalmazásnak sok más alkalmazást is érintő információja van, az publikál, amely alkalmazásokat az adott tárgyú üzenet érdekel, az feliratkozik rá, és figyeli az új üzeneteket.

Ez egy rendkívül hatékony hibakezelő/monitoring eszköz is lehet, mert az üzenet áramlást egy kiegészítő alkalmazással figyelni tudja anélkül, hogy a rendszer eredeti működését bármilyen módon megváltoztatnánk.

Leggyakrabban a rendszerben lezajló eseményekről, változásokról szóló jelentések közzétételére szokás ilyen elvű csatornákat használni. Nagyon gyakran, a rendszer nagy előnye itt is hátránnyá válhat: mivel ez a fajta broadcast üzenetküldés nagymértékben terheli a hálózatot, és bárki összeállíthat egy néhány soros programot, amivel lehallgathatja az üzeneteket, így titkos információk küldésére kódolatlan formában nem alkalmas.



9. ábra: publikálás-feliratkozás csatorna

Rögzített adattípusú csatorna

Az üzenetküldő rendszerben minden üzenet a rendszer által definiált típusú. Ez a típus azonban csak hordozza az üzenetet, mint egy bájtokból álló adathalmazt. Az üzenet fogadójának ennél több információra van szüksége az üzenet feldolgozásához.
Természetesen a legtöbb esetben meg tudjuk oldani, hogy a fogadó előre tudja, hogy milyen üzenetet kap éppen, vagy az üzenetbe csomagolhatunk jelölőket, amelyekkel jelezhetjük, hogy mi az aktuális tartalma az üzenetnek. Ez azonban ellentmond a rendszer alapelvével,

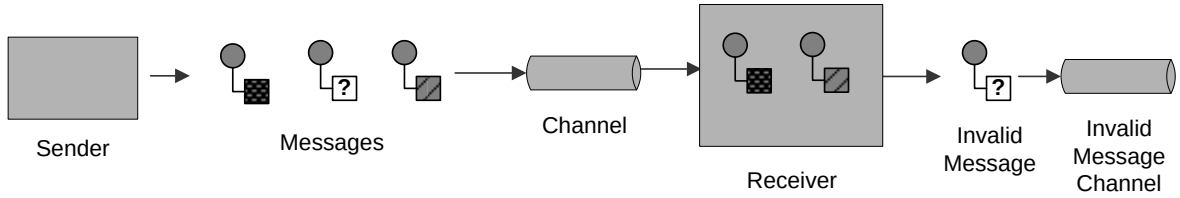
miszerint az egyes rendszereket nem szabad egymástól függővé tennünk, mert ez az egyes alkalmazások fejlesztését megnehezíti.
Ha a fogadó fél kell, hogy felismerje a neki küldött üzeneteket, akkor minden ilyen üzenet-formátum két alkalmazás között egyedileg jön létre, ami azt eredményezi egy nagy rendszerben, hogy egymáshoz nagyon hasonló üzenet létrehozását, és feldolgozását készítjük el. Ez nem csak a fejlesztés hatékonyságát rontja le, de a rendszer karbantartását is megnehezíti. Ezért vált elfogadott gyakorlattá, hogy minden csatornát, csakis egyféle típusú üzenet küldésére használunk fel. Ha szükséges többféle üzenetet is küldeni két alkalmazás között, akkor több párhuzamos csatornát hozunk léte. Így az üzenetek szintaktikai helyességét is sokkal könnyebb ellenőrizni, ráadásul, mivel a csatorna, és a hozzá tartozó ellenőrző és feldolgozó eszközök egy egységként jelennek meg a rendszerben, így a későbbi újra hasznosításuk is könnyen megvalósítható.

A rögzített adattípusú csatornák hátránya, hogy óriási mértékben megnövelhetik a csatornák darabszámát a rendszerben, ami magának az üzenetküldő rendszernek a túlterhelését eredményezheti. A probléma megoldására bevett gyakorlat, hogy üzenet elosztó eszközöket illesztünk a rendszerbe, amelyek egy kombinált adattípusú csatornáról veszi az üzeneteket és továbbítja más, rögzített adattípusú csatornák felé. Ezzel a megközelítéssel, a rendszer pókháló-szerű topológiáját csoportosítani lehet főgerincekre és álhálózatokra, amely már jobban skálázható, jobban felügyelhető megoldás.

Érvénytelen üzenet csatorna

Minden rendszer fejlesztésekor problémát jelent, hogy a működő rendszerben hibák is keletkeznek. Egy hiba bekövetkezése nem naplózható ki az egyes komponensek lokális fájlrendszerébe, mert nagy rendszer esetén ez a művelet akár több gépen, több száz fájlt is jelenthet. Mindegyiket áttekinteni csak azért, hogy megtudjuk, hogy hol keletkezett a hiba, nem hatékony megoldás. Ezért szokás hibajelentő csatornákat létrehozni, amelyek segítenek összegyűjteni a hibákat.
Az ilyen hibagyűjtő csatornatípus egyike az érvénytelen üzenet csatorna, amely kifejezetten az üzenetek feldolgozási hibáit hivatott gyűjteni. Ha egy alkalmazás nem tud feldolgozni egy üzenetet, mert az formailag hibás, vagy mert az üzenetben foglalt feladat teljesíthetetlen, akkor erről a problémáról jelentést kell tennie.

Az érvénytelen üzenetek csatornáján normál működés esetén nem lehet üzenet, azonban az alkalmazások rejtett hibái, vagy a fejlesztés során elkövetett hibás módosítások miatt megjelenhetnek üzenetek. Az ÜKR-ek nagyon gyakran előre gyártott hibaüzenet feldolgozó egységet is tartalmaznak, amely értesíti a rendszer üzemeltetőit a hibáról.



10. ábra: érvénytelen üzenetcsatorna

Halott levél csatorna

Az üzenetküldő rendszerek szintén beépítve tartalmazzák, elsősorban az üzenetküldő rendszer küldhet üzenetet rajta, a rendszer karbantartói, üzemeltetői számára.

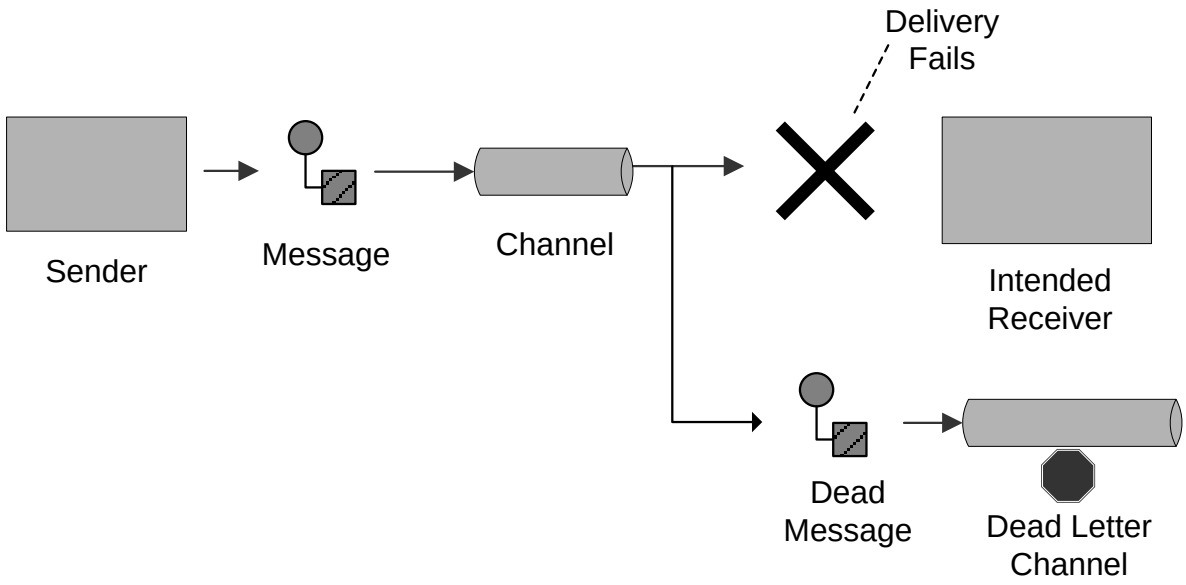
Rajzi jele általában:

11. ábra: halott levél csatorna

A halott levél vagy olyan formai hibát tartalmazó üzenet, ami miatt kézbesíthetetlen (például érvénytelen címzett), vagy a csatornát, amiben az egyébként érvényes üzenet haladt megszüntették, vagy az üzenet élettartama lejárt a kézbesítés előtt.

A halott levél és az érvénytelen üzenet közötti legfontosabb különbség az, hogy a halott levél valamiért nem kézbesült, ami akár még az üzenetküldő rendszer hibás működését is jelentheti, ellentétben az érvénytelen üzenettel, ami sikeresen megérkezett a címzetthez, de a címzett alkalmazás nem tudta feldolgozni azt (üzleti logikai oldalon).

Ezek az üzenetek mindig komoly hibára utalnak a rendszer működésében, ezért minden üzenetküldő rendszer nagy hangsúlyt fektet a kezelésükre.



12. ábra: halott levél csatorna működése

Garantáltan kézbesítő csatorna

Az üzenetküldő rendszer képes kezelni az egyes alkalmazások időleges működési hibáit, a helyes működés visszaállásakor továbbítják az üzenetet. Ez a rendszer biztonságát növeli, de a száz szálakés biztonságot önmagában nem garantálja, mivel az üzenetküldő rendszerben is keletkezhet hiba, illetve egy bizonyos idő után, mivel az üzenetküldő rendszer kapacitása nem végtelen, kénytelen a régi leveleket kitörölni.
Azokban az esetekben, amikor nagyon nagy biztonsággal kell eljuttatni az üzeneteket a címzettekhez, ügynevezett garantáltan kézbesítő csatornát kell használnunk. Ez a rendszer a lokális háttértárolón tárolja az üzeneteket, a küldő visszaigazolást kap a letárolásról a küldési folyamat részeként, vagyis a sikeresen elküldött levelek nem veszhetnek el, még áramszünet esetén sem.

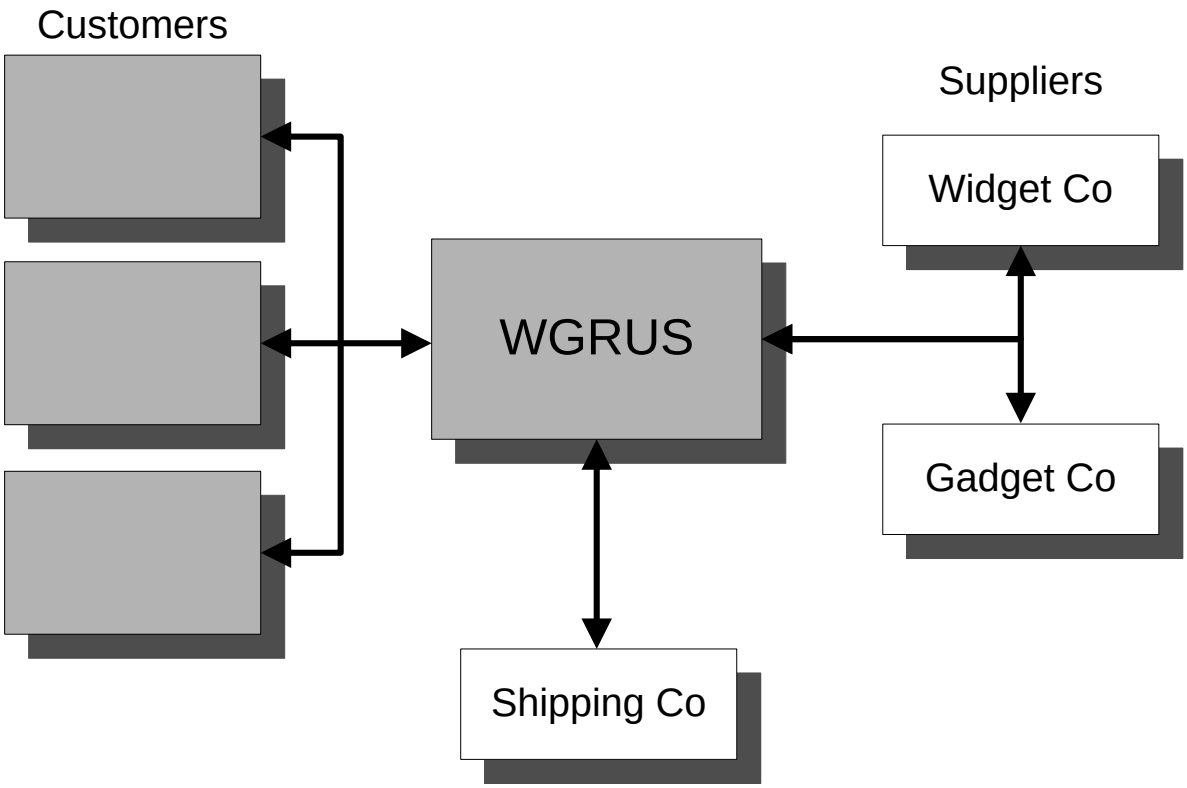
Az üzenet küldő rendszer a letárolt adatot a címzett rendelkezése állásakor átmozgatja a címzett gép háttértárolójára, majd a címzett innen olvassa azt fel. Ez az adatáramlás lassú, csak még nagy adatmennyiségek esetén sem túl hatékony, ezért csak ott érdemes használni, ahol a biztonság a legfontosabb szempont.

A teljesítményromlás mellett, további problémát okozhat a helyi meghajtók túlterhelése is. Ez ellen úgy védekezhetünk legegyszerűbben, hogy az adatokat egy adatbázis kezelőben tároljuk le, majd értesítést küldünk a címzettnek az új információ helyéről. Ez tovább ronthatja a hatékonyságot, de elkerülhetővé válik a helyi gépek állandó karbantartása.

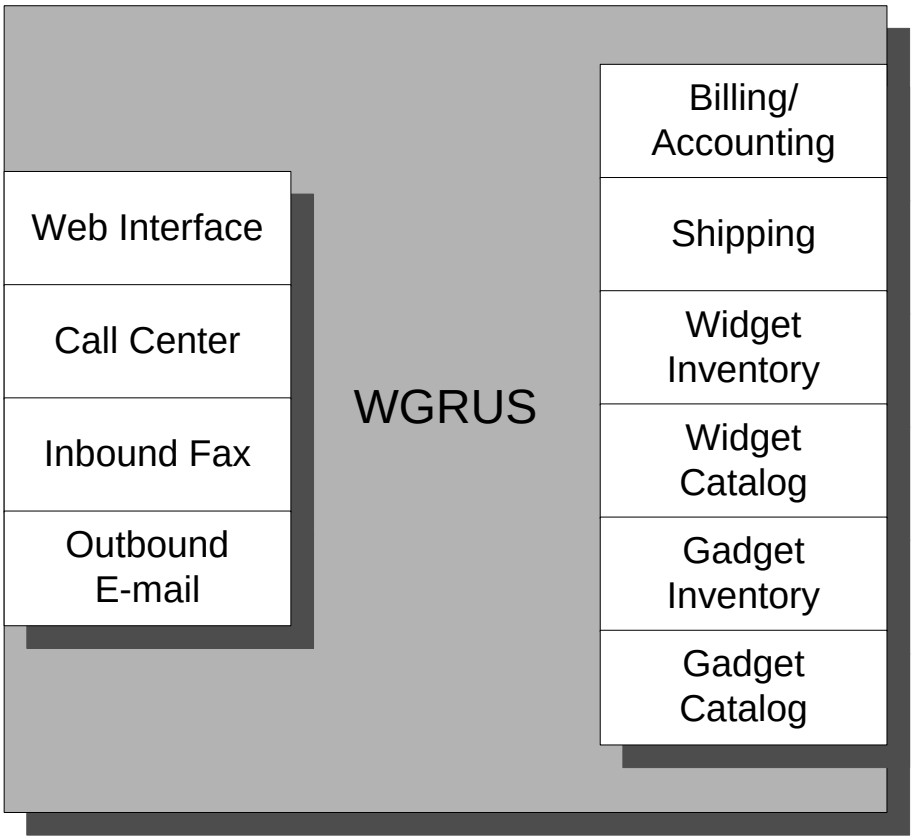
A WGRUS példa-vállalat

A WGRUS vállalat egy kereskedelmi vállalat, kezdetben egy beszállító termékeit forgalmazta, de a későbbiekben ezt bővítette egy másik beszállító termékeivel. A két beszállító „Widget” –eket és „Gadget” eket gyárt, innen a példavállalat neve:
Widgets & Gadgets'R Us (WGRUS).

A vállalat főbb kapcsolatai a következő 19-es ábrán látható:



A vállalat belső szervezése a következő ábrán látható:



Ábra 13

A vállalat kapcsolattartó egységei:

- Egy Internetes portál, és a mögötte lévő informatikai osztály,
- Egy telefonközpont, amely egy készen vásárol szoftver segítségével adminisztrálja a bejövő kéréseket.
- Fax rendszer. A faxon érkező kéréseket részben kézzel, részben egy házilag épített programmal veszik fel az alkalmazottak.
- A vállalat az ügyfelei felé e-mailben kommunikál.

A vállalat belső osztályai:

- Számlázás, egyenleg-kezelés
- Szállítmányozás
- Widget raktár (hagyományos okokból, a két beszállító kezelése egymástól független)
- Widget katalógus
- Gadget raktár
- Gadget katalógus

Elvárt eredmények

A vállalat elvárásai a kialakítandó rendszer felé:

- A megrendelések mindhárom irányból (Internet, telefon, fax) érkezhessenek be,
- A vevők egységes felületen keresztül láthassák a teljes kínálatot, függetlenül attól, hogy melyik beszállítótól érkezik,
- Az ügyfelek az adataikat a Webes interfészen keresztül bármikor módosíthassák,
- A Webes interfészen a megrendeléseik állapotát nyomon követhessék,
- A vállalat belső működését ne veszélyeztethesse a többi vállalat szoftvere.
- Az integráció megvalósítása ne veszélyeztesse a meglévő rendszerek stabilitását,
- Az integráció megvalósításakor a lehető legolcsóbb, és leggyorsabb megoldásokat kell választani.

A megvalósítás

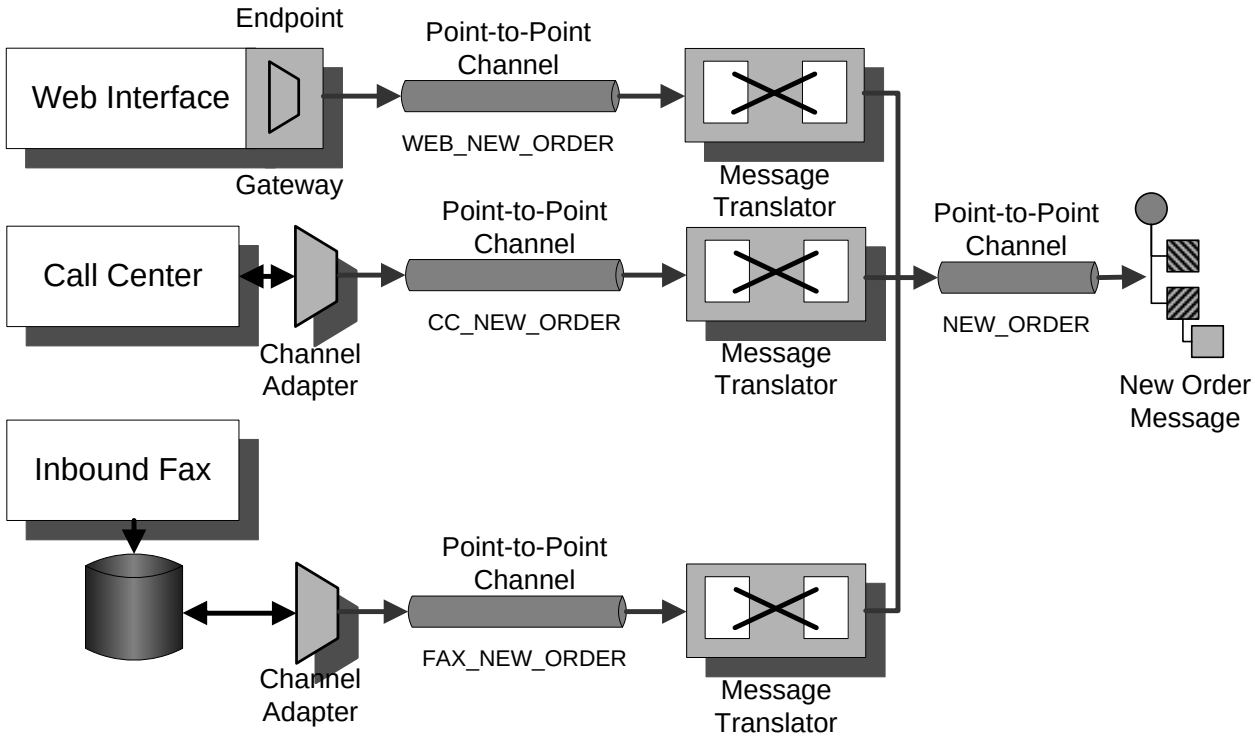
A megvalósítás első lépése a megrendelések fogadása, a megrendelések egységes alakra hozása, a forrástól függetlenül. Ennek elérése érdekében mindhárom rendelés felvevő eszközt alkalmassá kell tenni egy közös formátumú megrendelés objektum létrehozására.

Mivel a megrendelések fogadása egy aszinkron művelet, és sok rendszert összekapcsol, ezért üzenetküldő rendszer kialakítását választjuk.

Mivel a Web interfész saját fejlesztésű alkalmazás, ezért ezt bővíthetjük egy üzenetküldő rendszer végponttal, amely így közvetlenül képes kommunikálni a rendszerre. A telefonközpont szoftveréhez egy modult tudunk fejleszteni, amely kommunikál a rendszerrel, míg a fax üzenetek kezelésénél csak az adatok adatbázisból történő „kilopásával” tudunk kapcsolódni a rendszerhez.

Mivel mindhárom eszköznek saját adatformátuma van, ezért ezeket egységes alakra kell hozni, mivel a rendszer többi része felé transzparensé szeretnénk tenni az üzenet forrását. Ez azt eredményezi, hogy átalakítókat kell használnunk, minden üzenet-típusra.

A következő (21-es) ábra a megrendelések fogadásának elvi rajzát mutatja:



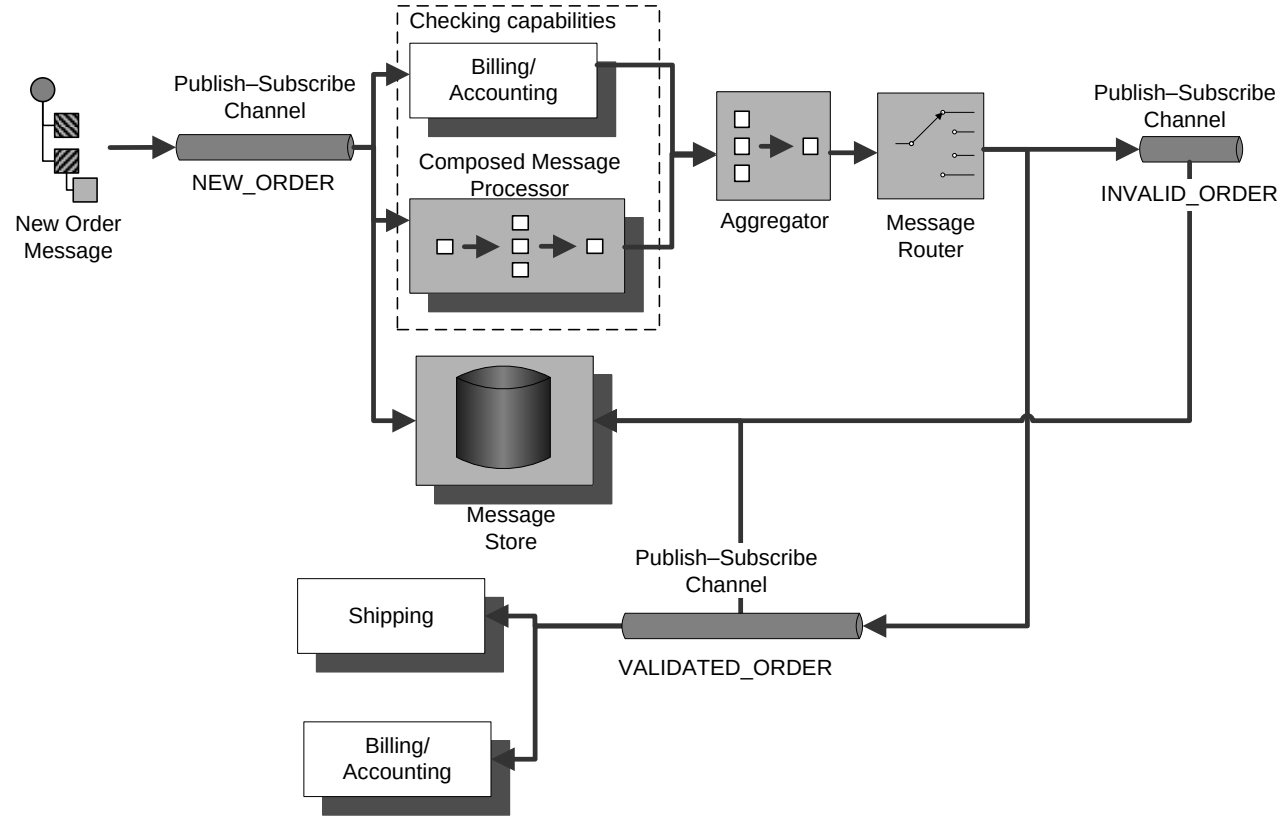
Mivel a megrendelések fogadása egy aszinkron művelet, és sok rendszert összekapcsol, ezért üzenetküldő rendszer kialakítását választjuk. A megvalósítás során az egyes fogadó egységeket az ábrán látható módon alkalmassá tesszük az üzenetküldő rendszerrel való kommunikációra, majd az egyes üzenet típusokat egy-egy fordító segítségével közös formátumra hozzuk. Ez a közös formátumú megrendelés a NEW_ORDER nevű csatornán lesz elérhető a rendszer többi eszközei számára.

A bejövő kérések kiszolgálása

A megrendelések teljesítése négy fő műveletet jelent:

- A megrendelő számlaegyenlegének ellenőrzése (képes e kifizetni a megrendelését),
- A raktárakban megvan e minden megrendelt termék (képesek vagyunk-e szállítani),
- A szállítást meg kell rendelni a szállítmányozó cégtől,
- Számla kiküldése a megrendelőnek.

Mivel az ellenőrzések közül bármelyik sikertelensége meg kell, hogy akadályozza a további műveleteket, ezért két fő csoportban hajtjuk végre a feladatokat. Ha minden ellenőrzés sikeres volt, akkor szállítunk, és számlázunk. Az alábbi 22-es ábra részletesen bemutatja a folyamatot.



Érdemes megjegyezni, hogy a megrendelés több termékre vonatkozik, melyek a vállalat két raktárának tartalmából tevődik össze. Mivel a raktáraknak külön kezelő szoftvere van, ezért a

bejövő összetett megrendelést szét kell szedni elemeire, majd szűrők segítségével az egyes raktárak felé irányítani őket. Ezek után az egyedi, ellenőrzött megrendelési kéréseket egy aggregátor segítségével újra össze kell főzni egy elemmé. Az ilyen „szétszedés-művelet-összeillesztés" folyamatok nagyon gyakoriak, és egységesen Composed Message Processor-nak nevezzük őket.

A karbantartási műveletek

A rendszer karbantartása alatt minden olyan műveletet értünk, ami a rendszer folyamatos üzemének fenntartását biztosítja. Így ide tartozik, a legnehezebben kezelhető karbantartás is: a felhasználó adatainak kezelése. Természetesen ezt a folyamatot is automatizálni kell, erre egy lehetséges példa a felhasználó cím módosítási kéréseit kezeli.
A címmódosítás kétféle cím módosítását jelentheti, a számlázási cím megváltozását vagy a szállítási cím módosítását. Természetesen itt is meg kell hagyni a lehetőséget a felhasználónak, hogy egyszerre mindkét címet módosíthassa, és itt is egységes felületet kell biztosítani.
Ennek megvalósítását mutatja az alábbi ábra: