ÚJ UTAK A SZOFTVERFEJLESZTÉSBEN

Építs magadnak szoftvert!

Venni vagy fejleszteni? A legtöbb vállalatnál még mindig így merül fel a kérdés, ha arról van szó, hogy új szoftvermegoldásra lenne szükség. A dilemma kézenfekvőnek látszik, de a legtöbb esetben elavult feltételezéseken alapul, és nem veszi figyelembe az egyre inkább digitalizálódó vállalatok valós igényeit.

FORRÁS: 123RF.COM

Egy tényleg digitalizáltan működő nagyvállalat esetében a „megveszem készen és használom” opció gyakorlatilag nem létezik, az „igényeim szerint nulláról kifejlesztem” megközelítés pedig ritkán célravezető. A valódi kérdés sokkal inkább így hangzik: szabjunk testre egy előre csomagolt, kész alkalmazást, vagy komponensekből állítsuk össze a nekünk szükséges egyedi megoldást? Esetleg alkalmazzuk a kettő valamilyen kombinációját?

Agilis és több csatornás a fejlesztés

Több, a szoftverfejlesztésben mélyreható változásokat előidéző trend is hozzájárul ahhoz, hogy átalakul a kérdésfeltevés. Először is, jól megfigyelhető a fejlesztési metodológiák változása. „Ma már alig van olyan ügyfelünk, amelyik vízesés modellben szeretne egy fejlesztési projektet végigvinni. Emögött többnyire az áll, hogy egy nagyvállalati rendszer esetében egyszerűen túl sokáig tart a teljes követelményspecifikáció elkészítése – mire előáll a teljes dokumentáció, az élet túl is lépett rajta. Az üzlet nem engedheti meg magának, hogy mindent a legapróbb részletekig előre eltervezzen; ehelyett mindenki igyekszik agilis fejlesztési módszertanokat bevezetni”, mondja *Tárkányi Ferenc,* az EPAM presales csapatának vezetője.

Az agilis fejlesztésnek nem akadálya, ha az ügyfél és a fejlesztő képviselői nem ugyanabban a szobában ülnek vagy akár nem ugyanannak a cégnek az alkalmazottai, teszi hozzá. Különféle módszertanok léteznek (például a Scaled Agile Framework, a SAFe), amelyek megmutatják, hogyan lehet az agilis működést az egyes csapatok szintjéről akár a teljes vállalatra is kiterjeszteni, miközben külső feleket (adott esetben fejlesztő partnereket) is bevonnak ebbe.

A második lényeges változás, hogy immár a szoftverfejlesztők ügyfeleinek ügyfelei is digitális csatornákat használnak, így ők is a vállalati szoftver felhasználóivá váltak. „Ennek megfelelően a vállalati szoftvereket fel kell készíteni az összes csatornára kiterjedő, azaz omnichannel működésre. És ez nem csak annyit jelent, hogy el kell készíteni a szoftver webes vagy mobil verzióját, hanem lehetővé kell tenni, hogy a végfelhasználó szabadon váltson az egyes csatornák között. Egy kereskedelmi szoftvernek például kezelnie kell azt az esetet, amikor az ügyfél a webshopban rendel, de az áruházban veszi át a terméket”, részletezi a második nagy trendet Tárkányi Ferenc.

A harmadik lényeges trend, hogy a vállalatok egyre nagyobb figyelmet fordítanak a munkavállalói élményre, a szoftverek kényelmes, magától értetődő használatára. Erre a Covid-járvány rátett egy lapáttal, és olyan funkciókat – távoli elérést, digitális aláírást – tett kötelezővé, amelyek korábban nem voltak jellemzőek.

FORRÁS: ITB

TÁRKÁNYI FERENC, EPAM

> A low-code/no-code platformokon egy üzleti felhasználó is – némi fejlesztői affinitással és rendszerszemlélettel, alapszintű programozói előképzettséggel – létrehozhat alkalmazásokat, kisebb-nagyobb rendszereket

A négybetűs követelmény: MACH

Technológiai tekintetben is egészen új igényeket támasztanak a vállalati szoftverekkel szemben az ügyfelek, folytatja Tárkányi Ferenc. Egyre inkább elvárás egy vállalati szoftvertől, legyen az dobozos vagy egyedileg fejlesztett, hogy megfeleljen a MACH-mintának, ahol minden egyes betű egy-egy technológiai követelményt jelent. Vagyis a szoftver legyen mikroszerviz alapú (M); rendelkezzen gazdag API-gyűjteménnyel (A); legyen felhő natív (C mint cloud); és mind fontosabb a headless jelleg is (H). De mit is takarnak ezek a kifejezések?

A mikroszerviz alapokra helyezett szoftver nem egyetlen nagy, monolitikus tömbben valósítja meg az üzleti folyamatokat és funkciókat. Ezeket igyekeznek minél kisebb funkcionális egységekre vágni, majd ezekből az egységekből összeállítani a rendszert. Amikor egy-egy (rész)funkción módosítani kell, vagy felmerül a továbbfejlesztés igénye, elegendő csak az adott mikroszervizhez hozzányúlni, a környezete, a nagy rendszer változatlan maradhat.

Az API-k, vagyis a programozási interfészek beépítése azért fontos, hogy a kész rendszer funkcionalitása könnyen hozzáférhető legyen más,

külső szoftverek számára. A vállalatok mindinkább kiterjedt ökoszisztémákban működnek, ahol alapelvárássá válik (például) a készletinformációk megosztása, az automatikus megrendelés lehetősége, és így tovább. A felhő natív elvárást nem kell különösebben magyarázni. Érdemes viszont kitérni a „headless" követelményre. Ez igazából annyit jelent, hogy egy nagyvállalati szoftver esetében elkülönülten kezelik a front-endet (a felhasználói felületet) és a back-endet, a háttérben megbúvó tranzakciós komponenseket. A Forrester egy 2021-es tanulmánya megállapítja, hogy a digitalizált korban a szoftver nem egyszerűen használati eszköz, hanem illeszkedik a céges brandhez, kifejezi a vállalkozás által képviselt értékeket, mutatja annak kreativitását. Ezt viszont nem lehet megtenni egyenszoftverekkel. Ha a cég meg akarja különböztetni magát a vetélytársaitól, a felhasználók felé kínált appjainak, online felületeinek is egyedinek kell lennie, legyen szó e-kereskedelemről vagy e-bankingról. A front-endet ezért számos esetben külön fejlesztik ki, és ültetik rá a viszonylag szabványos szoftverre – utóbbival kapcsolatban viszont így elvárássá válik, hogy szabadon lehessen hozzá kezelőfelületet készíteni, magyarázza a negyedik követelményt Tárkányi Ferenc.

Kódolás nélkül is lehet alkalmazást írni

Más technológiai fejlődések is elősegítik a szoftverfejlesztés új útjainak feltárását. Az utóbbi években például óriási fejlődésen mentek át az úgynevezett low-code/no-code platformok. Ezek az ügyfél és a külső fejlesztőpartner közötti együttműködést is gyökeresen megreformálhatják, de igazi előnyük a belső erőforrásokkal végzett fejlesztésekben mutatkozik meg, mondja *Basa Richárd,* az Oriana International vezérigazgatója. Egy ilyen platform fő jellegzetessége ugyanis az, hogy azon fejlesztői affinitással és rendszerszemlélettel rendelkező üzleti felhasználó akár alap programozói előképzettséggel is létrehozhat olyan alkalmazásokat, kisebb-nagyobb rendszereket, amelyekből akár 30-40-re is szüksége lehet évente már egy közepes vállalatnak is. Mondjuk a HR szeretné digitalizálni a dolgozók kiléptetésének folyamatát. Nem kell külső fejlesztőt keresnie, szállítót kiválasztania – az igényét a folyamat leírásával és a követelményspecifikációval együtt átadja a házon belül kialakított kompetenciaközpontnak, amelyik a low-code platformot használja. Ebben a központban nem programozók dolgoznak, hanem üzleti elemzők (business analystek) – ők a felhasználók közül kerülnek ki, csak éppen több affinitásuk van az informatika iránt, jobban átlátják a folyamatokat és megtanulták a platform kezelését.

Az üzleti elemzők a megkapott igények alapján nem a nulláról állnak neki a rendszer kialakításának, hanem kihasználják a low-code/no-code platform képességeit. A grafikus felületeken minimális kódolás mellett létrehozzák a digitalizált folyamatot, definiálják hozzá a szabályokat, az űrlapokat, meghatározzák, milyen adatokra és milyen forrásokból van szükség, és így tovább. Ebből villámgyorsan előáll egy olyan működő modell, amit az adott üzleti terület kipróbálhat, letesztelhet, és ha nem felel meg az igényeinek, módosíttathat. Az esetleges későbbi, kisebb módosításokat, szükségessé vált testreszabásokat pedig a végfelhasználók is el tudják végezni a no-code eszközökkel.

FORRÁS: ITB

BASA RICHÁRD, ORIANA INTERNATIONAL

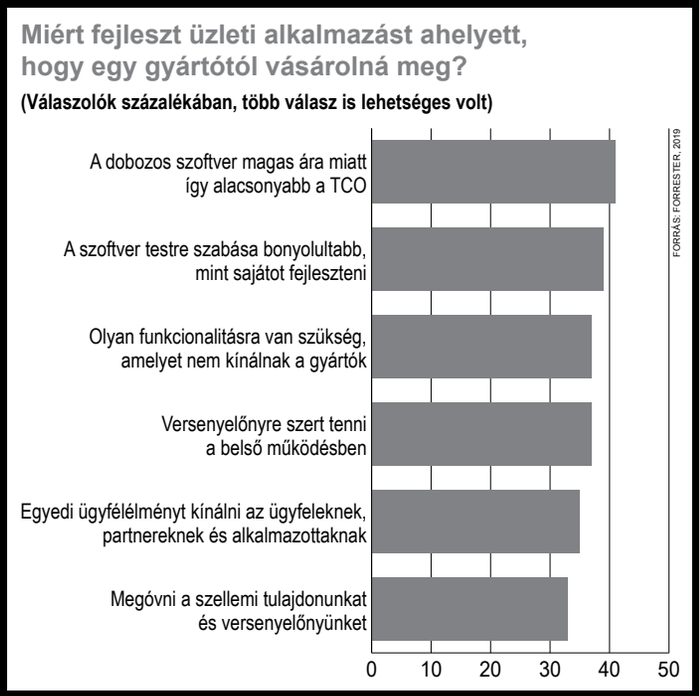
Már régen vártak erre a nagyvállalatok

A platform az integrációs képességeket is biztosítja. Szükség van iktatásra vagy elektronikus aláírásra az alkalmazásban? Az egyik lehetőség, hogy csak ilyenkor kell a programozók segítségét igénybe venni, akik megteremtik az összeköttetést az új alkalmazás és a háttérszolgáltatás között. Számos esetben azonban ezek a funkciók már kész formában rendelkezésre állnak magában a platformban is. Ilyenkor az üzleti elemzőnek nincs más dolga, mint hogy a szükséges funkciót behúzza a folyamat megfelelő lépéséhez, és onnantól fogva az alkalmazásba integrálva lesz az iktatás vagy az e-aláírás.

„A low-code/no-code platformok révén egy vállalat sokkal tudatosabban menedzselheti fejlesztési igényeit. A kompetencia-központ kapacitása és az üzleti igények ismeretében felállíthatja a prioritásokat, ütemezheti a kisebb fejlesztéseket, amelyeket a belső csapat agilis módon végez-

Fejlesszen a szoftver!

Nem meglepő módon a szoftverfejlesztésben is szerephez juthat a mesterséges intelligencia. Ennek egyik ága, hogy gépi tanulással automatikus döntéshozatali képességeket építsenek be a low-code/no-code platformokba, említi Basa Richárd. Ennél is izgalmasabbnak ígérkezik az a lehetőség, hogy a szoftver fejlessze a szoftvert. „Ha már van számos komponensem, amelyek a legkülönfélébb üzleti problémákra adnak választ, akkor nem látom akadályát olyan rendszer kialakításának, ahol az üzleti definíciók megadása után az MI felépíti a prototípus vázát, kiválogatja a szükséges alkalmazásblokkokat és az egyéb szükséges elemeket. Ezt a prototípust az üzleti elemző validálja, és már csak testre kell szabni, miáltal még gyorsabb lehet az üzleti problémák megoldása", vázolja a folyamatot az Oriana ügyvezetője.



het, egy-két hónapon belül előállva az eredménnyel, miközben végig szoros marad a kapcsolat az üzleti terület és a szoftver előállítói között. Ugyanakkor a platformok mai generációja már tökéletesen alkalmas arra, hogy nagyvállalati, B2B-szoftvereket írjanak benne, hiszen megbízhatóságban, biztonságban, méretezhetőségben, menedzselhetőségben megfelel a nagyvállalati követelményeknek", foglalja össze az előnyöket Basa Richárd.

Még egyszerűbb a testreszabás

A technológiai és módszertani fejlődések hatására átalakultak és kibővültek a vállalatok lehetőségei, amikor új szoftvereket szeretnének használatba venni. Az egyik fő csapásirány a testreszabás lett, amely azonban nem azt a fajta „customisationt" jelent, mint évekkel ezelőtt. A legnagyobb szoftvergyártók is felismerték, hogy nem tartható az az üzleti modell, amelyik a méregdrága licencdíj mellett még hasonló nagyságrendű bevezetési és testreszabási költségeket terhel az ügyfélre.

Ennek megfelelően egyrészt modulárisabbá tették megoldásaikat, hogy az ügyfeleknek ne kelljen olyan funkciók után is fizetniük, amelyekre semmi szükségük nincs. Másrészt ki is nyitották ezeket a rendszereket a külvilág felé. Fejlesztőeszközöket, keretrendszereket, API-készleteket kínálnak partnereiknek, amelyek úgy tudnak kiegészítő szoftvereket gyártani, hogy közben az alaprendszer megbízhatósága nem sérül – ezt a szerepet játssza például a Microsoft technológiai környezetében a PowerApps a Dynamics szoftvercsomag mellett. Vagy teljes ökoszisztémát építenek ki rendszereik köré: a Salesforce-nak például van saját alkalmazásboltja, ahonnan az ügyfelek könnyedén telepíthetnek az alaprendszerrel megbízhatóan együttműködő kiegészítőket. A headless megoldások pedig teljesen szabad kezet adnak az ügyfélnek a saját front-end kialakításához.

De a központi, back-office vállalati rendszerek mellé egyre gyakrabban vásárolnak low-code/no-code platformokat az ügyfelek, sőt, sokszor a gyártók ajánlják ezeket, teszi hozzá Basa Richárd. Egy ilyen platform jól integrálható a központi rendszerhez, onnantól kezdve pedig könnyedén kialakíthatók és bevezethetőek a napi munkavégzés hatékonyságát növelő appok anélkül, hogy hozzá kellene nyúlni a core alkalmazáshoz. „Egy ERP-rendszert legfeljebb évente kétszer frissítenek egy vállalatnál, úgy pedig nem lehet lekövetni a gyors üzleti változásokat. De ha könnyen fejleszthetünk mellé jól integrálható saját alkalmazásokat, máris nagyot léptünk előre a rugalmasan működő IT felé", mondja az Oriana ügyvezetője.

Kell az üzleti tudás a technológia mellé

A szoftverkészítés másik fő csapásiránya az előre elkészített elemekből való építkezés, a compose lesz. „Számos ügyfelünk keres meg minket azzal, hogy nem talált megfelelő kész szoftvert valamely üzleti igényére, ezért készítsünk neki egyet. Az a jó ebben, hogy ilyenkor sem nulláról kell indulni. Számtalan nyílt forráskódú szoftverelem, felhőszolgáltatás, SaaS-API és más komponens érhető el, amelyeket összeépítve pontosan olyan megoldás készíthető, amilyet az ügyfél szeretne", vázolja az alternatívát Tárkányi Ferenc. Ilyenkor persze nagy figyelmet kell szentelni az architekturális alapokra, a megbízhatóságra, a méretezhetőségre, a biztonságra, hiszen ezek nem feltétlenül következnek az egyes komponensekből.

A low-code/no-code platformok ezen a téren is komoly segítséget nyújthatnak. A low-code komponenssel akár a platform fejlesztője, akár annak használója létrehozhat alkalmazássablonokat, kész appokat vagy egyéb komponenseket, amelyeket később akárhányszor felhasználhat. Ha pedig már minden szükséges komponens készen áll, szó szerint „össze lehet kattintgatni" egy-egy szűkebb üzleti funkciót támogató alkalmazást.

A jövő a domainspecifikus low-code/no-code platformoké, állítja határozottan Basa Richárd. Az ügyfelek azokat a platformokat fogják előnyben részesíteni, amelyek nemcsak technológiát, hanem üzleti tudást is kínálnak nekik, alkalmazások vagy sablonok formájában. „Egy beszerzési rendszert például gyakorlatilag bármelyik platformon meg lehet csinálni. De ha az egyik már készen tartalmazza, ráadásul ott vannak mellette azok a no-coding eszközök, amelyekkel könnyen testre lehet szabni, akkor az vonzóbb lesz az ügyfél számára", mondja. A specializáció miatt azt is valószínűsítik a piackutató cégek, hogy egy-egy ügyfél három-négy low-code/no-code platformot is használ majd, és az egyes alkalmazásokat abban készíti el, amelyik éppen a legkényelmesebb, legalkalmasabb az adott feladatra.

Az új modellben kritikussá válik a folyamatos és szoros együttműködés az üzleti ügyfél és fejlesztő partner között a szoftverfejlesztés sikere szempontjából

FORRÁS: 123RF.COM

Csak közösen lehet

Az új módszerek viszont az ügyfelektől és kisebb részben a fejlesztő partnerektől is új gondolkodásmódot és hozzáállást igényelnek. Az egyik legfontosabb változás talán az, hogy a fejlesztés folyamatos és mély bevonódást igényel a későbbi üzleti felhasználók részéről. „Az nem működik, hogy az ügyfél leadja az igényét, majd hátradől, és várja a kész produktumot. A folyamatos együttműködés ügyfél és partner között kritikussá válik a szoftverfejlesztés sikere szempontjából", figyelmeztet Tárkányi Ferenc.

Cserébe nem feltétlenül kell jó előre pontosan definiálnia a kért megoldás minden részletét. Az sem baj, ha nem tudja pontosan, mit akar – a jó fejlesztőpartner segít megalkotni a magas szintű víziót, a részletes követelményeket, a specifikációt pedig majd a fejlesztéssel párhuzamosan dolgozzák ki. A partner készít egy egyszerű prototípust, akár kattintható változatban, ahol az ügyfél már maga elé tudja képzelni a kész terméket, és el tudja dönteni, tetszik-e neki az irány. Mindez persze sokkal szorosabb együttműködést igényel, de a végeredmény is közelebb fog állni az elképzelésekhez és igényekhez.

A másik oldalról viszont a fejlesztőnek is fel kell készülnie az ilyen jellegű közös munkára. Meg kell értenie az ügyfél mégoly ködös elképzeléseit is, formába önteni a homályos víziókat. Ismernie kell a legkülönfélébb technológiákat és módszereket, amelyekkel alkalmazkodhat az ügyfélhez. Az ügyfelek sem egyformák és nem szabad egy nagyon fejlett metodológiát erőltetni egy olyan vállalatra, amely még nem érett meg arra.

A tényleges megvalósítás során pedig nem csak a szoftver funkcionalitására kell koncentrálni, hanem óriási energiát kell fektetni a végfelhasználói élménybe. És ebbe nem csak a szoftver kinézete tartozik bele, hangsúlyozza Tárkányi Ferenc, hanem sokkal inkább az, hogy milyen könnyen tudja megoldani a problémáját, milyen könnyen tud végigmenni egy-egy folyamaton. Ez pedig már service design tapasztalatokat igényel, miközben az ügyféllel is meg kell értetni, hogy ne a belső (ügyintézői) nézőpontjából terveztesse meg a szolgáltatási útvonalakat, hanem gondolkodjon saját ügyfelei fejével. Ha a szoftver megrendelője magától nem gyűjtené a végfelhasználói visszajelzéseket, a fejlesztőpartnernek kell rábírnia erre a sikeres végeredmény érdekében.

Schopp Attila