ELSŐ FEJEZET

# A Team System áttekintése

**A fejezet tartalma:**

- Élet a Visual Studio 2005 Team System nélkül
- Módszertanra szükség van
- A Visual Studio 2005 Team System
- Szerepek a Team Systemben
- A Visual Studio 2005 kiadásai
- Visual Studio 2005 Express kiadásai
- Visual Studio 2005 Standard Edition
- Visual Studio 2005 Professional Edition
- Együttműködés más Microsoft termékekkel
- Összefoglalás

Jelen fejezet arra szeretne emlékeztetni, mennyire összetett és felelősségteljes lehet a munkánk. Úgy tűnik, minél gyorsabb és többet tud a számítógépes hardver és szoftver, annál bonyolultabb létrehozni az azokat használó intelligens rendszereket. Vajon a dolgok azért látszanak bonyolultabbnak, mert a szoftvertervezési lehetőségek szélesebb skálájáról választhatunk? Vagy azért, mert azoktól a régi, jól ismert problémáktól szenvedünk, amelyektől eddig is? Egy jó tanácsadó erre így válaszolna: „az attól függ."

## Élet a Visual Studio 2005 Team System nélkül

Mindannyian átéltünk már hasonlót. A kibocsátás határideje közeledik, a fejlesztők a tervezőkkel vitatkoznak valamin, a tesztelők csak ülnek, feladat nélkül, a programozók pedig olyan megoldások hozzáadásával vannak elfoglalva, amelyek nem szerepelnek a követelmények között. A bajokat tetézendő a megrendelő éppen minket hív. Ismerős a helyzet?

Szoftvert fejleszteni manapság rendkívül nehéz. A vállalati fejlesztők már nem generálhatnak egyetlen Microsoft® Visual Basic® végrehajtható fájlt, és nem telepíthetik flopilemezről egy Microsoft Office Access adatbázissal együtt. Nem, a mai vállalati méretű alkalmazások sok réteget és szolgáltatást tartalmaznak. Ilyen projektek

fejlesztéséhez nagyobb erőfeszítésre van szükség. Ugyanakkor magasabb a színvonaluk, mint a pár évvel ezelőtti hasonló projekteknek. Számításba kell vennünk azt is, hogy egyre több vásárló, CIO (Chief Information Officer) és más szereplő olvas magazinokat és web blogokat. Bár ők nem járatosak a dolgok technikai oldalában annyira, mint a fejlesztők, azt tudják, mit szeretnének, és ami még fontosabb, valószínűleg azt is, hogy mit nem: egy egyszerű .exe és .mdb fájlegyüttest.

A vállalati méretű szoftvereknek jelentősen több mozgó összetevője van, mint 10-15 évvel ezelőtt. A webalkalmazások, az XML és a webszolgáltatások könnyebbé és jobbá tehetik a rendszereket, de ez csak a tervezők, szereplők vagy felhasználók szempontjából érvényes. Elérhetőségi, együttműködési képességbeli és kiterjeszthetőséggel kapcsolatos előnyeik összehasonlíthatatlanul nagyok. A hátrányuk viszont, hogy az ilyen rendszereket nem könnyű létrehozni. Mindezen szolgáltatások, csomagok és API-k (Application Programming Interface, alkalmazásprogramozói felület), melyeket XML-webszolgáltatásokként ismerünk, összerakása és hatékony együttműködésük biztosítása kihívást jelent bármilyen méretű csapat számára.

## Globális kommunikáció

A földrajzilag különálló egységek a vállalati méretű alkalmazások létrehozásának és gyakorlatba való átültetésének akadályát jelentik. Ma, amikor a távmunka és a munkakihelyezés mindennaposnak számít, jó esélyünk van arra, hogy legalább egy olyan csapattagunk legyen, aki távoli munkaállomásról végez projektmenedzselést, tervezést, fejlesztést vagy tesztelést. A csapat nem minden tagja fér hozzá a belső hálózathoz, vagy élvezheti a virtuális magánhálózat előnyeit. Előnyös lehet, ha vannak távolról dolgozó csapattársaink. A fejlesztői csapat és a napi üzleti zaj fizikai szétválasztása kulcsfontosságú előny a legtöbb módszertanban. Amikor azonban egy projektben távolról dolgozó tagok is részt vesznek, fontos, hogy minden kommunikáció, beleértve a kódokat is, zökkenőmentesen átjusson bármilyen tűzfalon. Az olyan helyi hálózaton alapuló eszközök, mint a Microsoft Visual SourceSafe® 6.0, használata távoli munkára régóta nem praktikus. Aki már próbálta, tudja, miről beszélünk. Meg lehetett csinálni, de ehhez VPN-kapcsolatot kellett létesítenünk a SourceSafe adatbázisunkhoz. A Visual SourceSafe tervezésekor nem gondoltak arra, hogy valaki vékonyklienskent szeretné azt használni a 80-as porton és tűzfalakon keresztül.

**Megjegyzés:** A Visual SourceSafe egyik újdonsága, hogy távolról is tudjuk használni a 80-as porton át, egy webszolgáltatás-interfészen keresztül.

A kommunikáció sosem egyszerű, legyen a csapat szétszórva a földgolyón, egy országban, egy városban vagy akár csak egy irodaházban. A csapattagok általában a saját helyükön dolgoznak, és a projektet a saját személyi portáljukon keresztül látják. Remélhetőleg a kommunikációjukban alkalmazzák az e-mailt-t, a telefont vagy az azon-

nali üzenetküldést (Instant Messaging, IM). Különösen ez utóbbit. El sem tudjuk képzelni, milyen egyszerű együtt dolgozni ezen eszköz segítségével. Kódok másolása és beillesztése, fájlok átvitele és egy gyors kérdés megválaszolása: ezek mind olyan feladatok, melyeket az azonnali üzenetküldés könnyedén kezel. A Microsoft .NET tanulása során rendszeresen fordultam az elérhető ismerőseimhez, hogy olyan rövid és egyszerű kérdéseket tegyek fel, mint „Melyik névtérben találom a *StringBuilder* osztályt?" Ha jól használják, az azonnali üzenetküldés nagyban javítja a kommunikáció hatékonyságát egy projekt hasonló beállítottságú szakértői között. A hálózati élő találkozók és konferenciahívások is eredményesek lehetnek. Az e-mail pedig nagyszerűen archiválja a múltbeli kérdéseinket és a csapattársak által nyújtott erőforrásokat.

E sokféle kommunikációs eszköz hátránya az, hogy kényszerítik a fejlesztőket, hogy a Visual Studión kívüli eszközt használjanak, pedig elsődleges fejlesztői környezetüknek a Visual Studiónak kellene lennie. Sok fejlesztő tölt ezen a felületen napi nyolc órát vagy többet. A Visual Studio elhagyása és egy különálló eszköz, mint a Microsoft Office Outlook® vagy más egyéni feladatot megoldó vagy hibakövető eszköz elindítása nem kedvez az időnk hatékony kihasználásának. Tény, hogy a menedzserek és más fejlesztők számos különböző formában küldenek feladatokat: e-mailben, azonnali üzenetben, hanghívással vagy akár egy papírcetlivel az ajtónkon. E munkaelemek összegyűjtése és rendszerezése nagy erőfeszítésbe kerül.

Az Outlook vagy más együttműködést segítő eszköz Visual Studióba történő beágyazása helyett a Microsoft létrehozta a Team Systemet, mely egyesíti a feladatok, hibák és teljesítések küldését és fogadását közvetlenül az integrált fejlesztői környezeten (Integrated Development Environment, IDE) belül. Mint azt később látni fogjuk, nem szükséges elhagynunk a Visual Studiót, hogy más csapattagokkal a projektről beszéljünk. Ha a csapat minden tagja használja, mindenki nyer.

## Túl sok eszköz

Bármekkora szoftverfejlesztő csapatot figyelünk is meg, termékek tömegének a használatát vehetjük észre. A projektmenedzserek a Microsoft Office Projectet és a Microsoft Office Excel®-t használják a követelmények, iterációk, fázisok, mérföldkövek és kiadásra kész termékek nyomon követésére. A rendszertervezők a Microsoft Office Visio®-t vagy más külső gyártótól származó eszközöket használnak adatközpontjaik, hálózataik, alkalmazásszolgáltatásaik és osztályaik ábrázolására. A fejlesztőknek könnyű dolga van. Ők most és a későbbiekben is a Visual Studiót használják e szolgáltatások és osztályok implementálására. A szoftvert tesztelni kívánó fejlesztőknek és tesztelőknek is szükségük van eszközökre, melyekkel az egységteszteket hajtják végre, kódlefedettséget mérnek, statikus elemzést végeznek, terhelés- és webes teszteket végeznek. Ha szerencsések, ezek a tesztelőeszközök beépülnek a Visual Studióba, vagy legalábbis felügyelt kódban futnak. Ha ilyen sok eszközt használunk, a merevlemezünk tele lesz

különböző gyártóktól származó szoftverekkel. Az újonnan felvett csapattagoknak többhetes képzésre lesz szükségük, hogy tapasztalatot szerezzenek minden speciális alkalmazásban. Üdv a Team System nélküli életben!

A projektek másik lehetséges gyengesége az útmutatás hiánya. Ez nem azt jelenti, hogy a projektmenedzserek vagy csapatvezetők gyenge egyéniségek. Épp ellenkezőleg. Gondoljunk csak arra, amiről korábban már volt szó, hogy milyen bonyolultak tudnak lenni a mai rendszerek. Az útmutatás lehet a legjobb módja az összes szolgáltatás és keret összeillesztésének, valamint ez lehet az a módszertan is, melyet követve sikeresen létrehozhatjuk a szoftvert, megadva ehhez a csapattagokat, a fázisokat és a kiadásra kész dokumentumokat.

Találkoztam néhány rendkívül eszes fejlesztővel és rendszertervezővel a Microsofton belül és kívül is. A Microsoft területi vezetői között és MVP programjaiban nagyon sok ilyen ember van. Egy ember sem tudhat azonban mindent. Egy jó keresőmotorral, olvasási hajlandósággal és sok szabad idővel meg lehet közelíteni az eszményt. Ám amíg egy ilyen ritka ember nem csatlakozik a projektünkhöz, szükségünk lesz folyamatsegédletekre és szabályokra, lehetőleg szaktekintélyek által válogatva. Egy ilyen biztonsági megoldással előrehaladhatunk és írhatjuk a kódot, biztosan tudva, hogy azt megjelentetett és elfogadott szabályokat követve tesszük. Ám még ekkor is érvényes, hogy a fejlesztési folyamat során létrehozott útmutatóknak testre szabhatónak kell lenniük. Ez a követelmény fontos, hiszen lehetővé teszi, hogy csapatunk gyakorlata és szokásai felülírhassák a Microsoft által válogatottakat.

**Megjegyzés:** A Microsoft Patterns and Practices csapat feladata pontosan az alapvető szabályok dokumentálása és kiadása. Ez a csapat belső körökben Prescriptive Architecture and Guidance (PAG, előíró architektúra és útmutatás) néven ismert. Az ő dokumentációik szabályokat tartalmaznak minden elképzelhető területhez: adatok, biztonság, konfiguráció és telepítés, csak hogy párat felsoroljunk. E gyakorlatokat és szokásokat ingyenesen letölthetjük a *http://www.microsoft.com/patterns* címről.

A többféle eszköz használatával kapcsolatos másik probléma a kommunikációra vonatkozik. Nem a csapattagok közöttire, hanem a projektére. A projektnek magának is kommunikálnia kell a projektmenedzserrel és más szereplőkkel. E személyeknek folyamatosan rajta kell tartaniuk az ujjukat a projekt érverésén. A projektmenedzsereknek ma a Projectre és az Excelre kell bízniuk magukat, ha részletes és pontos képet szeretnének kapni. A projekt állapotáról így kapott kép azonban csak annyira pontos, amennyire a Project- vagy Excel-dokumentumokba bevitt adatok azok. Ez a függőség gondokat okozhat, ha be nem tervezett jelentésekre van szükség, az adatok pedig hiányoznak. Ideális esetben a projekt mutatóinak valós időben kellene a projektmenedzser rendelkezésére állniuk, további aggregáció nélkül. Ilyen mutatók a kiemelkedő feladatok, a kielégítő feladatok, a kód változása, a csapathatékonyság és a hibák. E mutatóknak távolról is hozzáférhetőeknek kell lenniük, könnyen elérhető módon, például egy webböngészőből.

## A problémák megoldása

Vannak szoftverfejlesztési problémák, melyek sosem lesznek megoldhatóak. Sok problémás helyzetben az ügyféllel, csapattal, projekttel, költségvetéssel és/vagy időkerettel kapcsolatos nehézségek már túlmutatnak bármilyen eszköz vagy módszertan által nyújtott segítségen. Az ilyen helyzetek kibogozása már túlmutat e könyv és a Team System keretein. Tanácsos az összes részt vevő személy közötti kommunikáció fejlesztése, és nemcsak technikai, hanem pszichológiai és filozófiai szempontból is.

És mi a helyzet a fentiekben felsorolt problémákkal? Tudjuk, melyek voltak ezek: a rossz csoportbeli kommunikáció, a távmunkával kapcsolatos gondok, a nem integrálható eszközök problémája, az útmutatás hiánya és a projekt állapotának ellenőrizhetetlensége. A sikert legnagyobb eséllyel az alábbi elvek alkalmazásával érhetjük el: megfelelő tervezés, jó rendszertervek, szabályok követése, tesztelés és hatékony kommunikáció. Ezt tudva nem lenne jó egy olyan eszköz, amely a segítségünkre lehetne e célok elérésében? Nem lenne nagyszerű, ha ez az eszköz egyből egy jól ismert fejlesztői környezetbe épülne, melyet már így is napi 8-18 órát használunk? Ez az eszköz már létezik, és Microsoft Visual Studio® 2005 Team Systemnek hívják.

## A Visual Studio 2005 Team System céljai

A Microsoft valami egyedit hozott létre a Visual Studio Team Systemmel: tett egy lépést hátrafelé, és megvizsgálta, hogyan fejlesztenek szoftvert sikeresen (vagy néha sikertelenül) a szoftverboltok és -csapatok. Ezután taktikus módon létrehozott egy terméket, amely megnöveli egy termék sikerének megjósolhatóságát. A Team System a projekt elejétől végéig történő útmutatásával és együttműködést segítő megoldásaival éri el ezt a megjósolhatósági szintet. A Microsoft a Team System azt is megváltoztatja, ahogyan a Visual Studióra gondolunk. A Visual Studio többé már nem csak egy szép integrált fejlesztő környezet. A Team System olyan hatékony eszközzé alakítja, amely a teljes csapatot integrálja a megoldások teljes halmazával a termék életciklusának egészében.

A Team System magja ugyanazon az eszközkészleten alapul, amelyet a Microsoft belsőleg használ szoftverei sikeres tervezéséhez, létrehozásához és telepítéséhez, a hozzánk hasonló felhasználók számára. Még ha a cégünk nem is egy több milliárd dolláros szoftverfejlesztő cég, a Microsoft tudja, hogy más szoftverfejlesztő csapatok is olyan problémákkal küzdenek, mint az övé. A Microsoft részéről érthető lépés volt felújítani és újra eladni ezt az eszközkészletet, ezáltal megosztva belső ajánlott gyakorlatait velünk, többiekkel.

A Visual Studio 2005 Team Systemet a következő elsődleges célokra tervezték:

- a projektsikeresség megjósolhatóságának növelése;
- a csapat termelékenységének növelése azáltal, hogy csökkentjük a szállítandó modern szolgáltatásorientált megoldások bonyolultságát;
- a csapat együttműködésének növelése az eszközök egyesítésével;
- a csapat hatékonyságának növelése azáltal, hogy a távoli felhasználók robusztus, biztonságos, és méretezhető környezetben dolgozhatnak;
- a bővíthetőség megteremtése azáltal, hogy a szolgáltatásai testre szabhatóak és kibővíthetőek.

Az utolsó cél azt jelenti, hogy külső szállítóknak is képesnek kell lenniük Team System beépülő modulok előállítására. Ha a csapatunk egy másfajta módszertant, tervezőt, verziókezelő- vagy tesztrendszert használ, akkor ezeket az eszközöket a Team Systemmel integrálni tudjuk. Ez a cél tükrözi a Microsoft folyamatos elkötelezettségét partnerei és a fejlesztőközösségek felé, ilyenek például a független szoftverszállító (Independent Software Vendor, ISV) vagy a Visual Studio Integrator Program (VSIP) programok.

**További információ:** Feltétlenül olvassuk el a 9. fejezetet, hogy megismerjük a Team System testreszabhatóságának és bővíthetőségének sokféle módját. Megtudhatjuk többek között, hogyan szabjuk testre az útmutató dokumentációkat és sablonokat.

## Folyamatsegédletek és módszertanok

Vajon jó egy olyan eszköz használata, amelyik megmondja, milyen módszertant kövessünk? Sok fejlesztővel és rendszertervezővel beszéltem, akinek az a véleménye, hogy a Microsoft tartsa távol magát a „hogyan fejlesszünk szoftvert" üzlettől, és maradjon meg az azt létrehozó eszközök fejlesztésénél. Ezzel nagyrészt egyetértek. Az emberek nem szeretik, ha módszertanokat vagy folyamatokat erőltetnek rájuk. A Team System fejlesztésekor a Microsoft is pontosan ezt tartotta szem előtt. Visszatérve a kiterjeszthetőséghez, a Team System minden irányban testre szabható és kiterjeszthető. Ha az eszköz által feltett, segédletekkel kapcsolatos vagy módszertani kérdések nem megfelelőek a csapatunk számára, akkor megváltoztathatjuk azokat. Ha a verziókezelésünk beadási irányelvei túl szigorúak, vagy éppen nem elég szigorúak, azokat is megváltoztathatjuk. Ha a csapatunk által kapott feladatok száma okoz zavart, akkor azt változtatjuk meg.

Sok éve tanítok tervezést, fejlesztést és alapszabályokat szoftverfejlesztő csapatoknak. Ezeknek a tagjai a létező legnagyobb kihívást jelentő és legaprólékosabb szakemberek. Pontosan tudják, mit akarnak. De ami még fontosabb, azt is pontosan tudják, mit *nem* akarnak. Nincs szükségük olyan eszközökre, tanfolyamokra vagy ta-

nácsokra, amelyek akadályozzák a munkájukat. Nincs szükségük új módszertanra. Már túl sok mindent láttak jönni és menni. Valami nem túl bonyolult dolgot szeretnének, ami bevált módszereket és a termelékenység növekedését jelenti. Ha ez a dolog történetesen kódot is generál, az csak még jobb. Mivel a Microsoft hasonló mentalitású emberekből áll, ezért már a kezdetekben tudta, hogy a Team System csak akkor lehet sikeres, ha bővíthető.

# Módszertanra szükség van

Álljunk meg egy pillanatra. Úgy gondolom, és ezzel a legtöbben egyet is értenének, fontos, hogy legyen egy módszertanunk, függetlenül attól, hogy milyen. Ha az a módszertanunk, hogy a napi teendőinket megmaradó jegyzetekre írjuk, rangsoroljuk őket, kirakjuk őket a falra, majd levesszük, ha elvégeztük őket, akkor máris van egy módszertanunk. Már maga a „módszertan" kifejezés sem elég egyértelmű. Egyszerűen annyit jelent, hogy van egy tömör és egyértelmű megközelítésünk valami elvégzésére. Találkozhatunk jól és kevésbé jól ismert módszertanokkal. Ez a terület már régóta érdekelt, mivel a diszkrét szervezési és a kemény technikai készségek határmezsgyéjén található.

**Megjegyzés:** A Team System önmagában nem módszertan. Csak egy egyszerű eszköz, amely útmutatást ad, és megmondja, mely területeken kell még javítanunk. Ezt a használt módszertantól függően teszi. A fentiek tudatában figyeljük meg a Team System iróniáját: néhány csapatnak szüksége van egy olyan eszközre, amely lehetővé teszi egy olyan módszertan használatát, amely segíti őket az eszköz használatában.

A következő néhány oldal bemutat néhány népszerűbb módszertant, többek között a Team Systemben található két Microsoft Solutions Framework (MSF) módszertant.

**További információ:** Feltétlenül olvassuk el a 8. fejezetet, ha mélyebben meg szeretnénk ismerni az MSF-módszertanokat.

## Microsoft Solutions Framework

A Microsoft Solutions Framework (MSF) szoftverfejlesztési folyamatok, elvek és bizonyított módszerek gyűjteménye, melyekkel a fejlesztők sikereket érhetnek el a szoftverfejlesztési életciklus (Software Development Life Cycle, SDLC) során. Az MSF mélyen gyökerezik az iparági ajánlott gyakorlatok körében. Első bemutatója 1994-ben volt, de a mai napig már 25 évnyi útmutatást gyűjtött össze belső és külső forrásokból egyaránt.

Az évek során az MSF folyamatosan igazodott a fejlesztők változó elvárásaihoz. Az MSF 4.0, amely a Team System része, két változatra osztható, mely lényegében két különálló szoftverfejlesztési filozófia: az *„MSF az agilis szoftverfejlesztéshez”* és az *„MSF a CMMI folyamatfejlesztéshez”* (MSF for CMMI Process Improvement). Az MSF az agilis szoftverfejlesztéshez azoknak fog tetszeni, akik agilisták, azaz az olyan csapatoknak, amelyek hozzászoktak a gyors, változékony környezetekhez, melyek szorosan összekapcsolódnak az ügyféllel. Az MSF a CMMI folyamatfejlesztéshez a nagyobb cégeknek vagy projekteknek előnyös, melyek sok jelentési szintet foglalnak magukba. Az ilyen projektekben a hosszú távú tervezés és a kommunikáció fontosabb, mint az állandó kiadásra kész termékek és a visszacsatolás.

## MSF az agilis szoftverfejlesztéshez

Az MSF az agilis szoftverfejlesztési folyamathoz teljesen új. Kisebb cégeknek szánják, amelyekben öt és húsz fő közötti a fejlesztői csapatok létszáma. Az MSF agilis szoftverfejlesztéshez szánt változatát külön a Team System számára tervezték? Valószínűleg igen. A Team Systemet azért hozták létre, hogy kiegészítse az MSF agilis szoftverfejlesztésre szánt változatát? Valószínűleg igen. A folyamat és az eszköz egyesítése nagyon is természetes.

Az agilis folyamatmodellt hivatalosan az Agile Alliance nevű szervezet hozta létre. Az alábbiakban olvasható az Agile Alliance néhány alapelve.

- Az egyének és kapcsolataik fontosabbak a folyamatoknál és az eszközöknél.
- Az ügyféllel történő együttműködés fontosabb a szerződéseknél.
- A működő szoftver fontosabb, mint az átfogó dokumentáció.
- A változásokra való reagálás fontosabb a terv követésénél.

A következetes és minőségi szoftverek szállítása agilis folyamatot feltételez. A formális specifikálási fázisok ideje lejárt. Az agilis szoftverfejlesztésre szánt MSF-fel a Microsoft megmutatja, hogy felismerte, ha voluntarista specifikációkat és követelményeket dobálunk a fejlesztők nyakára, az gyakran vezet sikertelen projektekhez.

Az MSF az agilis szoftverfejlesztéshez a Team System alapértelmezett módszertana. Ezt a szoros csatolást használat közben érezzük igazán. Az alapértelmezett objektumok, mint a feladatok és a hibák, maguktól értetődőek lesznek a Visual Studiót már használt fejlesztők számára.

## „MSF a CMMI folyamatfejlesztéshez” (MSF for CMMI Process Improvement)

A CMMI a Capability Maturity Model Integration kifejezés rövidítése. Célja, hogy modellt biztosítson az állandó folyamatjavítás számára. Ez rövidebb idejű szoftverfejlesztési ciklusokat eredményez, a költségekkel és határidőkkel kapcsolatos célok pontosabb betartását, továbbá jobb minőséget. Ez egy formális módszertan, melyet a Carnegie Mellon University Software Engineering Institute részlege kezel.

A CMMI használatának nagy előnye, hogy kipróbált szabvány, mellyel összehasonlíthatjuk szoftverfejlesztési képességeinket más cégekéivel (az Egyesült Államokban). A Védelmi Minisztérium és más nagy szoftverhasználók gyakran nézik meg a beszállítóik által elért CMMI-besorolást, amikor arról döntenek, ki kapjon meg egy fejlesztési szerződést.

A Team System a szoftverfejlesztéshez készített „CMMI a folyamatfejlesztéshez” változatát tartalmazza. Ez kiváló folyamat, ha a cégünk mért referenciaadat-besorolást szeretne kapni a szoftverfejlesztés területén. Az MSF agilis változatánál jóval több dokumentumot és jelentést kell írnunk ebben a modellben, de ez a formális fejlesztési folyamat csökkenti a kockázatot a nagy szoftverprojektek esetében, és olyan referenciaadatot nyújt, mellyel tanúsítványokat szerezhetünk a különböző CMMI-szintekről, illetve megkaphatjuk az ISO 9000/9001-es minősítéseket.

## eXtrém Programozás (XP)

Az extrém programozás egy másik agilis szoftverfejlesztési módszertan, amely lehetővé teszi, hogy a csapat (fejlesztők, ügyfelek és projektvezetők) kezelni tudja a projekt fejlődése során bekövetkező változásokat. Mint tudjuk, a projekt során gyakran változhatnak a követelmények. Az XP esetében majdnem minden változhat, beleértve a megrendelői és fejlesztői csapatok tagjait, valamint az üzleti környezetet. A vízesés és más rögzített szoftverfejlesztési életciklusok nem tudják könnyen kezelni a változásokat, különösen a későbbi ciklusaikban. Az XP egy olyan szoftverfejlesztési életciklust ad, amely nem csupán követi a változásokat, hanem elegáns módon kezeli is őket. Az XP-módszerek ahelyett, hogy megpróbálnák irányítani a változást, lehetővé teszik, hogy a csapattagok könnyedén igazodhassanak hozzá.

**Tipp:** „A szoftverben minden változik. A követelmények változnak. A rendszerterv változik. Az üzlet változik. A technológia változik. A csapat változik. A csapattagok változnak. A probléma nem maga a változás, mivel az mindenképpen bekövetkezik. A probléma inkább az, amikor képtelenek vagyunk megbirkózni a közelgő változással.”

Kent Beck, *eXtreme Programming Explained*

Az XP-nek számos előnye van:

- Az ügyfelek már egy iteráció múlva megkapják az első terméket, amellyel elkezdhetnek dolgozni. (Az első tényleges iteráció a fejlesztői csapat számára az lehet, hogy beállítsa a használandó környezeteket.)
- A fejlesztők a legfontosabb feladatokon dolgoznak, melyek fontossági sorrendjét a megrendelők adják meg.
- A rendszer legfontosabb funkcióit és megoldásait vetik alá a legtöbb tesztelésnek.
- A leszállított termék megfelel a megrendelő szükségleteinek.

**Megjegyzés:** Ez csak áttekintés az XP-ről. A fenti értékek és gyakorlatok teljes leírásait számos jelenleg elérhető, színvonalas könyvben megtalálhatjuk.

## Scrum

A *Scrum* egy másik agilis fejlesztési módszertan, amely a szakaszos átadásra koncentrál kis fejlesztői csapatokkal, rövid fejlesztési ciklusokkal és egy közvetítővel, aki elősegíti a csapat figyelmének összpontosítását és a termelékenységet.

Egy Scrum projekt egy vagy több *sprintre* bomlik. Ez egy jól meghatározott fejlesztési ciklus, mely általában négytől hat hétig tart. A kezdeti projekttervezés végeztével a megrendelők és a fejlesztői csapat közösen meghatározza az első sprint kiadásra kész termékeit. Ezután a csapat minden más feladatot félretéve megkezdi az első sprintet. A sprint rövid időtartama erősíti a közvetlenség érzetét, ami megőrzi a csapat motivációját. A sprint segít, hogy a csapat lássa a folyamat végét, mivel az legfeljebb csak néhány hétnyire van. Minden sprint visszatekintéssel végződik, amikor a csapat értékeli, hogyan sikerült. Ennek fényében javaslatokat tesznek a következő sprint javítására. A sprint vége jelzi az újabb sprint kezdetét. Ez addig folytatódik, amíg a projekt be nem fejeződik.

A csapat középpontja a *scrummester,* egy olyan közvetítő, akinek az elsődleges és gyakran egyetlen feladata útmutatást adni, hogy a csapat sikeresen teljesítse a sprintet. A scrummester *napi találkozókat* tart, amelyek a teljes fejlesztői csapat rövid megbeszélései. A napi találkozók a csapat minden tagja számára három kérdésből állnak:

- Mit végeztél az utolsó találkozó óta?
- Mit tervezel elvégezni a következő találkozóig?
- Mi akadályoz ebben?

A találkozó során felmerülő minden egyéb téma áthelyeződik azt követő megbeszélésekre, melyeken csak az érdekeltek vesznek részt, így a scrumtalálkozó rövid és hatékony marad.

A scrummester felelős a napi találkozókon felmerült problémák kiküszöböléséért. Ez a szerep, amely gyakran hiányzik a fejlesztőcsapatokból, kulcsösszetevője a scrumfolyamatnak. Az akadályok, melyek a fejlesztőkörnyezettel kapcsolatos problémáktól kezdve a függőben hagyott döntéseken át a felfelé átadható anyagok hiányáig terjedhetnek, nagy hatással lehetnek a fejlesztőcsapatra. Azzal, hogy e zavaró tényezőkért egy embert teszünk felelőssé, lehetővé tesszük, hogy a fejlesztők azokra a problémákra koncentrálhassanak, amelyekkel éppen foglalkoznak.

A fenti egyszerű technikákat alkalmazva a szoftverfejlesztő csapat jelentős fejlődést mutathat fel a termelékenységben és a hozzáállásban.

**Megjegyzés:** A módszertan neve a rögbiből ered. A *scrum* a *scrummage* (a játék elején felvett formáció elnevezése) rövidítése, amely később az amerikai fociban használatos *scrimmage* fogalommá alakult.

## Hogyan támogatja a Team System e módszertanokat?

Nem számít, milyen emelkedett és elméleti a módszertanunk, bele kell helyeznünk a gyakorlatba, és el kell végeznünk a munkánkat. Itt lép be a képbe a Team System. A Team System munkaalapú megközelítést használ, azaz egy folyamatot úgy határoz meg, hogy tevékenységek és altevékenységek gyűjteményét adja. Ez a modell jól illeszkedik a legtöbb módszertanra.

A Team System például tud az MSF Agile Software Development elemekről: a szerepekről, munkaelemekről, tevékenységekről és munkafolyamokról. Szerep lehet az üzleti elemző, a projektmenedzser, a rendszertervező, a -fejlesztő, a -tesztelő és a kiadásokért felelős menedzser. *Munkaelemek* a különböző forgatókönyvek, a QoS-követelmények, a kockázatok, a feladatok vagy a hibák. E munkaelemek kapcsolódhatnak különböző objektumokhoz, például dokumentumokhoz, táblázatokhoz, projekttervekhez, forráskódhoz és a tevékenységek más megfogható kimeneteihez. Munkaelemek akkor keletkeznek, amikor bizonyos tevékenységek befejeződnek, de lehetnek más tevékenységek előfeltételei is. A *tevékenységek* bizonyos célok érdekében együtt elvégzett munkamintázatok. Egy tevékenység használhat vagy előállíthat munkatermékeket, és nyomon követhető egy munkaelemmel. A tevékenységek munkafolyamokba csoportosíthatók. A *munkafolyamok* olyan tevékenységek, melyek más tevékenységekből állnak. A folyamat egyszerű építőkockái, melyek hozzárendelhetők egy vagy több szerephez.

## Forgatókönyvek

A forgatókönyv egyfajta munkaelem, amely a felhasználói beavatkozás egyetlen útját rögzíti a rendszeren keresztül. Miközben az egyén megpróbál elérni egy célt, a forgatókönyv rögzíti az általa véghezvitt lépéseket. Némelyik sikeres utat rögzít, mások sikerteleneket. Forgatókönyv írása közben legyünk konkrétak! Mivel a legegyszerűbb rendszereket leszámítva végtelen számú forgatókönyv lehetséges, fontos, hogy odafigyeljünk, melyik eseteket írjuk meg.

**Megjegyzés:** A forgatókönyveket gyakran hasonlítják az UML (Unified Modeling Language, Egységesített modellező nyelv) használati eseteihez.

## Szolgáltatásminőségi követelmények

A szolgáltatásminőségi (Quality of Service, QoS) követelmények egy rendszer jellemzőit írják le, például a teljesítményt, terhelést, rendelkezésre állást, teherbírást, elérhetőséget, javíthatóságot és karbantarthatóságot. E kívánalmak gyakran megkötéseket jelentenek a rendszer működésére nézve.

**Megjegyzés:** A QoS-követelmények nem egyeznek meg a funkcionális követelményekkel.

## Feladat

Egy *feladat*-munkaelem valamilyen munka elvégzésének szükségét jelzi. Minden szerep saját követelményekkel rendelkezik egy feladatra vonatkozóan. Egy fejlesztő például fejlesztői feladatokat használ olyan munkák komponens tulajdonosokhoz rendelésére, melyek forgatókönyvekből vagy QoS-követelményekből származnak. A tesztelő tesztfeladatokat használ arra, hogy hozzárendelje valakihez a tesztforgatókönyvek megírásával és lefuttatásával kapcsolatos munkákat. A feladatokat arra is használhatjuk, hogy regressziótesztet indítsunk, vagy javasoljuk felderítő tesztek elvégzését. Végül a feladatokat használhatjuk munkák általános hozzárendelésére a projekten belül. A munkaeleműrlapon néhány mezőt csak olyan esetben használunk, ha a feladat egy adott szereppel áll kapcsolatban.

## Kockázat

A projektmenedzsment lényeges összetevője az inherens kockázatok azonosítása és kezelése. *Kockázat* bármilyen valószínű esemény vagy állapot, melynek a jövőben negatív következménye lehet a projektre. A kockázat-munkaelem dokumentálja és nyomon követi a projekt műszaki vagy szervezeti kockázatait. Ha konkrét cselekvésre van szükség, ezeket a kockázatokat feladatoknak feleltethetjük meg, melyeket elvégezve mérsékelhetjük a kockázatot. Egy műszaki kockázat például hátráltathat egy architekturális prototípuskészítési munkát. A csapatnak mindig pozitívan kell viszonyulnia a kockázatazonosításhoz, hogy a lehető legtöbb információt biztosítsák a

fennálló kockázattal kapcsolatban. A környezetnek olyannak kell lennie, melyben a kockázatokat azonosító egyéneknek nem kell megtorlástól tartaniuk, ha őszintén hangot adnak kétségeiknek vagy ellentétes véleményüknek. A pozitív kockázatkezelési környezetben dolgozó csapatok sikeresebbek a kockázatok feltárásában és kezelésében, mint negatív kockázatkezelési környezetben dolgozó társaik.

### Hiba

A *hiba* olyan munkaelem, amely azt fejezi ki, hogy egy lehetséges probléma létezik vagy létezett a rendszerben. A hiba nyitásának célja, hogy pontosan jelenthessük a hibákat, melynek segítségével az olvasó megértheti a probléma teljes hatását. A hibajelentés leírásának olyannak kell lennie, hogy könnyen nyomon követhessük a lépéseket, melyek során a hibával találkoztunk, azaz a segítségével a hibának könnyen reprodukálhatónak kell lennie. A teszteredményeknek világosan mutatniuk kell a problémát. A hiba kijavításának esélyei gyakran függnek a leírás tisztaságától és érthetőségétől.

## Módszertanok testre szabása

Mint már korábban említettük, a Team System nem egy teljes módszertani eszköz. Például csak a szűk fejlesztői csapatot támogatja. Nem ütemez be megbeszéléseket, nem készít elő költségvetést, nem küld e-maileket vagy folytat közvetlen kommunikációt az ügyféllel vagy más szereplőkkel. Egy nagyvállalatnál, ahol az ügyfél helyben van, valószínűleg csak egy másik részlegen vagy osztályon, ez a funkció elhagyható. Egy független szoftverszállítónál viszont, amely termékeit a hálózaton keresztül értékesíti, nem nélkülözhető.

A Team System nagyszerű munkát végez projektmunkaelemeink, elvárásaink és együttműködési feladataink szervezése során. Emberekre és megbeszélésekre azonban ezentúl is szükség lesz, hogy azok valósítsák meg a módszertant. Ez már túlmutat a Team System keretein.

# A Visual Studio 2005 Team System

A Team System több a Visual Studio egy kiadásánál. Valójában szerepalapú kiadások sorozata. A Team Systemet nem igazán a magányos szakembereknek vagy tanácsadóknak találták ki. Az előnyei olyan csapatoknál érzékelhetőek, amelyeknél előfordulnak a projektmenedzseri, rendszertervezői, fejlesztői és tesztelői szerepek. Ha egy személyben látjuk el ezt a sok feladatot, akkor is hasznos lehet beszerezni a szoftvert.

**Megjegyzés:** A Team Systemet 5-től 500 aktív főig terjedő létszámú csapatokra optimalizálták.

## Visual Studio 2005 Team Edition rendszertervezőknek

Ezt a kiadást kifejezetten az infrastruktúra- és alkalmazáskonstruktőri szerepek számára tervezték. Találhatunk benne vizuális tervezőket, melyekre *elosztott alkalmazástervezők*ként, vagy szolgáltatásorientált architektúra- (Service Oriented Architecture, SOA) tervezőkként hivatkozunk. A rendszertervező létrehozhat ábrákat, melyek a logikai adatközpontot, az alkalmazást, alkalmazásrendszereket és az alkalmazás telepítését jelenítik meg. Ezek a tervezők az egyszerű „fogd, vidd és kapcsold össze" módszert követik, amely már régóta népszerű a Visual Studióban. Ám a szép alakzatokon túlmutatva az ábrákban megtalálhatók az intelligencia és a metaadatok, melyeknél jól ismert és egyénileg meghatározott megszorítások meglétét ellenőrizhetünk, majd egy gyors kattintással kóddá alakíthatjuk. Az elegáns ábrák kulisszái mögött a Microsoft az információt rendszerdefiníciós modell- (System Definition Model, SDM) fájlokban tárolja, melyek a dinamikus rendszerekkel kapcsolatos kezdeményezés (Dynamic Systems Initiative, DSI) megvalósításai.

**További információ:** A rendszertervezők számára készített Team Editionről az 5. fejezetben tudhatunk meg többet.

## Visual Studio 2005 Team Edition szoftverfejlesztőknek

Ez a kiadás a csapat programozóinak és fejlesztőinek szól. Valószínűleg ez lesz a leggyakoribb a Team System szerepalapú kiadásai közül. A Visual Studio 2005 Professional alaplehetőségein felül a fejlesztők kapnak egy statikus kódelemzőt (hasonló az FxCop-hoz), egy egységtesztelőt (hasonló az NUnit-hoz), egy kódlefedettség-elemzőt és egy kódelemzőt. Néhány megoldás megegyezik a Visual Studio 2005 Team Edition tesztelői változatával. A Microsoft tudja, hogy nehéz meghatározni, melyik szerep (a fejlesztő vagy a tesztelő) legyen felelős e forráskódtesztek megírásáért és lefuttatásáért, ezért ezek mindkét kiadásban megtalálhatók.

**További információ:** A fejlesztők számára készített Team Editionről a 6. fejezetben tudhatunk meg többet.

## Visual Studio 2005 Team Edition szoftvertesztelőknek

Legyen egy csapattag fejlesztő vagy kizárólag tesztelő, ez a kiadás elérhetővé teszi számára az összes kódlefedettségi, minőség- és terheléstesztelő eszközt, amelyre szüksége lehet egy projekt alapos tesztelése során, minden oldalról biztosítva a működését. A tesztelők Team Edition kiadása tartalmaz webes terheléstesztet (hasonló az Application Center teszthez), egységtesztelési és kódlefedettségi eszközöket, továbbá tesztforgatókönyv-kezelő megoldásokat a tesztek kezelésére, futtatására, majd központi monitorozásukra. A tesztelők Team Edition kiadása támogatja továbbá bármilyen saját manuális teszt beépítését. E tesztek közül néhány megegyezik a fejlesztők számára készített Team Editionnel.

**További információ:** A tesztelők számára készített Team Editionről a 7. fejezetben tudhatunk meg többet.

## Visual Studio 2005 Team Foundation Server

A Visual Studio e kiadása sok háttérkiszolgáló-oldali adatbázist és webszolgáltatást tartalmaz, hogy lehetővé tegye a csapat együttműködését a munkaelemek, forráskódok és más erőforrások megosztásával. Ha a Team Systemet a javaslat szerint csapatban szeretnénk használni, szükségünk lesz erre a termékre, hogy összekapcsolja a csapat tagjait. A Team Foundation Server több, mint a Visual Studio 2005 Team System egyszerű „kiadása". Ez a szoftverfejlesztési életciklus hátterében található motor.

A Team Foundation Server tartalmaz egy Team Explorer nevű különálló kliensalkalmazást. Ez az alkalmazás lényegében a Visual Studio 2005 kis teljesítményű kiadása, mely a munkaelemek létrehozásának és kezelésének egy más útját (a Visual Studio egy másik kiadásának, az Excelnek vagy a Projectnek a használata mellett) teszi lehetővé. Az „alkalmi szereplőnek" szánták, vagyis a csapat azon tagjának , akinek dokumentációt kell beadnia, egy webprojekt képeit kell kezelnie és így tovább.

**További információ:** A Team Foundation Serverről a 2. fejezetben, a különböző kliensprogramokról pedig a 3. fejezetben tudhatunk meg többet.

## Visual Studio 2005 Team Suite

Az egynél több szerepet betöltő csapattagnak, illetve az összes feladatot ellátó tanácsadónak ott a Team Suite. A Microsoft az egyszerűség kedvéért egy kiadássá gyúrta össze mindhárom szerepalapú (rendszertervező, fejlesztő és tesztelő) kiadást.

**Megjegyzés:** A Visual Studio 2005 indulásakor az MSDN® Universal és az MSDN Enterprise már nem lesznek megvásárolhatók. Ekkor az aktív MSDN Universal előfizetők frissítési ajánlatot fognak kapni MSDN Premium előfizetésre, valamint további költségek nélkül egy általuk választott szoftverre a következők közül: Visual Studio 2005 Team Edition rendszertervezőknek, Visual Studio 2005 Team Edition szoftverfejlesztőknek vagy Visual Studio 2005 Team Edition szoftvertesztelőknek. Azoknak, akik mindegyik szerepalapú Visual Studio Team System kiadást meg szeretnék venni különleges kedvezménnyel lehetővé teszik, hogy az MSDN Universal előfizetői a Visual Studio 2005 Team Suitere frissítsenek. A legfrissebb információkért és frissítésekért nézzük meg a Team System weblapot az MSDN-en.

# Szerepek a Team Systemben

Ne felejtsük el, hogy egy szerep nem feltétlenül egy embert jelent. Valójában kétséges, hogy bármelyik csapat is pontosan megfelel a Team System modelljének, ahol minden szerep tökéletesen be van töltve, átfedések nélkül. E részt átolvasva emlékezzünk arra, amit a Team System rugalmasságáról olvastunk.

A Team Systemben az alábbi négy szerep létezik:

- Projektmenedzser
- Rendszertervező
- Fejlesztő
- Tesztelő

Ha elég kreatívak vagyunk, elő tudunk állni néhány más szereppel is. Feloszthatjuk például a rendszertervező szerepét két részszerepre:

- Alkalmazásarchitekt: a szoftvereket és szolgáltatásokat tervezi
- Infrastruktúraarchitekt: a telepítési környezetet tervezi, beleértve a hálózatot és az infrastruktúrát

Egy másik Team System által érintett szerep az IT szakértői gárda, amelyet a kész termék célkörnyezetbe történő telepítésére fogunk megkérni. Ezt figyelembe véve más kvázi- és kombinált Team System szerepkört is kitalálhatunk:

- IT szakértő: szoftvertelepítés
- Tesztelő/IT szakértő: szoftvertesztelés és telepítés
- Fejlesztő/tesztelő: szoftverfejlesztés és tesztelés
- Alkalmazásarchitekt/fejlesztő: szoftvertervezés és fejlesztés

# A Visual Studio 2005 kiadásai

A Team System nem része a Visual Studio összes kiadásának. A kezdőknek, hobbiprogramozóknak, diákoknak és a professzionális fejlesztőknek elérhetők a Visual Studio Express, Standard és Professional kiadásai.

- Visual Studio 2005 Express kiadások
- Visual Studio 2005 Standard Edition
- Visual Studio 2005 Professional Edition
- Visual Studio 2005 Team Edition rendszertervezőknek
- Visual Studio 2005 Team Edition szoftverfejlesztőknek
- Visual Studio 2005 Team Edition szoftvertesztelőknek
- Visual Studio 2005 Team Foundation Server
- Visual Studio 2005 Team Suite

**Megjegyzés:** A Team System minden kiadása a Professional kiadáson alapul.

# Visual Studio 2005 Express kiadásai

A Visual Studio 2005 Express kiadásai általában jó megoldások a gyerekek, hobbiprogramozók, diákok, kezdő és lelkes fejlesztők számára. Választhatunk a Microsoft Visual C#® Express, a Visual Basic Express, a Visual C++®, a Visual J#® és a Web Developer Express közül. Az Express-kiadások mindenki számára elérhetők, aki kezdő fejlesztőként használni szeretné a Visual Studio 2005-öt.

# Visual Studio 2005 Standard Edition

A Visual Studio korábbi változatainak standard kiadásaihoz hasonlóan ez a belépőszint bárki számára, aki komolyan gondolja az alkalmazásfejlesztést. A célközönség tagjai a webes szakértők, a Visual Basic 6.0 fejlesztők, valamint a részmunkaidős fejlesztők, akik önálló alkalmazásokat szeretnének írni Visual Basic .NET-ben vagy C#-ban.

# Visual Studio 2005 Professional Edition

A Visual Studio korábbi változatainak Professional kiadásaihoz hasonlóan ez a kiadás való a komoly fejlesztésekhez: tanácsadóknak, egyedül, illetve kis csapatban dolgozó szakembereknek, akiknek a Team System már több, mint amire szükségük van. A Professional kiadás abban is különbözik a Standardtől, hogy megtalálható benne minden megoldás, amely az elosztott alkalmazások fejlesztéséhez szükséges.

# Együttműködés más Microsoft termékekkel

A Team System sok darabból és részből áll össze. A kiszolgáló nélküli kiadásokban a Team System egyszerűen a Visual Studio 2005 Professional kiadása a megfelelő beépülő modulokkal. A Team Foundation Server, amelyről a 2. fejezetben tudhatunk meg többet, szolgáltatások teljes architektúrája.

Az alábbiakban néhány másik Microsoft-terméket sorolunk fel azzal együtt, hogy ezek hogyan épülnek be.

- **Microsoft SQL Server™ 2005** Tárhely az összes munkaelem, forráskód és fordítási adat, beleértve minden csapatszintű erőforrást, számára. Analízisszolgáltatások (OLAP-kockák) és -jelentés. A szolgáltatások támogatják a portált és a különböző csapatjelentéseket.
- **Microsoft Windows® SharePoint® Portal Services (de nem a SharePoint Portal Server)** A csapatprojekt-portál mögötti szoftver.
- **Visual Studio 2005** Elsődleges környezet minden szerep számára.

- **Microsoft Project 2003 (de nem a Project Server)** Egy másik eszköz a projektmenedzserek számára.
- **Microsoft Excel 2003** Egy másik eszköz a projektmenedzserek számára.
- **Microsoft Internet Explorer** A csapatprojekt-portállal történő együttműködésre használják, továbbá jelentések megtekintésére.

**Megjegyzés:** A Microsoft tervezi, hogy letöltésként megjelentet egy Microsoft Source Code Control (MSSCCI) beépülő modult valamikor a Visual Studio 2005 indulása után. Ez lehetővé tesz egy alapszintű forráskód-kezelő integrálást a Team Foundation Version Control és a Visual Studio korábbi verziói között. Addig is, valamint minden más szoftvercsomag vagy platform esetén használhatjuk a különböző parancssori eszközöket vagy API-kat más fejlesztői környezetekkel történő kibővítésre és az azokkal történő integrációra.

A fenti eszközökön felül a Microsoft és partnerei sok Team Systemmel integrált szolgáltatást jelentettek be. A Microsoft tervezi, hogy megjelentet migrációs eszközöket a Microsoft Visual SourceSafe-hez, más partnerek pedig további modellező eszközök, követelménygyűjtő rendszerek és tesztelő eszközök támogatását ígérték.

# Összefoglalás

A Team Systemet a Microsoft hozta létre, egy olyan cég, amely tud egy-két dolgot a tervezésről, fejlesztésről és a nagy volumenű szoftvertermékek szállításáról. Ha a Team Systemet használjuk szolgáltatásorientált megoldásaink fejlesztésére, azzal növeljük projektünk sikerességének megjósolhatóságát azáltal, hogy elősegítjük csapatunk kommunikációját és növeljük termelékenységét. Ezt az integrált eszközök használatával érjük el, dolgozzon a csapatunk helyi hálózaton vagy akár elosztott globális környezetben. A módszertanokat és az útmutatásokat, valamint a lehetőséget a Team System különböző kibővítéseire készen kapjuk.