

FOGALOMTÁR

absztrakt adattípus Olyan adattípus, amelynek rejtett a reprezentációja, és csak műveletei definiálják.

Ada Programozási nyelv, amelyet az amerikai Védelmi Minisztérium számára fejlesztettek ki a hadsereg szoftvereinek szabványos nyelveként. Megalapozásául a 70-es évek programnyelvekkel kapcsolatos kutatásai szolgáltak. Olyan konstrukciókat tartalmaz, mint az absztrakt adattípus és a konkurencia támogatása. Ma is használják nagy és komplex hadászati és légvédelmi szoftverek fejlesztéséhez.

adattfeldolgozó rendszer Olyan rendszer, amely nagy mennyiségű strukturált adat feldolgozásáért felelős. Az ilyen rendszerek általában kötegekben dolgozzák fel az adatokat, és gyakran beolvasás-feldolgozás-kiírás modell szerint működnek. Jó példák adattfeldolgozó rendszerekre a számlázórendszerek.

agilis módszerek Szoftverfejlesztési módszer, ahol a szoftverek gyors legyártását helyezik előtérbe. A szoftvert inkrementekben fejlesztik és adják át, miközben a folyamat dokumentációját és a bürokráciát minimális mértékűre csökkentik.

algebrai formális specifikáció A matematikai rendszer-specifikáció egyik módszere, amelyben egy rendszert vagy komponenst a biztosítandó interfészében megadott műveletei közötti kapcsolatokkal definiálnak.

algoritmikus költségmodellezés A szoftverkötségek becslésének olyan megközelítése, amelyben egy formulát használnak a projekt költségének előrejelzésére. A formula paraméterei a projekt és a szoftver bizonyos tulajdonságai.

alkalmazáscsalád Alkalmazások halmaza, amelyek közös architektúrával és általános funkcionalitással rendelkeznek. A komponensek és a programok paramétereinek módosításával ezek az ügyfelek speciális igényeinek megfelelően személyre szabhatók.

alkalmazási keretrendszer Egy specifikus terület valamilyen generikus struktúrája, amely alkalmazások egy családjának alapjául szolgálhat. Az alkalmazási keretrendszereket általában konkrét és absztrakt osztályok együtteseként implementálják, amelyek később specializálhatók és példányosíthatók, így alkalmazások készíthetők belőlük.

alkalmazásprogramozói interfész (Application Program Interface, API) Egy interfész (általában művelethalmazként specifikálják), amelyet egy alkalmazás definiál, hogy hozzáférést biztosítson saját funkcionalitásához. Ez azt jelenti, hogy a funkcionalitást közvetlenül is használhatják más programok, nem csak egy felhasználói interfészen keresztül.

aspektusorientált programozás Programozási módszer, amely kombinálja a generatív és a komponensalapú fejlesztést. A programban keresztező vonatkozásokat azonosítanak, és ezek implementációját aspektusokként definiálják. Ezeket az aspektusokat később egy programszövő építi a programba a megfelelő helyeken.

biztonságosság A rendszer katasztrofális kimenetelű hibák nélküli működése.

biztonságossági tanulmány A rendszer biztonságosságának strukturált bizonyítása. Általában felügyeleti szervezetek számára készül, mint például a nukleáris biztonságot felügyelő szervezetek.

C Programozási nyelv, amelyet eredetileg a Unix-rendszer implementációjához fejlesztettek ki. A C viszonylag alacsony szintű programozási nyelv: biztosítja a rendszer hardvereinek elérését, valamint hatékony kód fordítható belőle. Ma is széles körben használják alacsony szintű rendszerprogramozásra.

C++ Objektorientált programozási nyelv, a C kibővített változata.

CASE (Computer-Aided Software Engineering) Számítógéppel segített szoftvertervezés. A szoftverek automatizált támogatással történő fejlesztésének folyamata.

CASE-eszköz Eszköz, amelyet a szoftverfejlesztési folyamat valamely tevékenységének támogatására használnak, például tervszerkesztők és hibakeresők.

CASE-eszközkészlet CASE-eszközök integrált halmaza, amely eszközök együtt dolgoznak valamilyen fontos fejlesztési folyamat (például szoftvertervezés vagy konfigurációkezelés) támogatásában.

Cleanroom-szoftverfejlesztés Szoftverfejlesztési megközelítés, amelynek célja, hogy a készülő szoftverbe ne kerüljenek hibák (az elnevezés a félvezetőgyártás hasonló nevű módszerétől ered). A folyamat tartalmazza a szoftver formális specifikációját, a specifikáció strukturált transzformációját programmá, helyességbizonyítások kifejlesztését és statisztikai programtesztelést.

CMMI A folyamatérettség-modellezés integrált megközelítése.

COCOMO (CONstructive COst MODELing) Konstruktív költségmodellezés. Talán a legismertebb algoritmikus költségbecslési modell.

COM+ A Microsoft-platformokra kifejlesztett komponensmodell.

CORBA (Common Object Request Broker Architecture) Közös objektumkérés-közvetítő architektúra. Az OMG által készített szabványok gyűjteménye, amelyek egy elosztott objektummodellt definiálnak, valamint meghatározzák az objektumok kommunikációját.

CORBA-komponensmodell A CORBA-platformhoz kifejlesztett komponensmodell.

elosztott objektumrendszer Elosztott rendszer, amelyben a végrehajtandó komponensek objektumok.

elosztott rendszer Olyan szoftverrendszer, amelyben az alrendszerek és komponensek több különböző processzoron futnak.

Enterprise Java Beans (EJB) Java-alapú komponensmodell.

eseményalapú rendszerek Olyan rendszerek, amelyekben a műveletek végrehajtásának vezérlését a rendszer környezetében generált események határozzák meg. A legtöbb valós idejű rendszer eseményalapú.

etikai és szakmai gyakorlati kódex Vezérelvek gyűjteménye, amelyben megtalálhatjuk a szoftverfejlesztők etikai és szakmai viselkedésével szemben támasztott elvárásokat. Ezeket a legnagyobb amerikai szakmai társaságok (az ACM és az IEEE) határozták meg, és nyolc címszó alatt definiálják az etikai viselkedés normáit.

etnográfia Megfigyelésalapú technika, amely használható a követelmények feltárásában és elemzésében. Az etnográfiát készítő személy beépül a felhasználók környezetébe, és megfigyeli mindennapi munkavégzésüket. A szoftverek követelményei aztán kikövetkeztethetők ezekből a megfigyelésekből.

extrém programozás Egy agilis módszer, amely olyan elemeket tartalmaz, mint a forgatókönyv-alapú követelményfeltárás, az előrehozott teszteléssel történő fejlesztés és a páros programozás.

felhasználói felület tervezése Annak megtervezése, hogy a felhasználók hogyan érhetik el a rendszer funkcionalitását, valamint hogy a rendszer által előállított információk hogyan jelennek meg a képernyőn.

felhasználói felületek tervezésének alapelvei Olyan alapelvek gyűjteménye, amelyek a felhasználói felületek tervezésével kapcsolatos legjobb gyakorlatokat tükrözik.

folyamat továbbfejlesztése Az a tevékenység, melynek során megváltoztatunk egy folyamatot azzal a céllal, hogy előrejelezhetőbbé tegyük azt, vagy javítsunk végeredményeinek minőségén. Például ha azt szeretnénk, hogy csökkenjen a legyártott szoftverekben előforduló hibák száma, akkor továbbfejleszthetjük a folyamatot úgy, hogy új validációs tevékenységeket adunk hozzá.

folyamatfejlettségi modell Azt modellezi, hogy a folyamat mennyire teszi magáévá a bevált módszereket, valamint metrikákat tartalmaz, amelyek segítségével a folyamatok továbbfejleszthetők.

folyamatmodell Egy folyamat absztrakt reprezentációja. Folyamatmodellek több nézőpontból is fejleszthetők, és a különböző nézőpontok modelljei tartalmazhatják a folyamatban részt vevő elemeket, a folyamat résztvékenységeit, azokat a megszorításokat, amelyek teljesülnek a folyamatra, valamint a folyamatban részt vevő személyek szerepkörét.

forгатókönyv Leír egy tipikus módot, ahogyan a felhasználók használják a rendszert, vagy ahogyan egy felhasználó végrehajt valamilyen tevékenységet.

formális módszerek A szoftverfejlesztés olyan módszerei, amelyek matematikai szempontból szigorú elméleteken alapulnak, és amelyek formális matematikai konstrukciók (például predikátumok és halmazok) segítségével modellezik a szoftvereket.

funkcionális követelmény Néhány olyan funkció és eszköz felsorolása, amelyeket a rendszernek implementálnia kell.

Gantt-táblázat Projektmenedzserek által használt diagram, amelyen megjelennek a projekt feladatai, az ezekhez tartozó ütemezések és a rajtuk dolgozó emberek. Egy időskála segítségével szemléltethetők rajta a feladatok kezdetének és befejezésének időpontjai, valamint az emberi erőforrások allokációja.

gyors alkalmazásfejlesztés (rapid application development, RAD) A szoftverek gyors legyártását célzó szoftverfejlesztési módszertan. Gyakran alkalmaz-

nak benne adatbázis-programozást és fejlesztést támogató eszközöket, mint amilyenek a képernyő- és lekérdezésgenerátorok.

használati eset A rendszerrel történő interakciók egy típusának specifikációja.

hibaelkerülés A szoftverfejlesztés olyan módja, amelyben nem kerülnek hibák a fejlesztendő szoftverbe.

hibakeresés A programban található hibák felderítése és eltávolítása folyamatok és futási idejű ellenőrzés segítségével, mielőtt azok a rendszer meghibásodásához vezethetnének.

hibatűrés A rendszerek azon képessége, hogy működésüket akkor is folytatni tudják, ha hibák következnek be.

információrejtés Az adatstruktúrák reprezentációjának elrejtése, és az ezen adatstruktúrákhoz történő hozzáférés szabályozása programnyelvi eszközökkel.

inkrementális fejlesztés Szoftverfejlesztési megközelítés, amelyben a szoftvert inkrementekben fejlesztik és helyezik üzembe.

interfész Egy szoftverkomponens műveleteinek és attribútumainak specifikációja. Az interfészen keresztül érhető el a komponens funkcionalitása.

ISO 9000 A Nemzetközi Szabványügyi Hivatal (ISO) által definiált szabvány a minőségkezelési folyamatokra.

iteratív fejlesztés Szoftverfejlesztési megközelítés, amelyben a specifikáció, a tervezés, a programozás és a tesztelés folyamatai ismétlődnek.

Java Objektorientált programozási nyelv, amelyet a Sun¹ fejlesztett ki a platformfüggetlenség támogatására.

karbantartás A rendszer üzembe helyezés utáni változtatásának folyamata.

kiadás Szoftverrendszer verziója, amely elérhető a rendszer ügyfelei számára.

kliens-szerver architektúra Az elosztott rendszerek architekturális modellje, amelyben a rendszerfunkcionalitást egy szerver által nyújtott szolgáltatások jelentik. Ezeket kliensszámítógépek érik el, amelyek felhasználják ezt a funkcionalitást. A modell bizonyos változatai, mint a háromrétegű kliens-szerver modell, több szervert alkalmaznak.

kockázat Nemkívánatos kimenetel, amely akadályt gördíthet bizonyos célok elérése elé. A folyamatkockázat fenyegeti a folyamatok ütemezését és költségét; a termékkockázat olyan kockázat, amely meghiúsíthatja néhány rendszerkövetelmény megvalósítását.

kockázatkezelés Az a folyamat, amelynek során azonosítjuk a kockázati tényezőket, felbecsüljük komolyságukat, metrikákat tervezünk arra az esetre, ha a kockázati események bekövetkeznének, valamint felügyeljük a szoftvert és a szoftverfolyamatot a kockázatok figyelése érdekében.

komponens A szoftver használható, független egysége, amely teljesen definiálva van, és interfészeken keresztül érhető el.

komponensalapú szoftverfejlesztés (Component-Based Software Engineering, CBSE) A szoftverek független, működőképes komponensek kompozíciójával történő fejlesztése.

komponensmodell A komponensek implementációját, dokumentációját és üzembe helyezését meghatározó szabványok gyűjteménye. Ezek specifikálják azokat az interfészeket, amelyeket a komponenseknek biztosítaniuk kell;

meghatározzák a komponensek elnevezésének és együttműködésének szabályait, továbbá azt, hogyan történik a komponensek kompozíciója. A komponensmodellek biztosítják azt, hogy a köztes szoftver képes támogatni a komponensek végrehajtását.

konfigurációkezelés Egy fejlődő szoftver változásainak kezelését megvalósító folyamat. A konfigurációkezelés magában foglalja a konfigurációtervezést, a verziókezelést, a rendszer felépítését és a változáskezelést.

konfigurációs tétel Egy számítógépek számára olvasható egység (például dokumentum vagy forráskódállomány), amelyet változtatni készülünk, és a változtatást egy konfigurációkezelő rendszernek kell irányítania.

követelmények kezelése A követelmények megváltozásának kezelése. A folyamatnak biztosítania kell, hogy a változtatások pontos elemzése megtörtént, és hogy megvalósul a változtatások megfelelő nyomon követése a rendszeren keresztül.

köztes szoftver Egy elosztott rendszer infrastruktúrát megvalósító szoftvere. Segít kezelni a rendszer egyedei és adatbázisai közötti interakciókat. Példák köztes szoftverre: objektumkérés-kezelők, tranzakciókezelő rendszerek.

kritikus rendszer Olyan számítógépes rendszer, amelynek meghibásodása jelentős gazdasági, emberi vagy környezeti veszteségeket okozhat.

megbízhatóság növekedésének modellezése Olyan modell kifejlesztése, amely tükrözi, hogyan változik a megbízhatóság (remélhetőleg növekszik) a tesztek és a program hiányosságainak megszüntetése után.

megbízhatóság A rendszer képessége, hogy szolgáltatásait a specifikációnak megfelelően biztosítsa. A megbízhatóság kétféleképpen adható meg: annak valószínűségéként, hogy a rendszer hibázik, valamint a hibák előfordulásának számával.

minőségbiztosítás Az az átfogó folyamat, amelyben meghatározzák, hogyan teljesíthető az elvárt szoftverminőség, valamint hogyan tudja megállapítani a fejlesztő szervezet, hogy az elkészült szoftver megfelelő minőségű.

minőség-ellenőrzés Az a folyamat, amelyben bizonyosságot nyer, hogy a szoftvert fejlesztő csapat betartja a minőségi szabványokban foglaltakat.

minőségi terv Ez a terv definiálja a használandó minőségi folyamatokat és eljárásokat. Ebbe beletartozik a termékek minőségi szabványainak kiválasztása, valamint a rendszer elvárt minőségi jellemzőinek definiálása.

modellalapú formális specifikáció A matematikai rendszer-specifikáció olyan módszere, ahol egy rendszert vagy egy komponenst úgy specifikálnak, hogy megadják azokat az előfeltételeket, utófeltételeket és invariánsokat, amelyeknek teljesülniük kell a rendszer állapotára.

nemfunkcionális követelmény Egy megszorításról vagy valamilyen elvárt viselkedésről szóló kijelentés, amelynek teljesülnie kell a rendszerre. Ez a megszorítás vonatkozhat a fejlesztendő rendszer tulajdonságaira vagy magára a fejlesztési folyamatra.

nyelvfeldolgozó rendszer Olyan rendszer, amely egy nyelvről fordít egy másikra. Például egy fordító olyan nyelvfeldolgozó rendszer, amely a forráskódot tárgykóddá fordítja.

objektummodell Olyan szoftverrendszer modellje, amely objektumosztályokból és az ezen osztályok közötti kapcsolatokból épül fel. A modellnek számos különböző nézőpontból készített változata létezhet: állapotmodell, szekvenciadiagram stb.

objektumorientált fejlesztés Szoftverfejlesztési megközelítés, amelyben a rendszerben megjelenő alapvető absztrakciók egymástól független objektumok. Ugyanilyen típusú absztrakció használható a specifikáció, a tervezés és a fejlesztés alatt.

objektumosztály Egy objektumosztály objektumok attribútumait és műveleteit definiálja. Az objektumok futási időben jönnek létre az osztálydefiníció példányosításával. Néhány objektumorientált rendszerben az objektumosztály neve típusnévként használható.

OCL (Object Constraint Language) Az UML részét képező nyelv, amelyet olyan predikátumok megadásához használnak, amelyek teljesülnek egy adott UML-modell osztályaira és interakcióira.

OMG (Object Management Group) Vállalatcsoport, amely azzal a céllal jött létre, hogy megalkossa az objektumorientált fejlesztés szabványait. Az OMG által kiadott szabványok például a CORBA, az UML és az MDA.

„Óz, a nagy varázsló” típusú prototípus-készítés Prototípus-készítési módszer, amelyben a felhasználó által begépelte parancsokat egy ember értelmezi, aki úgy reagál ezekre a parancsokra, mint egy számítógép.

ősrendszer Elavult technológiákkal és módszerekkel fejlesztett szociotechnikai rendszer, amely azonban hasznos, esetleg pótolhatatlan egy szervezet számára. Mivel az ősrendszerek gyakran kritikus üzleti funkciókat valósítanak meg, karbantartásuk elengedhetetlen.

peer-to-peer rendszer Olyan elosztott rendszer, ahol nem különböztetnek meg klienseket és szervereket. A rendszer számítógépei egyaránt betölthetnek szerver és kliens szerepet is. A peer-to-peer alkalmazások közé tartoznak az állománymegosztó alkalmazások, az azonnali üzenetküldés és az együttműködést segítő rendszerek.

People Capability Maturity Model (P-CMM) Folyamatfejlettségi modell, amely azt vizsgálja, milyen hatékony egy szervezet alkalmazottai képességeinek, oktatásának és tapasztalatának kezelésében.

programátvizsgálás Verifikációs folyamat, amelyben egy csoport soronként átvizsgál egy programot azzal a céllal, hogy hibákat fedezzen fel benne.

programgenerátor Olyan program, amely programokat állít elő magas szintű, absztrakt specifikációkból. A generátor olyan tudást tartalmaz, amelyet minden egyes generálásnál újrafelhasznál.

Rational Unified Process (RUP) Általános szoftverfolyamat-modell, amely a szoftverfejlesztést négyfázisú iteratív tevékenységnek tekinti, ahol a négy fázis: indítás, előkészítés, létrehozás és átadás. Az indítás megteremti a rendszer üzleti hátterét; az előkészítés definiálja az architektúrát; a létrehozás implementálja a rendszert; az átadás pedig üzembe helyezi a rendszert az ügyfél környezetében.

- referenciaarchitektúra** Egy általános, idealizált rendszer-architektúra, amely tartalmaz minden olyan eszközt és funkcionalitást, amelyeknek a rendszerekben meg kellene jelenniük. A tervezők a referenciaarchitektúrákon keresztül ismerhetik meg a rendszerek egy adott osztályának általános architektúráját.
- rejtett tulajdonság** Olyan tulajdonság, amely csak akkor válik láthatóvá, ha minden komponenst integráltunk a rendszerbe.
- rendelkezésre állás** Azt mutatja meg, hogy a rendszer milyen mértékben képes kiszolgálni a tőle szolgáltatást igénylőket. Általában decimális számként adják meg: a 0,999-es rendelkezésre állás azt jelenti, hogy a rendszer 1000 időegységből 999-ben képes szolgáltatáskérések teljesítésére.
- rendszerfelépítés** Folyamat, amelyben lefordítjuk a rendszert alkotó komponenseket és egységeket, majd összeszerkesztjük ezeket más komponensekkel, így elkészítve egy végrehajtható programot. A rendszerfelépítés általában automatizált, így az újrafordítás minimális. Ez az automatizáció be lehet építve a nyelvfeldolgozó rendszerbe (mint a Javában), vagy megvalósítható CASE-eszközök segítségével.
- rendszertervezés** Olyan folyamat, amelynek célja egy rendszer specifikálása, a rendszer komponenseinek integrálása és a rendszer tesztelése úgy, hogy az eredmény megfeleljen a vele szemben támasztott követelményeknek. A rendszertervezés az egész szociotechnikai rendszerrel foglalkozik (szoftver, hardver, működési folyamatok), nem csupán a rendszerszoftverrel.
- spirális modell** A fejlesztési folyamat modellje, amelyben a folyamatot egy spirál reprezentálja, és a spirál minden köre a folyamat egy újabb állomását jelöli. Ahogy az egyik spirálfordulatról a másikra haladunk tovább, a folyamat minden lépését megismételjük.
- SQL (Structured Query Language)** A relációs adatbázisok programozásához használt szabványos nyelv.
- statikus elemzés** Egy program forráskódjának eszközalapú elemzése, amelynek célja a hibák és az anomáliák felderítése. Az anomáliák, például egy változónak egymás utáni értékadások a változó közbeeső felhasználása nélkül, általában programozási hibák.
- strukturált módszer** Szoftvertervezési módszer, amely definiálja, milyen rendszermodellek kifejlesztése szükséges, milyen szabályoknak kell teljesülniük ezekre a modellekre, és meghatározza, milyen folyamatot kell követni a terv kialakítása során.
- szakterület** Speciális üzleti terület, amelyben szoftverrendszereket alkalmaznak, például a valós idejű rendszerek, az üzleti adatfeldolgozás stb.
- szakterületi modell** Szakterületi absztrakciók specifikációja. Ilyen absztrakciók lehetnek a politika, eljárások, objektumok, kapcsolatok, események stb. Valamely szakterületről szóló tudás alapjául szolgál.
- szekvenciadiagram** Olyan diagram, amelyen egy művelet végrehajtásához szükséges interakciók sorozata szemléltethető. Az UML-ben a szekvenciadiagramokhoz használati esetek rendelhetők.
- szerver** Olyan program, amely szolgáltatásokat kínál más programok (kliensek) számára.

- szociotechnikai rendszer** Olyan rendszer, amelynek vannak hardver- és szoftverkomponensei, meghatározott működési folyamatokkal rendelkezik, emberek használják és egy szervezeten belül működik. Fenti tulajdonságaiból következően hatással van rá a szervezeti politika, a szervezet eljárásai és struktúrája.
- szoftverarchitektúra** Egy szoftverrendszer alapvető struktúrájának és szerveződéskének modellje.
- szoftveréletciklus** Gyakran a szoftverfolyamat szinonimájaként használják, eredetileg azonban a szoftverfolyamat vízesésmoddelljének elnevezésére alkották meg a kifejezést.
- szoftverfolyamat** Folyamatok és tevékenységek olyan összetartozó halmaza, amelyek részt kapnak egy szoftverrendszer kialakításában és fejlesztésében.
- szoftvermetrika** Egy szoftverrendszer vagy folyamat olyan tulajdonsága, amely számokkal kifejezhető és mérhető. Az egy feladat befejezéséhez szükséges idő olyan metrika, amely egy folyamatra vonatkozik, míg a szoftver mérete és összetettsége magának a szoftvernek a tulajdonsága.
- szoftvertermékvonal** *Lásd* alkalmazáscsalád.
- tervezési minta** Egy gyakran felvetődő probléma jól bevált megoldása, amely tapasztalatot és kipróbált gyakorlatot épít magába, továbbá újrafelhasználható. Egy tervezési minta egy absztrakt reprezentáció, amely számos különböző módon példányosítható.
- tevékenység (PERT) táblázat** Projektmenedzserek által a végrehajtandó feladatok közötti függőségek szemléltetésére használt táblázat. Tartalmazza a feladatokat, végrehajtásuk várható idejét és a köztük fennálló függőségeket. *A kritikus út* a tevékenységtáblázaton keresztül vezető (a feladatok végrehajtási idejében mérve) leghosszabb út.
- tranzakció** A számítógéppel történő interakciók egysége. A tranzakciók függetlenek és atomiak (nem bonthatók kisebb elemekre), és a helyreállítás, a konzisztencia és a konkurencia alapvető egységét képezik.
- tranzakciófeldolgozó rendszer** Olyan rendszer, amely azért felelős, hogy a tranzakciók olyan módon kerüljenek feldolgozásra, hogy azok ne zavarják egymás működését, továbbá egy tranzakció meghibásodása ne legyen hatással semelyik másik tranzakcióra és a rendszer adataira.
- újratervezés** A rendszer módosítása annak érdekében, hogy érthetőbb és könnyebben változtatható legyen. Az újratervezés gyakran maga után vonja a szoftverek és az adatok átszervezését, a programok egyszerűsítését és a dokumentáció megújítását.
- UML (Unified Modeling Language)** Grafikus nyelv, amelyet az objektumorientált fejlesztésben alkalmaznak, amely számos különböző rendszermodell kidolgozását magában foglalja, amelyek a rendszer különböző nézeteit mutatják. Az UML az objektumorientált modellezés *de facto* szabványává vált.
- üzembiztonság** A rendszer üzembiztonsága összetett tulajdonság, amelyet a rendszer biztonságossága, biztonsága, rendelkezésre állása, védelmi szintje és más tulajdonságok együttesen határoznak meg. A rendszer üzembiztonsága azt tükrözi, hogy a felhasználók mennyire bízhatnak meg benne.

- üzembiztonsági követelmény** Olyan rendszerkövetelmény, amelynek az a célja, hogy segítsen megvalósítani a rendszer elvárt üzembiztonsági szintjét. A nem-funkcionális üzembiztonsági követelmények specifikálják az üzembiztonsági jellemzők értékeit; a funkcionális üzembiztonsági követelmények például azt határozzák meg, hogyan kell viselkednie a rendszernek meghibásodás esetén.
- üzembiztonsági tanulmány** Strukturált dokumentum, amelyet arra használnak, hogy rögzítsék a fejlesztők üzembiztonsággal kapcsolatos állításait.
- üzleti folyamat újratervezése** Üzleti folyamat megváltoztatása annak érdekében, hogy az megfeleljen néhány új szervezeti célnak (például alacsonyabb költségek, gyorsabb végrehajtás stb.).
- validáció** Az a folyamat, amelyben megerősítik, hogy a rendszer megfelel az ügyfél igényeinek és elvárásainak.
- valós idejű rendszer** Olyan rendszer, amelynek valós időben kell eseményeket feldolgoznia és válaszolnia azokra. A rendszer helyességét nemcsak az határozza meg, hogy mit csinál a rendszer, hanem az is, milyen gyorsan teszi a dolgát. A valós idejű rendszereket általában szekvenciális folyamatok együttműködő halmazaként implementálják.
- védelem** A rendszer azon tulajdonsága, hogy képes megvédeni magát a véletlen vagy szándékos behatolásokkal szemben.
- verifikáció** Az a folyamat, amelyben ellenőrzik, hogy a rendszer megfelel specifikációjának.
- vízesésmodell** Olyan szoftverfolyamat-modell, amely diszkrét fejlesztési szakaszokat tartalmaz. Ezek: specifikáció, tervezés, implementáció, tesztelés és karbantartás. Elméletben minden szakasznak be kell fejeződnie, mielőtt elkezdődik a következő. A gyakorlatban azonban ezt nem tartják be, és a szakaszokon iteratívan haladnak végig.
- webszolgáltatás** Független szoftverkomponens, amely szabványos protokollok segítségével érhető el az interneten keresztül. A webszolgáltatások a SOAP (Standard Object Access Protocol) protokollt használják információcserére, míg a webszolgáltatások interfészének definiálására a WSDL (Web Service Definition Language) protokollt alkalmazzák.
- XML (Extended Markup Language)** Szöveges jelölőnyelv, amely támogatja a strukturált információ cseréjét. Minden adatmezőt tagok határolnak. Ezek a tagok szolgálnak információval a mezővel kapcsolatban. Az XML-t manapság elterjedten használják, és mára a webszolgáltatási protokollok alapjává vált.
- Z** Egy modellalapú, formális specifikációra szolgáló nyelv, amelyet az angliai Oxford University fejlesztettek ki.

IRODALOMJEGYZÉK

- Abbott, R. (1983). Program design by informal English descriptions. *Comm. ACM*, 26(11), 882–94. (14. fejezet)
- Abdel-Ghaly, A. A., Chan, P. Y., et al. (1986). Evaluation of competing software reliability predictions. *IEEE Trans. on Software Engineering*, SE-12(9), 950–67. (24. fejezet)
- Ackroyd, S., Harper, R., et al. (1992). *Information Technology and Practical Police Work*. Milton Keynes: Open University Press. (2. fejezet)
- Adams, E. N. (1984). Optimizing preventative service of software products. *IBM J. Res. & Dev.*, 28(1), 2–14. (3. fejezet)
- Ahern, D. M., Clouse, A., et al. (2001). *CMMI Distilled*. Reading, MA: Addison-Wesley. (28., 29. fejezet)
- Albrecht, A. J. (1979). Measuring application development productivity. *Proc. SHARE/GUIDE IBM Application Development Symposium*. (26. fejezet)
- Albrecht, A. J.–Gaffney, J. E. (1983). Software function, lines of code and development effort prediction: a software science validation. *IEEE Trans. on Software Engineering*, SE-9(6), 639–47. (26. fejezet)
- Alexander, C., Ishikawa, S., et al. (1977). *A Pattern Language*. Oxford: Oxford University Press. (18. fejezet)
- Ambler, S. W.–Jeffries, R. (2002). *Agile Modeling*. New York: John Wiley & Sons. (17. fejezet)
- Anderson, R. (2001). *Security Engineering: A Guide to Building Dependable Distributed Systems*. Chichester: John Wiley & Sons. (24. fejezet)
- Appelrath, H.-J.–Ritter, J. (2000). *SAP R/3 Implementation: Methods and Tools (SAP Excellence)*. Berlin: Springer-Verlag. (13. fejezet)
- Armour, P. (2002). Ten unmyths of project estimation. *Comm. ACM*, 45(11), 15–18. (26. fejezet)
- Aron, J. D. (1974). *The Program Development Process*. Reading, MA: Addison-Wesley. (25. fejezet)
- Arthur, L. J. (1988). *Software Evolution*. New York: John Wiley & Sons. (21. fejezet)
- Avizienis, A. (1985). The N-version approach to fault-tolerant software. *IEEE Trans. on Software Engineering*, SE-11(12), 1491–501. (20. fejezet)
- Avizienis, A. A. (1995). A methodology of N-version programming. In *Software Fault Tolerance* (M. R. Lyu, ed.). New York: John Wiley & Sons, 23–46. (20. fejezet)
- Bagert, D. J. (2002). Texas licensing of software engineers: all's quiet for now. *Comm. ACM*, 45(11), 92–4. (24. fejezet)
- Baker, F. T. (1972). Chief programmer team management of production programming. *IBM Systems J.*, 11(1), 56–73. (25. fejezet)
- Baker, T. (2002). Lessons learned integrating COTS into systems. *Proc. ICCBSS 2002 (1st Int. Conf on COTS-based Software Systems)*, Orlando, FL: Springer-Verlag. (18. fejezet)
- Balk, L. D.–Kedia, A. (2000). PPT: a COTS integration case study. *Proc. Int. Conf. on Software Engineering*, Limerick, Ireland: ACM Press. (18. fejezet)
- Bamford, R.–Deibler, W. J., eds. (2003). *ISO 9001: 2000 for Software and Systems Providers: An Engineering Approach*. CRC Press. (27. fejezet)
- Banker, R. D., Datar, S. M., et al. (1993). Software complexity and maintenance costs. *Comm. ACM*, 36(11), 81–94. (21., 26. fejezet)

- Banker, R., Kauffman, R., *et al.* (1994). An empirical test of object-based output measurement metrics in a computer-aided software engineering (CASE) environment. *J. of Management Info. Sys.*, **8**(3), 127–50. (26. fejezet)
- Bansler, J. P.–Bødker, K. (1993). A reappraisal of structured analysis: design in an organizational context. *ACM Trans. on Information Systems*, **11**(2), 165–93. (4. fejezet)
- Barker, R. (1989). *CASE* Method: Entity Relationship Modelling*. Wokingham: Addison-Wesley. (8. fejezet)
- Barnard, J.–Price, A. (1994). Managing code inspection information. *IEEE Software*, **11**(2), 59–69. (22., 27. fejezet)
- Basili, V. R.–Rombach, H. D. (1988). The TAME project: towards improvement-oriented software environments. *IEEE Trans. on Software Engineering*, **14**(6), 758–73. (27., 28. fejezet)
- Basili, V.–Green, S. (1993). Software process improvement at the SEL. *IEEE Software*, **11**(4), 58–66. (28. fejezet)
- Bass, B. M.–Dunteman, G. (1963). Behaviour in groups as a function of self, interaction and task orientation. *J. Abnorm. Soc. Psychology*, **66**(4), 19–28. (25. fejezet)
- Bass, L., Clements, P., *et al.* (2003). *Software Architecture in Practice*, 2nd edn. Boston: Addison-Wesley. (11. fejezet)
- Baumer, D., Gryczan, G., *et al.* (1997). Framework development for large systems. *Comm. ACM*, **40**(10), 52–9. (18. fejezet)
- Beck, K. (1999). Embracing change with extreme programming. *IEEE Computer*, **32**(10), 70–8. (6., 17. fejezet)
- Beck, K. (2000). *Extreme Programming Explained*. Boston: Addison-Wesley. (4., 17., 25., 26. fejezet)
- Beck, K.–Cunningham, W. (1989). A laboratory for teaching object-oriented thinking. *Proc. OOPSLA'89*, New Orleans: ACM Press. (14. fejezet)
- Bentley, R., Rodden, T., *et al.* (1992). Ethnographically informed systems design for air traffic control. *Proc. CSCW'92*, Toronto: ACM Press. (16. fejezet)
- Berczuk, S. P.–Appleton, B. (2002). *Software Configuration Management Patterns: Effective Teamwork, Practical Integration*. Boston: Addison-Wesley. (18. fejezet)
- Berghel, H. (2001). The code red worm. *Comm. ACM*, **44**(12), 15–19. (3. fejezet)
- Berliner, B. (1990). CVS II: parallelizing software development. *Proc. 1990 Winter USENIX Conference*, Washington, DC: USENIX Assoc. (29. fejezet)
- Bernstein, P. A. (1996). Middleware: a model for distributed system services. *Comm. ACM*, **39**(2), 86–97. (12. fejezet)
- Bersoff, E. H.–Davis, A. M. (1991). Impact of life cycle models on software configuration management. *Comm. ACM*, **34**(8), 104–18. (29. fejezet)
- Bezier, B. (1990). *Software Testing Techniques*, 2nd edn. New York: Van Nostrand Reinhold. (23. fejezet)
- Biggerstaff, T. (1998). A perspective of generative reuse. *Annals of Software Engineering*, **5**, 169–226. (18. fejezet)
- Bishop, P.–Bloomfield, R. E. (1995). The SHIP safety case approach. *Proc. Safecomp'95*, Belgrade, Italy: Springer-Verlag. (24. fejezet)
- Bishop, P.–Bloomfield, R. E. (1998). A methodology for safety case development. *Proc. Safety-critical Systems Symposium*, Birmingham, UK: Springer-Verlag. (24. fejezet)
- Blevins, D. (2001). Overview of the Enterprise Java Beans component model. In *Component-Based Software Engineering* (G. T. Heineman–W. T. Council, eds.). Boston: Addison-Wesley, 589–606 (19. fejezet)
- Boehm, B. (1997). *COCOMO II Model Definition Manual*. Center for Software Engineering, Univ. of Southern California (<http://sunset.usc.edu/research/COCOMOII>). (26. fejezet)

- Boehm, B. W. (1979). Software engineering; R & D trends and defense needs. In *Research Directions in Software Technology* (P. Wegner, ed.). Cambridge, MA: MIT Press, 1–9. (22. fejezet)
- Boehm, B. W. (1981). *Software Engineering Economics*. Englewood Cliffs, NJ: Prentice Hall. (26. fejezet)
- Boehm, B. W. (1988). A spiral model of software development and enhancement. *IEEE Computer*, **21**(5), 61–72. (4., 5. fejezet)
- Boehm, B. W., Abts, C., et al. (2000). *Software Cost Estimation with COCOMO II*. Upper Saddle River, NJ: Prentice Hall. (19., 22., 26. fejezet)
- Boehm, B. W., McClean, R. L., et al. (1975). Some experience with automated aids to the design of large-scale reliable software. *IEEE Trans. on Software Engineering*, **SE-1**(1), 125–33. (3. fejezet)
- Boehm, B., Clark, B., et al. (1995). Cost models for future life cycle processes: COCOMO 2. *Annals of Software Engineering*, **1**, 57–94. (26. fejezet)
- Boehm, B.–Abts, C. (1999). COTS integration: plug and pray? *IEEE Computer*, **32**(1), 135–8. (18. fejezet)
- Boehm, B.–Royce, W. (1989). Ada COCOMO and the Ada process model. *Proc. 5th COCOMO Users' Group Meeting*, Pittsburgh: Software Engineering Institute. (26. fejezet)
- Bolognesi, T.–Brinksma, E. (1987). Introduction to the ISO specification language LOTOS. *Computer Networks*, **14**(1), 25–59. (10. fejezet)
- Booch, G. (1987). *Software Components with Ada: Structures, Tools, and Subsystems*. Menlo Park, CA: Benjamin-Cummings. (18. fejezet)
- Booch, G. (1994). *Object-Oriented Analysis and Design with Applications*. Redwood City, CA: Benjamin-Cummings. (1., 4., 8., 14. fejezet)
- Booch, G., Rumbaugh, J., et al. (1999). *The Unified Modeling Language User Guide*. Reading, MA: Addison-Wesley. (1., 4., 8. fejezet)
- Borchers, J. (2001). *A Pattern Approach to Interaction Design*. New York: John Wiley & Sons. (18. fejezet)
- Bosch, J. (2000). *Design and Use of Software Architectures*. Harlow: Addison-Wesley. (11. fejezet)
- Bourne, S. R. (1978). The Unix shell. *Bell Sys. Tech. J.*, **57**(6), 1971–90. (17. fejezet)
- Bracket, M. H. (1994). *Data Sharing Using a Common Data Architecture*. New York: John Wiley & Sons. (13. fejezet)
- Brazendale, J.–Bell, R. (1994). Safety-related control and protection systems: standards update. *IEEE Computing and Control Engineering J.*, **5**(1), 6–12. (9. fejezet)
- Brilliant, S. S., Knight, J. C., et al. (1990). Analysis of faults in an N-version software experiment. *IEEE Trans. on Software Engineering*, **16**(2), 238–47. (20. fejezet)
- Brinch-Hansen, P. (1973). *Operating System Principles*. Englewood Cliffs, NJ: Prentice Hall. (15. fejezet)
- Brooks F. P. (1975). *The Mythical Man Month*. Reading, MA: Addison-Wesley. (25. fejezet)
- Brown, A. W., Earl, A. N., et al. (1992). *Software Engineering Environments*. London: McGraw-Hill. (11. fejezet)
- Budgen, D. (2003). *Software Design, 2nd edn*. Harlow: Addison-Wesley. (8. fejezet)
- Burns, A.–Wellings, A. (2001). *Real-Time Systems and Programming Languages*. Harlow: Addison-Wesley. (15. fejezet)
- Butler, H. (1994). Virtual remote: the centralized expert. *HP Journal*, **45**(5), <http://www.hpl.hp.com/hpjournal/94oct/oct94a13.htm>. (18. fejezet)
- Buxton, J. (1980). *Requirements for Ada Programming Support Environments: Stoneman*. Washington, DC: US Department of Defense. (11. fejezet)

- Checkland, P. (1981). *Systems Thinking, Systems Practice*. Chichester: John Wiley & Sons. (2. fejezet)
- Checkland, P.–Scholes, J. (1990). *Soft Systems Methodology in Action*. Chichester: John Wiley & Sons. (2. fejezet)
- Chen, P. (1976). The entity relationship model—towards a unified view of data. *ACM Trans. on Database Systems*, 1(1), 9–36. (8. fejezet)
- Chidamber, S.–Kemerer, C. (1994). A metrics suite for object-oriented design. *IEEE Trans. on Software Engineering*, 20(6), 476–93. (27. fejezet)
- Chikofsky, E. J.–Cross, J. H. (1990). Reverse engineering and design recovery: a taxonomy. *IEEE Software*, 7(1), 13–17. (21. fejezet)
- Clements, P., Bachmann, F., et al. (2002). *Documenting Software Architectures: Views and Beyond*. Boston: Addison-Wesley. (11. fejezet)
- Coad, P.–Yourdon, E. (1990). *Object-Oriented Analysis*. Englewood Cliffs, NJ: Prentice Hall. (8., 14. fejezet)
- Cobb, R. H.–Mills, H. D. (1990). Engineering software under statistical quality control. *IEEE Software*, 7(6), 44–54. (22. fejezet)
- Cockburn, A. (2001). *Agile Software Development*. Reading, MA: Addison-Wesley. (17. fejezet)
- Codd, E. F. (1979). Extending the database relational model to capture more meaning. *ACM Trans. on Database Systems*, 4(4), 397–434. (8. fejezet)
- Cohen, B., Harwood, W. T., et al. (1986). *The Specification of Complex Systems*. Wokingham: Addison-Wesley. (10. fejezet)
- Constantine, L. L.–Yourdon, E. (1979). *Structured Design*. Englewood Cliffs, NJ:
- Cooling, J. (2003). *Software Engineering for Real-Time Systems*. Harlow: Addison-Wesley. (15. fejezet)
- Coulouris, G., Dollimore, J., et al. (2001). *Distributed Systems: Concepts and Design*. Harlow: Addison-Wesley. (12. fejezet)
- Councill, W. T.–Heineman, G. T. (2001). Definition of a software component and its elements. In *Component-Based Software Engineering* (G. T. Heineman–W. T. Councill, eds.). Boston: Addison-Wesley, 5–20. (19. fejezet)
- Crabtree, A. (2003). *Designing Collaborative Systems: A Practical Guide to Ethnography*. London: Springer. (16. fejezet)
- Crosby, P. (1979). *Quality Is Free*. New York: McGraw-Hill. (27. fejezet)
- Curtis, B., Hefley, W. E., et al. (2001). *The People Capability Model: Guidelines for Improving the Workforce*. Boston: Addison-Wesley. (25., 28. fejezet)
- Cusamano, M. (1989). The software factory: a historical interpretation. *IEEE Software*, 6(2), 23–30. (18. fejezet)
- Czarnecki, K.–Eisenecher, U. (2000). *Generative Programming: Methods, Tools, and Applications*. Boston: Addison-Wesley. (18. fejezet)
- Davis, A. M. (1993). *Software Requirements: Objects, Functions, & States*. Englewood Cliffs, NJ: Prentice Hall. (6. fejezet)
- Dehbonei, B.–Mejia, F. (1995). Formal development of safety-critical software systems in railway signalling. In *Applications of Formal Methods* (M. Hinchey–J. P. Bowen, eds.). London: Prentice Hall, 227–52. (10. fejezet)
- DeMarco, T. (1978). *Structured Analysis and System Specification*. New York: Yourdon Press. (8. fejezet)
- DeMarco, T. (1978). *Structured Analysis and System Specification*. New York: Yourdon Press. (1. fejezet)
- DeMarco, T.–Boehm, B. (2002). The agile methods fray. *IEEE Computer*, 35(6), 90–2. (17. fejezet)

- DeMarco, T.-Lister, T. (1985). Programmer performance and the effects of the workplace. *Proc. 8th Int. Conf. on Software Engineering*, London: IEEE Press. (25. fejezet)
- DeMarco, T.-Lister, T. (1999). *Peopleware: Productive Projects and Teams*. New York: Dorset House. (25. fejezet)
- Diaper, D. (1989). *Task Analysis for Human-Computer Interaction*. Chichester: Ellis Horwood. (16. fejezet)
- Dijkstra, E. W. (1968). Cooperating sequential processes. In *Programming Languages* (F. Genuys, ed.). London: Academic Press, 43–112. (15. fejezet)
- Dijkstra, E. W. (1968). Goto statement considered harmful. *Comm. ACM*, **11**(3), 147–8. (20. fejezet)
- Dijkstra, E. W., Dahl, O. J., et al. (1972). *Structured Programming*. London: Academic Press. (23. fejezet)
- Dix, A., Finlay, J., et al. (2004). *Human Computer Interaction*, 3rd edn. Harlow: Addison-Wesley. (16. fejezet)
- Douglass, B. P. (1999). *Real-Time UML: Developing Efficient Objects for Embedded Systems*, 2nd edn. Boston: Addison-Wesley. (15. fejezet)
- DuBois, P. (1996). *Software Portability with imake*, 2nd edn. Sebastopol, CA: O'Reilly & Associates. (29. fejezet)
- Easterbrook, S. (1993). Domain modelling with hierarchies of alternative viewpoints. *Proc. RE'93*, San Diego, USA. (7. fejezet)
- Easterbrook, S., Lutz, R., et al. (1998). Experiences using lightweight formal methods for requirements modeling. *IEEE Trans. on Software Engineering*, **24**(1), 4–14. (10. fejezet)
- ECMA. (1991). A reference model for frameworks of computer-assisted software engineering environments. In *Reprints of the Seventh International Software Process Workshop*, Yountville, CA: ACM Press. (11. fejezet)
- Ehrlich, W., Prasanna, B., et al. (1993). Determining the cost of a stop-test decision. *IEEE Software*, **9**(4), 33–42. (24. fejezet)
- El-Amam, K. (2001). *Object-Oriented Metrics: A Review of Theory and Practice*. (27. fejezet)
- Elliott, J., Eckstein, R., et al. (2002). *Java Swing*, 2nd edn. Sebastopol, CA: O'Reilly & Associates Inc. (16. fejezet)
- Ellison, R. J., Fisher, D. A., et al. (1999). Survivability: protecting your critical systems. *IEEE Internet Computing*, **3**(6), 55–63. (3. fejezet)
- Ellison, R. J., Linger, R. C., et al. (1999). Survivable network system analysis: a case study. *IEEE Software*, **16**(4), 70–7. (3. fejezet)
- Ellison, R., Linger, R., et al. (2002). Foundations of survivable systems engineering. *Crosstalk: The Journal of Defense Software Engineering*, **12**, 10–15. (3. fejezet)
- Elrad, T., Askit, M., et al. (2001). Discussing aspects of AOP. *Comm. ACM*, **44**(10), 33–8. (18. fejezet)
- Endres, A. (1975). An analysis of errors and their causes in system programs. *IEEE Trans. on Software Engineering*, **SE-1**(2), 140–9. (3. fejezet)
- Erlikh, L. (2000). Leveraging legacy system dollars for e-business. *IT Pro*, **May/June 2000**, 17–23. (21. fejezet)
- Estublier, J.-Casallas, R. (1994). The Adele configuration manager. In *Configuration Management* (W. Tichy, ed.). Chichester: John Wiley & Sons, 99–134. (29. fejezet)
- Evans, D.-Laroche, D. (2002). Improving security using extensible lightweight static analysis. *IEEE Software*, **19**(1), 42–51. (22. fejezet)
- Ewald, T. (2001). Overview of COM+. In *Component-Based Software Engineering* (G. T. Heineman-W. T. Councill, eds.). Boston: Addison-Wesley, 573–88. (19. fejezet)
- Fagan, M. E. (1976). Design and code inspections to reduce errors in program development. *IBM Systems J.*, **15**(3), 182–211. (22. fejezet)

- Fagan, M. E. (1986). Advances in software inspections. *IEEE Trans. on Software Engineering*, **SE-12**(7), 744–51. (22. fejezet)
- Fayad, M. E.–Schmidt, D. C. (1997). Object-oriented application frameworks. *Comm. ACM*, **40**(10), 32–8. (18. fejezet)
- Feldman, S. I. (1979). MAKE—a program for maintaining computer programs. *Software-Practice and Experience*, **9**(4), 255–65. (29. fejezet)
- Firesmith, D. G. (2003). Analysing and specifying reusable security requirements. *Journal of Object Technology*, **2**(1), 53–68. (9. fejezet)
- Foster, I., Kesselman, C., et al. (2002). Grid services for distributed system integration. *IEEE Computer*, **35**(6), 37–46. (12. fejezet)
- Fowler, M.–Scott, K. (1997): *UML Distilled: Applying the Standard Object Modelling Language*. Reading, MA: Addison-Wesley. (7. fejezet)
- Frewin, G. D.–Hatton, B. J. (1986). Quality management—procedures and practises. *IEEE/BCS Software Engineering J.*, **1**(1), 29–38. (22. fejezet)
- Fromme, B.–Walker, J. (1993). An open architecture for tool and process integration. *Proc. 6th Conf. on Software Engineering Environments*, Reading, UK: IEEE Press. (11. fejezet)
- Fuggetta, A. (1993). A classification of CASE technology. *IEEE Computer*, **26**(12), 25–38. (4. fejezet)
- Fujiwara, E.–Pradhan, D. K. (1990). Error-control coding in computers. *IEEE Computer*, **23**(7), 63–72. (20. fejezet)
- Furey, S.–Kitchenham, B. (1997). Point/counterpoint: function points. *IEEE Software*, **14**(2), 28–31. (26. fejezet)
- Futatsugi, K., Goguen, J. A., et al. (1985). Principles of OBJ2. *Proc. 12th ACM Symp. on Principles of Programming Languages*, New Orleans: ACM Press. (10. fejezet)
- Gamma, E., Helm, R., et al. (1995). *Design Patterns: Elements of Reusable Object-Oriented Software*. Reading, MA: Addison-Wesley. (18. fejezet)
- Gane, C.–Sarson, T. (1979). *Structured Systems Analysis*. Englewood Cliffs, NJ: Prentice Hall. (4., 8. fejezet)
- Garlan, D., Allen, R., et al. (1995). Architectural mismatch: why reuse is so hard. *IEEE Software*, **12**(6), 17–26. (18. fejezet)
- Garlan, D., Kaiser, G. E., et al. (1992). Using tool abstraction to compose systems. *IEEE Computer*, **25**(6), 30–8. (11. fejezet)
- Garlan, D.–Shaw, M. (1993). An introduction to software architecture. *Advances in Software Engineering and Knowledge Engineering*, **1**, 1–39. (11., 13. fejezet)
- Garmus, D.–Herron, D. (2000). *Function Point Analysis: Measurement Practices for Successful Software Projects*. Boston: Addison-Wesley. (26. fejezet)
- Gilb, T.–Graham, D. (1993). *Software Inspection*. Wokingham: Addison-Wesley. (22. fejezet)
- Goldberg, A.–Robson, D. (1983). *Smalltalk-80. The Language and Its Implementation*. Reading, MA: Addison-Wesley. (16. fejezet)
- Gollmann, D. (1999). *Computer Security*. Chichester: John Wiley & Sons. (24. fejezet)
- Gomaa, H. (1993). *Software Design Methods for Concurrent and Real-Time Systems*. Reading, MA: Addison-Wesley. (15. fejezet)
- Gordon, V. S.–Bieman, J. M. (1995). Rapid prototyping: lessons learned. *IEEE Software*, **12**(1), 85–95. (17. fejezet)
- Gotterbarn, D., Miller, K., et al. (1999). Software engineering code of ethics is approved. *Comm. ACM*, **42**(10), 102–7. (1. fejezet)
- Grady, R. B. (1993). Practical results from measuring software quality. *Comm. ACM*, **36**(11), 62–8. (27. fejezet)
- Grady, R. B.–Van Slack, T. (1994). Key lessons in achieving widespread inspection use. *IEEE Software*, **11**(4), 46–57. (22. fejezet)

- Graham, I. (1994). *Object-Oriented Methods*, 2nd edn. Wokingham: Addison-Wesley. (14. fejezet)
- Griss, M. L.–Wosser, M. (1995). Making reuse work at Hewlett-Packard. *IEEE Software*, 12(1), 105–7. (18. fejezet)
- Groff, J. R., Weinberg, P. N., et al. (2002). *SQL: The Complete Reference*, 2nd edn. New York: McGraw-Hill Osborne. (17. fejezet)
- Grudin, J. (1989). The case against user interface consistency. *Comm. ACM*, 32(10), 1164–73. (16. fejezet)
- Guimaraes, T. (1983). Managing application program maintenance expenditures. *Comm. ACM*, 26(10), 739–46. (21. fejezet)
- Gunning, R. (1962). *Techniques of Clear Writing*. New York: McGraw-Hill. (27. fejezet)
- Gutttag, J. (1977). Abstract data types and the development of data structures. *Comm. ACM*, 20(6), 396–405. (10. fejezet)
- Gutttag, J. V.–Horning, J. J. et al. (1985): The Larch family of specification languages. *IEEE Software*, 2(5), 24–36. (10. fejezet)
- Gutttag, J., Horning, J., et al. (1993). *Larch: Languages and Tools for Formal Specification*. Heidelberg: Springer-Verlag. (10. fejezet)
- Haase, V., Messnarz, R., et al. (1994). Bootstrap: fine tuning process assessment. *IEEE Software*, 11(4), 25–35. (28. fejezet)
- Hall, A. (1990). Seven myths of formal methods. *IEEE Software*, 7(5), 11–20. (10. fejezet)
- Hall, A. (1996). Using formal methods to develop an ATC information system. *IEEE Software*, 13(2), 66–76. (3., 9., 10. fejezet)
- Hall, A.–Chapman, R. (2002). Correctness by construction: developing a commercially secure system. *IEEE Software*, 19(1), 18–25. (3., 9., 10. fejezet)
- Hall, E. (1998). *Managing Risk: Methods for Software Systems Development*. Reading, MA: Addison-Wesley. (5. fejezet)
- Hall, T.–Fenton, N. (1997). Implementing effective software metrics programs. *IEEE Software*, 14(2), 55–64. (27. fejezet)
- Halstead, M. H. (1977). *Elements of Software Science*. Amsterdam: North-Holland. (21. fejezet)
- Hamlet, D. (1992). Are we testing for true reliability? *IEEE Software*, 9(4), 21–7. (24. fejezet)
- Hammer, M. (1990). Reengineering work: don't automate, obliterate. *Harvard Business Review*, July–August 1990, 104–12. (28. fejezet)
- Hammer, M.–McLeod, D. (1981). Database descriptions with SDM: A semantic database model. *ACM Trans. on Database Sys.*, 6(3), 351–86. (8. fejezet)
- Hardin, D., Frerking, M., et al. (2002). Getting down and dirty: device-level programming using the real-time specification for Java. *Proc. Fifth IEEE International Symp. on*
- Harel, D. (1987). Statecharts: a visual formalism for complex systems. *Sci. Comput. Programming*, 8(3), 231–74. (8., 14., 15. fejezet)
- Harel, D. (1988). On visual formalisms. *Comm. ACM*, 31(5), 514–30. (8., 15. fejezet)
- Harker, S. D. P.–Easton, K. D. et al. (1993): The change and evolution of requirements as a challenge to the practice of software engineering. *Proc. RE'93*, San Diego, USA. (7. fejezet)
- Harold, E. R.–Means, W. S. (2002). *XML in a Nutshell*. Sebastopol, CA: O'Reilly & Associates. (13. fejezet)
- Hass, A. M. J. (2003). *Configuration Management: Principles and Practice*. Boston: Addison-Wesley. (29. fejezet)
- Hayes, I. (1987). *Specification Case Studies*. London: Prentice Hall. (10. fejezet)

- Heath, C.–Jirotko, M.–Luff, P.–Hindmarsh, J. (1993). Unpacking collaboration: the interactional organisation of trading in a City dealing room. In G. de Michelis, C. Simone, K. Schmidt (Eds.), *Proceedings of the Third European Conference on Computer-Supported Cooperative Work – ECSCW'93*, (pp. 155–170). Milan, Italy: Kluwer.
- Heath, C.–Luff, P. (1991): Collaborative activity and technological design: task coordination in the London Underground control room. *Proc. ECSCW'91*, Amsterdam, Kluwer. (7. fejezet)
- Heath, C.–Luff, P. (1992). Collaboration and control: crisis management and multimedia technology in London Underground control rooms. *Computer Supported Cooperative Work*, 1(1), 69–94. (7. fejezet)
- Heninger, K. L. (1980). Specifying software requirements for complex systems: new techniques and their applications. *IEEE Trans. on Software Engineering*, SE-6(1), 2–13. (6. fejezet)
- Highsmith, J. A. (2000). *Adaptive Software Development: A Collaborative Approach to Managing Complex Systems*. New York: Dorset House. (17. fejezet)
- Higuera-Toledano, M. T.–Issarny, V. (2000). Java embedded real-time systems: an overview of existing solutions. *Proc. Third IEEE International Symp. on Object-Oriented Real-Time Distributed Computing*, Newport Beach, CA: IEEE Computer Society Press. (15. fejezet)
- Hoare, C. A. R. (1974). Monitors: an operating system structuring concept. *Comm. ACM*, 21(8), 666–77. (15. fejezet)
- Hoare, C. A. R. (1985). *Communicating Sequential Processes*. London: Prentice Hall. (10. fejezet)
- Hofmeister, C., Nord, R., et al. (2000). *Applied Software Architecture*. Boston: Addison-Wesley. (11. fejezet)
- Horswill, J.–Miller, S. A. (2000). *Designing and Programming CICS Applications*. Sebastopol, CA: O'Reilly & Associates. (13. fejezet)
- Huang, Y.–Kintala, C. M. R. (1993). Software implemented fault tolerance: technologies and experience. *Proc. 23rd Fault-tolerant Computing Symposium (FTCS-23)*, Toulouse, France: IEEE Computer Society Press. (20. fejezet)
- Huff, C. C. (1992). Elements of a realistic CASE tool adoption budget. *Comm. ACM*, 35(4), 45–54. (4. fejezet)
- Huff, C.–Martin, C. D. (1995). Computing consequences: a framework for teaching ethical computing. *Comm. ACM*, 38(12), 75–84. (1. fejezet)
- Huff, K. E. (1996). Software process modeling. In *Trends in Software: Software Process* (A. Fuggetta–A. Wolf, eds.). Chichester: John Wiley & Sons, 1–24. (28. fejezet)
- Hughes, J. A., O'Brien, J., et al. (1997). Designing with Ethnography: A Presentation Framework for Design. In *Symposium on Designing Interactive Systems – DIS'97*, Amsterdam: ACM Press. (16. fejezet)
- Hughes, J.–Rodden, T. et al. (1994): Moving out from the control room: ethnography in system design. *Proc. CSCW'94*, Greensborough, North Carolina. (7. fejezet)
- Hull, R.–King, R. (1987). Semantic database modeling: survey, applications and research issues. *ACM Computing Surveys*, 19(3), 201–60. (8. fejezet)
- Humphrey, W. (1989). *Managing the Software Process*. Reading, MA: Addison-Wesley. (22., 27., 28. fejezet)
- Humphrey, W. S. (1988). Characterizing the software process. *IEEE Software*, 5(2), 73–9. (28. fejezet)
- Humphrey, W. S. (1995). *A Discipline for Software Engineering*. Reading, MA: Addison-Wesley. (28. fejezet)
- IEC. (1998). *Standard IEC 61508: Functional safety of electrical/electronic/programmable electronic safety-related systems*. (9. fejezet)

- IEEE. (1998). IEEE recommended practice for software requirements specifications. In *IEEE Software Engineering Standards Collection*. Los Alamitos, CA: IEEE Computer Society Press. (6. fejezet)
- IEEE. (2003). *IEEE Software Engineering Standards Collection on CD-ROM*. Los Alamitos, CA: IEEE Computer Society Press. (27. fejezet)
- Ince, D. (1994). *ISO 9001 and Software Quality Assurance*. London: McGraw-Hill. (27. fejezet)
- Jackson, M. A. (1983). *System Development*. London: Prentice Hall. (1., 8. fejezet)
- Jackson, M. A. (1995). *Requirements and Specifications*. Wokingham: Addison-Wesley. (6. fejezet)
- Jacky, J. (1995). Specifying a safety-critical control system. *IEEE Trans. on Software Engineering*, **21**(2), 99–106. (10. fejezet)
- Jacky, J. (1997). *The Way of Z: Practical Programming with Formal methods*. Cambridge, UK: Cambridge University Press. (10. fejezet)
- Jacky, J., Unger, J., et al. (1997). Experience with Z: developing a control program for a radiation therapy machine. *Proc. ZUM'97*, Reading: Springer. (10. fejezet)
- Jacobsen, I., Christerson, M., et al. (1993). *Object-Oriented Software Engineering*. Wokingham: Addison-Wesley. (6., 8., 15. fejezet)
- Jacobsen, I., Griss, M., et al. (1997). *Software Reuse*. Reading, MA: Addison-Wesley. (18., 19. fejezet)
- Jahanian, F.–Mok, A. K. (1986). Safety analysis of timing properties in real-time systems. *IEEE Trans. on Software Engineering*, **SE-12**(9), 890–904. (9. fejezet)
- Janis, I. L. (1972). *Victims of Groupthink. A Psychological Study of Foreign Policy Decisions and Fiascos*. Boston: Houghton Mifflin. (25. fejezet)
- Jelinski, Z.–Moranda, P. B. (1972). Software reliability research. In *Statistical Computer Performance Evaluation* (W. Frieberger, ed.). New York: Academic Press, 465–84. (24. fejezet)
- Johnson, P. L. (1993). *ISO 9000: Meeting the New International Standards*. New York: McGraw-Hill. (27. fejezet)
- Jones, C. B. (1980). *Software Development—A Rigorous Approach*. London: Prentice Hall. (10. fejezet)
- Jones, C. B. (1986). *Systematic Software Development Using VDM*. London: Prentice Hall. (10., 22. fejezet)
- Kafura, D.–Reddy, G. R. (1987). The use of software complexity metrics in software maintenance. *IEEE Trans. on Software Engineering*, **SE-13**(3), 335–43. (21. fejezet)
- Kan, S. H. (2003). *Metrics and Models in Software Quality Engineering*. Boston: Addison-Wesley. (24. fejezet)
- Kiczales, G., Hilsdale, E., et al. (2001). Getting started with AspectJ. *Comm. ACM*, **44**(10), 59–65. (18. fejezet)
- Kilpi, T. (2001). Implementing a software metrics program at Nokia. *IEEE Software*, **18**(6), 72–7. (27. fejezet)
- Kit, E. (1995). *Software Testing in the Real World: Improving the Process*. Reading, MA: Addison-Wesley. (22. fejezet)
- Kitchenham, B. (1990). Measuring software development. In *Software Reliability Handbook* (P. Rook, ed.). Amsterdam: Elsevier, 303–31. (27. fejezet)
- Kleppe, A., Warmer, J., et al. (2003). *MDA Explained: The Model-Driven Architecture—Practice and Promise*. Boston: Addison-Wesley. (14. fejezet)
- Knight, J. C.–Leveson, N. G. (1986). An experimental evaluation of the assumption of independence in multi-version programming. *IEEE Trans. on Software Engineering*, **SE-12**(1), 96–109. (20. fejezet)

- Knight, J. C.–Leveson, N. G. (2002). Should software engineers be licensed? *Comm. ACM*, **45**(11), 87–90. (24. fejezet)
- Knuth, D. E. (1971). *The Art of Computer Programming: Fundamental Algorithms*. Reading, MA: Addison-Wesley. (18. fejezet)
- Kotonya, G.–Sommerville, I. (1992): Viewpoints for requirements definition. *BCS/IEEE Software Engineering J.*, **7**(6), 375–87. (7. fejezet)
- Kotonya, G.–Sommerville, I. (1998). *Requirements Engineering: Processes and Techniques*. Chichester: John Wiley & Sons. (6. fejezet)
- Kreger, H. (2001). *Web Services Conceptual Architecture (WSCA 1.0)*. IBM. www.ibm.com/software/solutions/webservices/pdf/WSCA.pdf (12. fejezet)
- Krutchén, P. (2000). *The Rational Unified Process – An Introduction*. Reading, MA: Addison-Wesley. (4., 8. fejezet)
- Kumaran, S. I. (2001). *JINI Technology: An Overview*. Englewood Cliffs, NJ: Prentice Hall. (12. fejezet)
- Kuvaja, P., Similä, J., et al. (1994). *Software Process Assessment and Improvement: The BOOTSTRAP Approach*. Oxford: Blackwell Publishers. (28. fejezet)
- Lamping, J., Rao, R., et al. (1995). A focus + context technique based on hyperbolic geometry for visualising large hierarchies. *Proc. CHI'95*, Denver, CO: ACM Press. (16. fejezet)
- Laprie, J.-C. (1995). Dependable computing: concepts, limits, challenges. *Proc. 25th IEEE Symposium on Fault-Tolerant Computing*, Pasadena, CA: IEEE Press. (3. fejezet)
- Laprie, J.-C., Arlat, J., et al. (1995). Architectural issues in software fault tolerance. In *Software Fault Tolerance* (M. R. Lyu, ed.). New York: John Wiley & Sons 47–80. (20. fejezet)
- Larman, C. (2002). *Applying UML and Patterns: An Introduction to Object-Oriented Analysis and Design and the Unified Process*. Englewood Cliffs, NJ: Prentice Hall. (17. fejezet)
- Laudon, K. (1995). Ethical concepts and information technology. *Comm. ACM*, **38**(12), 33–9. (1. fejezet)
- Leblang, D. B.–Chase, R. P. (1987). Parallel software configuration management in a network environment. *IEEE Software*, **4**(6), 28–35. (29. fejezet)
- Lehman, M. M. (1996). Laws of software evolution revisited. *Proc. European Workshop on Software Process Technology (EWSPT'96)*, Nancy, France: Springer-Verlag. (21. fejezet)
- Lehman, M. M., Perry, D. E., et al. (1998). On evidence supporting the FEAST hypothesis and the laws of software evolution. *Proc. Metrics'98*, Bethesda, MD: IEEE Computer Society Press. (21. fejezet)
- Lehman, M. M., Ramil, J. F., et al. (2001). An approach to modelling long-term growth trends in software systems. *Proc. Int. Conf. on Software Maintenance*, Florence, Italy: IEEE Computer Society Press. (21. fejezet)
- Lehman, M. M.–Belady, L. (1985). *Program Evolution: Processes of Software Change*. London: Academic Press. (21. fejezet)
- Leveson, N. G. (1985). Software safety. In *Resilient Computing Systems* (T. Anderson, ed.). London: Collins, 12343. (3., 9. fejezet)
- Leveson, N. G. (1995). *Safeware: System Safety and Computers*. Reading, MA: Addison-Wesley. (9., 20. fejezet)
- Leveson, N. G.–Harvey, P. R. (1983). Analysing software safety. *IEEE Trans. on Software Engineering*, **SE-9**(5), 569–79. (9. fejezet)
- Leveson, N.–Stolzy, J. (1987). Safety analysis using Petri nets. *IEEE Transactions on Software Engineering*, **13**(3), 386–97. (9. fejezet)
- Lewis, P. M., Bernstein, A. J., et al. (2003). *Databases and Transaction Processing: An Application-Oriented Approach*. Boston: Addison-Wesley. (13. fejezet)

- Lientz, B. P.–Swanson, E. B. (1980). *Software Maintenance Management*. Reading, MA: Addison-Wesley. (21. fejezet)
- Linger, R. C. (1994). Cleanroom process model. *IEEE Software*, **11**(2), 50–8. (4., 22. fejezet)
- Liskov, B.–Guttag, J. (1986). *Abstraction and Specification in Program Development*. Cambridge, MA: MIT Press. (10. fejezet)
- Littlewood, B. (1990). Software reliability growth models. In *Software Reliability Handbook* (P. Rook, ed.). Amsterdam: Elsevier, 401–12. (3., 24. fejezet)
- Littlewood, B.–Verrall, J. L. (1973). A Bayesian reliability growth model for computer software. *Applied Statistics*, **22**, 332–46. (24. fejezet)
- Londeix, B. (1987). *Cost Estimation for Software Development*. Wokingham: Addison-Wesley. (26. fejezet)
- Lovelock, C., Vandermerwe, S., et al. (1996). *Services Marketing*. Englewood Cliffs, NJ: Prentice Hall. (12. fejezet)
- Lutz, M. (1996). *Programming Python*. Sebastopol, CA: O'Reilly & Associates. (17. fejezet)
- Lutz, R. R. (1993). Analysing software requirements errors in safety-critical embedded systems. *Proc. RE'93*, San Diego CA: IEEE Computer Society Press. (3., 22., 23. fejezet)
- MacDonell, S. G. (1994). Comparative review of functional complexity assessment methods for effort estimation. *BCS/IEE Software Engineering J.*, **9**(3), 107–17. (26. fejezet)
- Marshall, J. E.–Heslin, R. (1975). Boys and girls together: sexual composition and the effect of density on group size and cohesiveness. *J. of Personality and Social Psychology*, **35**(5), 952–61. (25. fejezet)
- Martin, D., Rodden, T., et al. (2001). Finding patterns in the fieldwork, *Proc. ECSCW'01*, Bonn: Kluwer. (18. fejezet)
- Martin, D., Rouncefield, M., et al. (2002). Applying patterns of interaction to work (re)design: e-government and planning. *Proc CHI'2002*, ACM Press. (18. fejezet)
- Maslow, A. A. (1954). *Motivation and Personality*. New York: Harper and Row. (25. fejezet)
- Massol, V.–Husted, T. (2003). *JUnit in Action*. Greenwich, CT: Manning. (23. fejezet)
- Matsumoto, Y. (1984). Some experience in promoting reusable software: presentation in higher abstract levels. *IEEE Trans. on Software Engineering*, **SE-10**(5), 502–12. (18. fejezet)
- McCabe, T. J. (1976). A complexity measure. *IEEE Trans. on Software Engineering*, **SE-2**(4), 308–20. (21. fejezet)
- McCue, G. M. (1978). IBM's Santa Teresa laboratory: architectural design for program development. *IBM Systems J.*, **17**(1), 4–25. (25. fejezet)
- McDougall, P. (2000). The power of peer-to-peer. *Information Week*, August 28, <http://www.informationweek.com>. (12. fejezet)
- McGuffin, R. W., Elliston, A. E., et al. (1979). CADES—software engineering in practice. *Proc. 4th Int. Conf. on Software Engineering*, Munich: IEEE Computer Society Press. (11. fejezet)
- McIlroy, M. D. (1968). Mass-produced software components. *Proc. NATO Conf. on Software Engineering*, Garmisch, Germany: Springer-Verlag. (18. fejezet)
- Meyer, B. (1992). Design by contract. *IEEE Computer*, **25**(10), 40–51. (19. fejezet)
- Meyer, B. (2003). The grand challenge of trusted components. *Proc. ICSE 25: Int. Conf. on Software Engineering*, Portland, OR: IEEE Press. (19. fejezet)
- Mili, H., Mili, A., et al. (2002). *Reuse-Based Software Engineering*. New York: John Wiley & Sons. (19. fejezet)
- Miller, G. A. (1957). The magical number 7 plus or minus two: Some limits on our capacity for processing information. *Psychological Review*, **63**, 81–97. (16. fejezet)
- Mills, H. D., Dyer, M., et al. (1987). Cleanroom software engineering. *IEEE Software*, **4**(5), 19–25. (3., 4., 22. fejezet)

- Mitschele-Thiel, A. (2001). *Systems Engineering with SDL: Developing Performance-Critical Communication Systems*. Chichester: John Wiley & Sons. (22. fejezet)
- MOD, (1995). *The Procurement of Safety Critical Software (Revised edn)*. UK Ministry of Defence, Interim Standard, 00–55. (22. fejezet)
- Mosley, D. J.–Posey, B. A. (2002). *Just Enough Test Automation*. Englewood Cliffs, NJ: Prentice Hall. (23. fejezet)
- Mullery, G. (1979): CORE – a method for controlled requirements specification. *Proc. 4th Int. Conf. on Software Engineering*, Munich, IEEE Press. (7. fejezet)
- Mumford, E. (1989). User participation in a changing environment—why we need it. In *Participation in Systems Development* (K. Knight, ed.). London: Kegan Paul. (2. fejezet)
- Munch, B. P., Larsen, J.-O., et al. (1993). Uniform versioning: the change-oriented model. *Proc. 4th Workshop on Software Configuration Management*, Baltimore, MD: ACM Press. (29. fejezet)
- Musa, J. D. (1993). Operational profiles in software reliability engineering. *IEEE Software*, **10**(2), 14–32. (24. fejezet)
- Musa, J. D. (1998). *Software Reliability Engineering: More Reliable Software, Faster Development and Testing*. New York: McGraw-Hill. (24. fejezet)
- Musciano, C.–Kennedy, B. (2002). *HTML & XHTML: The Definitive Guide*. Sebastopol, CA: O'Reilly & Associates. (16. fejezet)
- Myers, W. (1989). Allow plenty of time for large-scale software. *IEEE Software*, **6** (4), 92–9. (26. fejezet)
- Nakajo, T.–Kume, H. (1991). A case history analysis of software error-cause relationships. *IEEE Trans. on Software Engineering*, **18**(8), 830–8. (3. fejezet)
- Neil, M., Ostrolenk, G., et al. (1998). Lessons from using Z to specify a software tool. *IEEE Trans. on Software Engineering*, **24**(1), 15–23. (10. fejezet)
- Neilsen, J. (1993). *Usability Engineering*. New York: Academic Press. (16. fejezet)
- Nii, H. P. (1986). Blackboard systems, parts 1 and 2. *AI Magazine*, **7**(3 and 4), 38–53 and 62–9. (11. fejezet)
- Nilsen, K. (1998). Adding real-time capabilities to Java. *Comm. ACM*, **41**(6), 49–56. (15. fejezet)
- Norman, D. A.–Draper, S. W. (1986). *User-Centered System Design*. Hillsdale, NJ: Lawrence Erlbaum. (16. fejezet)
- Nosek, J. T.–Palvia, P. (1990). Software maintenance management: changes in the last decade. *Software Maintenance: Research and Practice*, **2**(3), 157–74. (21. fejezet)
- Nuseibeh, B. (1997). Ariane 5: who dunnit? *IEEE Software*, **14**(3), 15–16. (18. fejezet)
- O'Connor, J., Mansour, C., et al. (1994). Reuse in command and control systems. *IEEE Software*, **11**(4), 70–9. (18. fejezet)
- O'Leary, D. E. (2000). *Enterprise Resource Planning Systems: Systems, Life Cycle, Electronic Commerce and Risk*. Cambridge, UK: Cambridge University Press. (18. fejezet)
- Object-Oriented Real-Time Distributed Computing*, Washington, DC: IEEE Computer Society Press. (15. fejezet)
- Offen, R. J.–Jeffrey, R. (1997). Establishing software measurement programs. *IEEE Software*, **14**(2), 45–54. (27. fejezet)
- Oram, A. (2001). *Peer-to-Peer: Harnessing the Power of Disruptive Technologies*. Sebastopol, CA: O'Reilly & Associates. (12. fejezet)
- Oram, A.–Talbot, S. (1991). *Managing Projects with make*, 2nd edn. Sebastopol, CA: O'Reilly & Associates. (29. fejezet)
- Orcero, D. S. (2000). The code analyser LCLint. *Linux Journal*, **73**, <http://www.linuxjournal.com/article.php?sid=3599>. (22. fejezet)

- Orfali, R.–Harkey, D. (1998). *Client/Server Programming with Java and CORBA*. New York: John Wiley & Sons. (12. fejezet)
- Oskarsson, O.–Glass, R. L. (1995). *An ISO 9000 Approach to Building Quality Software*. Englewood Cliffs, NJ: Prentice Hall. (27. fejezet)
- Ould, M. A. (1995). *Business Processes: Modelling and Analysis for Re-engineering and Improvement*. Chichester: John Wiley & Sons. (28. fejezet)
- Ould, M. A. (1999). *Managing Software Quality and Business Risk*. Chichester: John Wiley & Sons. (5. fejezet)
- Ousterhout, J. (1994). *TCL and the TK toolkit*. Reading, MA: Addison-Wesley. (17. fejezet)
- Ousterhout, J. K. (1998). Scripting: higher-level programming for the 21st century. *IEEE Computer*, **31**(3), 23–30. (17., 26. fejezet)
- Palmer, S. R.–Felsing, J. M. (2002). *A Practical Guide to Feature-Driven Development*. Englewood Cliffs, NJ: Prentice Hall. (17. fejezet)
- Parnas, D. L., van Schouwen, J., et al. (1990). Evaluation of safety-critical software. *Comm. ACM*, **33** (6), 636–51. (20., 24. fejezet)
- Paulk, M. C., Curtis, B., et al. (1993). Capability maturity model, version 1.1. *IEEE Software*, **10**(4), 18–27. (28. fejezet)
- Paulk, M. C., Weber, C. V., et al. (1995). *The Capability Maturity Model: Guidelines for Improving the Software Process*. Reading, MA: Addison-Wesley. (28., 29. fejezet)
- Paulk, M. C.–Konrad, M. (1994). An overview of ISO's SPICE project. *IEEE Computer*, **27**(4), 68–70. (28. fejezet)
- Peach, R. W. (1996). *The ISO 9000 Handbook, 3rd edn*. New York: Irwin, Professional. (27., 29. fejezet)
- Perrow, C. (1984). *Normal Accidents: Living with High-Risk Technology*. New York: Basic Books. (3. fejezet)
- Peterson, J. L. (1981). *Petri Net Theory and the Modeling of Systems*. New York: McGraw-Hill. (9., 10. fejezet)
- Pfaff, G.–ten Hagen, P. J. W. (1985). *Seeheim Workshop on User Interface Management Systems*. Heidelberg: Springer-Verlag. (16. fejezet)
- Pfarr, T.–Reis, J. E. (2002). The integration of COTS/GOTS within NASA's HST command and control system. *Proc. ICCBSS 2002 (1st Int. Conf on COTS-based Software Systems)*, Orlando, FL: Springer-Verlag. (18. fejezet)
- Pfleeger, C. P. (1997). *Security in Computing, 2nd edn*. Englewood Cliffs, NJ: Prentice Hall. (3. fejezet)
- Pope, A. (1998). *CORBA*. Harlow: Addison-Wesley. (19. fejezet)
- Potter, B., Sinclair, J., et al. (1996). *An Introduction to Formal Specification and Z*. London: Prentice Hall. (10. fejezet)
- Preiser, W.–Ostoff, E. (2001). *The Universal Design Handbook*. New York: McGraw-Hill. (16. fejezet)
- Prentice Hall. (4., 8. fejezet)
- Prieto-Diaz, R.–Arango, G. (1991). *Domain Analysis and Software System Modeling*. Los Alamitos CA: IEEE Press. (7. fejezet)
- Pritchard, J. (1999). *COM and CORBA Side by Side: Architectures, Strategies, and Implementations*. Boston: Addison-Wesley. (12. fejezet)
- Prowell, S. J., Trammell, C. J., et al. (1999). *Cleanroom Software Engineering: Technology and Process*. Reading, MA: Addison-Wesley. (4., 10., 22., 24. fejezet)
- Pulford, K., Kuntzmann-Combelles, A., et al. (1996). *A Quantitative Approach to Software Management*. Wokingham: Addison-Wesley. (27., 28. fejezet)
- Pullum, L. L. (2001). *Software Fault Tolerance Techniques and Implementation*. Norwood, MA: Artech House. (20. fejezet)

- Putnam, L. H. (1978). A general empirical solution to the macro software sizing and estimating problem. *IEEE Trans. on Software Engineering*, **SE-4**(3), 345–61. (26. fejezet)
- Randell, B. (1975). System structure for software fault tolerance. *IEEE Trans. on Software Engineering*, **SE-1**(2), 220–32. (20. fejezet)
- Randell, B.–Xu, J. (1995). The evolution of the recovery block concept. In *Software Fault Tolerance* (M. R. Lyu, ed.). New York: John Wiley & Sons, 1–22. (20. fejezet)
- Rankin, C. (2002). The software testing automation framework. *IBM Systems J.*, **41**(1), 126–40. (23. fejezet)
- Redmill, F. (1998). IEC 61508: principles and use in the management of safety. *IEEE Computing and Control Engineering J.*, **9**(10), 205–13. (9. fejezet)
- Reiss, S., P. (1990). Connecting tools using message passing in the field environment. *IEEE Software*, **7**(4), 57–66. (11. fejezet)
- Rettig, M. (1994). Practical programmer: prototyping for tiny fingers. *Comm. ACM*, **37**(4), 21–7. (17. fejezet)
- Rittel, H.–Webber, M. (1973). Dilemmas in a general theory of planning. *Policy Sciences*, **4**, 155–69. (2. fejezet)
- Robertson, S.–Robertson, J. (1999). *Mastering the Requirements Process*. Harlow: Addison-Wesley. (6. fejezet)
- Robinson, P. J. (1992). *Hierarchical Object-Oriented Design*. Englewood Cliffs, NJ: Prentice Hall. (4., 8., 14. fejezet)
- Ross, D. T. (1977). Structured analysis (SA): a language for communicating ideas. *IEEE Trans. on Software Engineering*, **SE-3**(1), 16–34. (6. fejezet)
- Royce, W. W. (1970). Managing the development of large software systems: concepts and techniques. *Proc. IEEE WESTCON*, Los Angeles CA: IEEE Computer Society Press. (4. fejezet)
- Rubin, K.–Goldberg, A. (1992). Object behaviour analysis. *Comm. ACM*, **35**(9), 48–62. (14. fejezet)
- Rumbaugh, J., Blaha, M., et al. (1991). *Object-Oriented Modeling and Design*. Englewood Cliffs, NJ: Prentice Hall. (1., 4., 8. fejezet)
- Rumbaugh, J., Jacobson, I., et al. (1999). *The Unified Modeling Language Reference Manual*. Reading, MA: Addison-Wesley. (1., 4., 8., 14. fejezet)
- Rumbaugh, J., Jacobson, I., et al. (1999). *The Unified Software Development Process*. Reading, MA: Addison-Wesley. (1., 4., 8. fejezet)
- Sackman, H., Erikson, W. J., et al. (1968). Exploratory experimentation studies comparing on-line and off-line programming performance. *Comm. ACM*, **11**(1), 3–11. (26. fejezet)
- Schmidt, D. C. (1997). Applying design patterns and frameworks to develop object-oriented communications software. In *Handbook of Programming Languages*, Vol. 1 (P. Salus, ed.). London: Macmillan Computer Publishing. (18. fejezet)
- Schneidewind, N. F.–Keller, T. W. (1992). Applying reliability models to the space shuttle. *IEEE Software*, **9**(4), 28–33. (24. fejezet)
- Schoman, K.–Ross, D. T. (1977). Structured analysis for requirements definition. *IEEE Trans. on Software Engineering*, **SE-3**(1), 6–15. (6. fejezet)
- Schwaber, K.–Beedle, M. (2001). *Agile Software Development with Scrum*. Englewood Cliffs, NJ: Prentice Hall. (17. fejezet)
- Scott, J. E. (1999). The FoxMeyer Drug's bankruptcy: was it a failure of ERP? *Proc. Association for Information Systems 5th Americas Conf. on Information Systems*, Milwaukee, WI. (18. fejezet)
- Selby, R. W., Basili, V. R., et al. (1987). Cleanroom software development: an empirical evaluation. *IEEE Trans. on Software Engineering*, **SE-13**(9), 1027–37. (4., 22. fejezet)

- Sheldon, F. T., Kavi, K. M., *et al.* (1992). Reliability measurement: from theory to practice. *IEEE Software*, 9(4), 13–20. (24. fejezet)
- Shlaer, S.–Mellor, S. (1988). *Object-Oriented Systems Analysis: Modeling the World in Data*. Englewood Cliffs, NJ: Yourdon Press. (14. fejezet)
- Shneiderman, B. (1998). *Designing the User Interface, 3rd edn.* Reading, MA: Addison-Wesley. (16. fejezet)
- Siegal, J. (1998). OMG overview: CORBA and the OMA in enterprise computing. *Comm. ACM*, 41(10), 37–43. (12. fejezet)
- Silberschatz, A., Galvin, P. B., *et al.* (2002). *Operating System Concepts, 6th edn.* New York: John Wiley & Sons. (15. fejezet)
- Skonnard, A.–Gudgin, M. (2002). *Essential XML Quick Reference: A Programmer's Reference to XML, XPath, XSLT, XML Schema, SOAP, and More*. Boston: Addison-Wesley. (12. fejezet)
- Snyder, C. (2003). *Paper Prototyping: The Fast and Easy Way to Design and Refine User Interfaces*. San Francisco: Morgan Kaufmann. (16. fejezet)
- Sommerville, I.–Rodden, T. *et al.* (1993): Integrating ethnography into the requirements engineering process. *Proc. RE'93*, San Diego, CA, IEEE Computer Society Press. (7. fejezet)
- Spafford, E. (1989). The Internet worm: crisis and aftermath. *Comm. ACM*, 32(6), 678–87. (3. fejezet)
- Spivey, J. M. (1990). Specifying a real-time kernel. *IEEE Software*, 7(5), 21–8. (10. fejezet)
- Spivey, J. M. (1992). *The Z Notation: A Reference Manual, 2nd edn.* London: Prentice Hall. (10., 22. fejezet)
- Stal, M. (2002). Web services: beyond component-based computing. *Comm. ACM*, 45(10), 71–6. (12. fejezet)
- Stapleton, J. (1997). *DSDM Dynamic Systems Development Method*. Harlow: Addison-Wesley. (17. fejezet)
- Stephens, M.–Rosenberg, D. (2003). *Extreme Programming Refactored*. Berkley, CA: Apress. (17. fejezet)
- Stevens, P.–Pooley, R. (1999). *Software Engineering with Objects and Components*. Harlow: Addison-Wesley. (6. fejezet)
- Storey, N. (1996). *Safety-Critical Computer Systems*. Harlow: Addison-Wesley. (9., 20. fejezet)
- Suchman, L. (1983). Office procedures as practical action. *ACM Trans. on Office Information Systems*, 1(3), 320–28. (16. fejezet)
- Swartz, A. J. (1996). Airport 95: automated baggage system? *ACM Software Engineering Notes*, 21(2), 79–83. (2. fejezet)
- Symons, C. R. (1988). Function-point analysis: difficulties and improvements. *IEEE Trans. on Software Engineering*, 14(1), 2–11. (26. fejezet)
- Szyperski, C. (2002). *Component Software: Beyond Object-Oriented Programming, 2nd edn.* Harlow: Addison-Wesley. (12., 19. fejezet)
- Tanenbaum, A. S. (2001). *Modern Operating Systems, 2nd edn.* Englewood Cliffs, NJ: Prentice Hall. (15. fejezet)
- Thayer, R. H. (1997). Software system engineering: an engineering process. In *Software Requirements Engineering* (R. H. Thayer–M. Dorfmann, eds.). Los Alamitos: IEEE Computer Society Press, 84106. (2. fejezet)
- Thayer, R. H. (2002). Software system engineering: a tutorial. *IEEE Computer*, 35(4), 6873. (2. fejezet)
- Tichy, W. (1985). RCS—a system for version control. *Software Practice and Experience*, 15(7), 637–54. (29. fejezet)

- Tracz, W. (2001). COTS myths and other lessons learned in component-based software development. In *Component-Based Software Engineering* (G. T. Heineman–W. T. Councill, eds.). Boston: Addison-Wesley, 99–112. (18. fejezet)
- Turner, M., Budgen, D., et al. (2003). Turning software into a service. *IEEE Computer*, **36**(10), 38–45. (12. fejezet)
- Ulrich, W. M. (1990). The evolutionary growth of software reengineering and the decade-ahead. *American Programmer*, **3**(10), 14–20. (21. fejezet)
- Vesperman, J. (2003). *Essential CVS*. Sebastopol, CA: O'Reilly & Associates. (29. fejezet) REF 727-742.qxd 4/3/04 3:46 PM Page 741
- Wall, L., Christiansen, T., et al. (1996). *Programming Perl*. Sebastopol, CA: O'Reilly & Associates. (17. fejezet)
- Wang, N., Schmidt, D. C., et al. (2001). Overview of the CORBA component model. In *Component-Based Software Engineering* (G. T. Heineman–W. T. Councill, eds.). Boston: Addison-Wesley, 557–72. (19. fejezet)
- Ward, P.–Mellor, S. (1985). *Structured Development for Real-Time Systems*. Englewood Cliffs, NJ: Prentice Hall. (8. fejezet)
- Warmer, J.–Kleppe, A. (1998). *The Object Constraint Language: Precise Modeling with UML*. Boston: Addison-Wesley. (19. fejezet)
- Warren, I. (1998). *The Renaissance of Legacy Systems*. London: Springer. (21. fejezet)
- Weinberg, G. (1971). *The Psychology of Computer Programming*. New York: Van Nostrand. (17., 25. fejezet)
- Weinreich, R.–Sametinger, J. (2001). Component models and component services: concepts and principles. In *Component-Based Software Engineering* (G. T. Heineman–W. T. Councill, eds.). Boston: Addison-Wesley, 33–48. (19. fejezet)
- Weiss, S. (2002). *Handheld Usability*. New York: John Wiley & Sons. (16. fejezet)
- White, B. A. (2000). *Software Configuration Management Strategies and Rational ClearCase*. Reading, MA: Addison-Wesley. (29. fejezet)
- White, S., Alford, M., et al. (1993). Systems engineering of computer-based systems. *IEEE Computer*, **26**(11), 54–65. (2. fejezet)
- Whitgift, D. (1991). *Software Configuration Management: Methods and Tools*. Chichester: John Wiley & Sons. (29. fejezet)
- Whittaker, J. W. (2002). *How to Break Software: A Practical Guide to Testing*. Boston: Addison-Wesley. (23. fejezet)
- Williams, L., Kessler, R. R., et al. (2000). Strengthening the case for pair programming. *IEEE Software*, **17**(4), 19–25. (17. fejezet)
- Wirfs-Brock, R. J.–Johnson, R. E. (1990). Surveying current research in object-oriented design. *Comm. ACM*, **33**(9), 104–24. (18. fejezet)
- Wirfs-Brock, R., Wilkerson, B., et al. (1990). *Designing Object-Oriented Software*. Englewood Cliffs, NJ: Prentice Hall. (14. fejezet)
- Wordsworth, J. (1996). *Software Engineering with B*. Wokingham: Addison-Wesley. (4., 9., 10., 22. fejezet)
- Wordsworth, J. B. (1991). The CICS application programming interface definition. *Proc. Z User Workshop*, Oxford, Berlin: Springer-Verlag. (10. fejezet)
- Zimmermann, H. (1980). OSI reference model—the ISO model of architecture for open systems interconnection. *IEEE Transactions on Communications*, **COM-28**(4), 425–32. (11. fejezet)

TÁRGYMUTATÓ

A, Á

absztrakciók 174
 absztrakt adattípus 227–229
 ACM 15–17
 ActiveX-vezérlők 277
 adaptív karbantartás 501
 adat újratervezés 510
 adatbázisok 409
 adatfeldolgozó rendszerek 300–302
 adatfolyam
 diagramok 301–302
 modellek 9, 177–179
 adatgyűjtő
 rendszerek 358–361
 valós idejű szoftver tervezése 358–361
 adatintegrációs szolgáltatások 266
 adatmodellek 183–185
 adatszerkezet tervezése 79
 adatszótár 184
 adattároló szolgáltatások 266
 additív kompozíció 461
 ADL 250
 aggregáció 189–190
 agilis módszerek 400–402
 aktív objektumok 323
 alapértelmezett input feldolgozása 479
 ALARP 200
 alfa-tesztelés 83
 algebrai specifikáció 227–233
 algoritmusok
 hiba 205
 tervezés 79
 alkalmazás
 adatok 40
 architektúra 297–299
 adatfeldolgozó rendszerek 300–302
 eseményfeldolgozó rendszerek 309–312
 nyelvfeldolgozó rendszerek 312–314
 tranzakciófeldolgozó rendszerek 302–309
 kapcsolatok 412
 keretrendszerek 432–433
 rendszer-újrafelhasználás 434–442

rétegek 275
 szakterület 425
 újrafelhasználás 420
 alkalmazás programozói interfészek 138 *lásd még* interfészek
 alkalmazáscsaládok 433, 438–442
 alkalmazásszintű védelem 737
 alkalmazói szoftver 40
 állapotátmenet modellek 179–182, 334
 állapotdiagramok 179
 álnevek használata 479
 alrendszerek 22
 azonosítása 28
 eseményszóró modellek 262
 fejlesztése 32–33
 formális specifikáció 226–233
 modellek 334
 általános modellek 264
 általános termékek 6
 aperiodikus ingerek 344
 API 138 *lásd még* interfészek
 alkalmazásrendszer újrafelhasználása 434
 APSE 254
 áramszünet 354
 architektúra 87
 alkalmazás
 adatfeldolgozó rendszerek 300–302
 eseményfeldolgozó rendszerek 309–312
 nyelvfeldolgozó rendszerek 312–314
 tranzakciófeldolgozó rendszerek 302–309
 erőforrás-lefoglaló rendszerek 441
 hibatűró 490–494
 MDA 318
 osztott rendszerek 270–273 *lásd még* osztott rendszerek
 rétegek 308
 architekturális terv felépítése 251–255
 architekturális tervezés 79, 245–248, 329

biztonsági tervezésben 736–739
 moduláris felbontás 256–259
 referenciaarchitektúrák 264–267
 vezérlési stílusok 259–264
 aritmetikai hiba 205
 AspectJ 786, 800
 aspektusok 787, 788, 791–803
 azonosítása 798
 szövése 431, 787, 790–791
 aspektusorientált szoftverfejlesztés 430, 780–782
 aspektusok, csatlakozási pontok és vágási pontok 786–791
 vonatközfások szétválasztása 782–786
 aspektusorientált tervezés és programozás 796–800
 átadás 85–86
 ATM 136–137
 hibák osztályozása 215
 kliens-szerver architektúra 276–277
 környezeti modell 175
 tranzakciófeldolgozó rendszerek 302–309
 átmeneti követelmények 165–166
 átvételi tesztelés 80
 átvizsgálások 475
 V & V (verifikáció és validáció) 530–535
 automatizált statikus elemzés 535–538
 azonosítás
 evolúciós folyamatok 507
 kockázat 199–200
 komponensek 458
 objektumok 330–332

B

belövés 81, 87
 bemenet validálása 743
 bemenet-feldolgozás-kimenet 300
 beszerzés
 fejlesztés 37
 rendszer 436

bizalmasság 15, 49
 bizalmasságot érintő fenyegetés 728
 információ kiadása 60
 biztonság és használhatóság 741–742
 biztonsági tervezés 725–727
 architektúrális tervezés 736–739
 irányelvek 739–744
 fogalmak 727–729
 kockázatkezelés 730–735
 rendszer túlélőképessége 746–749
 biztonságos bukás 741
 biztonságos programozás 478–480
 biztonságosság 49
 kritikus rendszerek 56–59
 kritikus rendszerek specifikációja 206–208
 tervezése 246
 biztonságpolitika 740
 BPMN 773
 British Computer Society 15

C, CS

CASE 12
 eszközkészlet 252
 generátor alapú szoftver újrafelhasználás 429–431
 meta-CASE-eszközök 313
 osztályozás 89–91
 szoftverfolyamatok 87–91
 célok kijelölése 75
 Cleanroom
 folyamatok 66
 szoftverfejlesztés 540–543
 CORBA 282–287, 452–454
 COTS 33
 alkalmazásrendszer újrafelhasználása 434–442
 biztonsági tervezés 746
 Create művelet 231
 csatlakozási pontok 431, 786–791
 modell 787, 789
 csoportok, ősrendszerek 512

D

dinamikus memóriefoglalás 478
 dinamikus modellek 333
 diverzitás 742
 dobozos 6
 dokumentáció *lásd még* architektúrális tervezés

szoftverkövetelmények 139–142
 terv dokumentálása 335
 E, É
 egyedi igényekre szabott termékek 6
 együttműködés 436
 eldobható prototípus készítése 69
 elemzés
 felhasználók 380–381, 382–385
 fenyegetettség 209 *lásd még* kockázat
 hatás 506
 kockázat 110–112, 200–202
 komponensek 71
 környezeti modellek 175–177
 követelmények 77
 követelménytervezési folyamat 149–151
 objektumorientált 185
 rendszerek 246
 statikus 475
 V & V (verifikáció és validáció) 535–538
 életciklus 68
 elfogadhatatlan kockázat 200
 elfogadható kockázat 200
 elkerülési stratégiák 112
 kockázat 198
 ellentmondás-mentesség
 ellenőrzések 162
 szoftverkövetelmények 123
 elosztás 736
 előkészítés 85
 előrejelzés, karbantartási 504–506
 előzetes kockázatkértékelés 731–733
 elsődlegesen biztonságosság-kritikus szoftverek 57
 ENTER művelet 231
 eredendő rendszertulajdonások 23–24
 erőforrás-foglaló rendszerek 306, 441
 erőforrás-kezelő rendszerek 304–309
 ERP 6, 298, 439
 értékelés 388–389
 érthetőség 164
 eseményfeldolgozó rendszerek 309–312
 eseményszóró modellek 262
 eseményvezérelt rendszerek 262

eszközkészletek 90
 eszközök
 meta-CASE 313
 módszertámogatás 192–193
 szoftverfolyamatok 90
 eszközszolgáltatás 769
 etikai előírások 15
 etikai felelősség 14–18
 ETK (egyed-tulajdonosság-kapcsolat) modellezés 183
 etnográfia 160–162, 384–385
 evolúció 8, 84, 496–498
 COTS 436
 folyamatok 506–512
 karbantartás 500–506
 ősrendszerek 512–518
 programdinamika 498–500
 terv 339–340
 evolúciós fejlesztés 66, 69–70
 extrém programozás (XP) 403–409

F

FAQ 5–14
 fejlesztés
 alkalmazáscsaládtag 442
 Cleanroom szoftverfejlesztés 540–543
 evolúciós 66, 69–70
 folyamatok 446
 gyors szoftverfejlesztés *lásd* gyors szoftverfejlesztés
 integráció 71
 iteráció 87
 kritikus rendszerek 470–474
 hibatűrés 483–490
 hibatűrő architektúrák 490–494
 megbízható programozás 475–476
 megbízható szoftverfejlesztési eljárások 474–475
 objektummodellek 185–191
 spirális 74–76
 újrafelhasználható komponensek 454–457
 Unified Software Development Process 85
 validálás 76
 feladatelemzés 383–384
 feladatkezelő szolgáltatások 266
 felbontás 202–204
 felderítés
 hiba 484–489
 kockázat 204

- felhasználói felületek
gyors szoftverfejlesztés 396
szolgáltatások 266
tervezés 79, 365–369
értékelés 388–390
folyamat 380–385
prototípus-készítés 386–387
- felhasználók
interakciók 370–373
szoftverkövetelmények 129–132
- felmérések 389
- feltáró fejlesztés 69
- fenyegetettség elemzése 209, 728–729
- figyelés 96
kockázat 113
- figyelőrendszerek 353–358
- fluxusfigyelő rendszer architektúrája 359
- folyamatok 8
CBSE 457–460
Cleanroom 66
evolúció 506–512
fejlesztés 446
felhasználói felületek tervezése 380–381
folyamatkezelés 351–353
formális specifikáció 223–226
kényszerjavítás 509
követelménytervezés 145–147 *lásd még* követelménytervezési folyamat
megbízható szoftver 474–475
modellek 9
működtetési 37
nyomkövetés, belövés 81, 87
objektumorientált tervezés 325–338
paradigmák 66
prototípusfejlesztés 416
Rational Unified Process 66, 148
szervezeti 36–39
szoftver *lásd* szoftverfolyamatok
tervezése 98
Unified Software Development Process 85
üzleti 40
XP (extrém programozás) 403–409
- folyamatszolgáltatások 762
- forogatókönyvek, követelménytervezés 156–158
- formális módszerek 221–223, 538–543
- formális specifikáció 221–223
alrendszerinterfészek 226–233
szoftverfolyamat 223–226
viselkedés 233–239
- forráskód lefordítása 510
- funkcionális követelmények 26, 121–129
- funkcionális tulajdonságok 23
- funkcionalitás
alrendszerek 28
COTS 434
- funkcióorientált csövezeték 256, 258–259
- függelékek, szoftverkövetelmények dokumentuma 140
- független komponensek 446
- függőségek 103
biztonságtól 728–729
- függvények 421
- G, GY**
generalizációs hierarchia 322
generalizációs kapcsolatok 188
generatív programozás 430
generátor alapú újrafelhasználás 429–431
- GIOP 286
- gyakran ismétlődő kérdések 5–14
- gyors szoftverfejlesztés 395–400
agilis módszerek 400–402
prototípezálás 413–417
RAD 409–413
XP (extrém programozás) 403–409
- H**
hálózatok
kliens-szerver architektúra 274–279
protokollok 286
- hardver 39
COTS 34
hibatűrés 491
megbízhatóság 24, 211–212
- háromrétegű kliens-szerver architektúra 277
- használat modelljei objektumorientált tervezésnél 327–329
- használati esetek követelmények tervezésénél 158–160
- határtalan tömbök 479
- hatáselemzés 506
- helyreállítás
hiba 489–490
tervezés 744
túlélőképessé rendszerek 747
visszaállító blokkok 492
- heterogenitás kihívása 14
- hiba bekövetkezési hányadosa 213
- hibák 44 *lásd még* kritikus rendszerek
áramszünet 354
elkerülése 54
fák 203
hibamentes szoftver 472
keresés és eltávolítás 54
osztályozás 215
tűrés 54, 483–490
- hibatűrés 50 *lásd még* kritikus rendszerek
- hierarchiák
generalizációs 322
osztályok 187–188
- hierarchikus kompozíció 461
- hívás-visszatérés modell 260
- hozzzáértés, 15
- I**
IDL 284
időszzerűség 49
IEEE 15
igény hibás teljesítésének valószínűsége 213
implementáció 67, 78–81
indítványok írása 96–97
információelrejtés 476–477
információkezelő rendszerek 304–309
információmegjelenítés 373–378
inkrementális fejlesztés 72–74
integráció 33–34
COTS 434
fejlesztés 71
UDDI 292
integritást érintő fenyegetések 728
interakció 370–373
interaktor nézőpontok 153
interdiszciplináris vonatkozások 25
interfészek 28
generátorok 410
komponensek 450
objektum 337–338
specifikáció 137–138, 226–233
interjúk 154–156

- internetalapú prototípus-készítés 387
- inzulinadagoló 46–48
- állapot megszorítások 484
- biztonságossági indoklás 585–588
- formális specifikáció 235–239
- futási idejű ellenőrzés 590–591
- követelmények 135, 205
- IOR 284
- irányelvek, üzleti 40
- iteráció
- fejlesztés 9, 87
 - RUP 86
 - szoftverfolyamatok 72–76
- J**
- Java
- appletok 277
 - épületfigyelő folyamat 357
 - hibatűrés 483–490
 - információelrejtés 476–477
 - kivételkezelés 480–483
 - valós idejű verziói 346
- javítás időtartamának átlagos ideje 212
- javíthatóság 50
- jelentéskészítő 410
- JINI modell 290
- K**
- kapcsolatok 409, 412
- karbantartás 67
- evolúció 500–506
 - gyors szoftverfejlesztés 398
 - karbantartási szolgáltatás 769
- karbantarthatóság 50
- tervezése 247
- károk korlátozása 59
- kényszerjavítási folyamatok 509
- kérdőívek 388
- keretrendszerek
- alkalmazás 432–433
 - MVC 433
- kétrétegű kliens-szerver architektúra 275
- keveredés 786, 788
- kezelés
- biztonságosság 206–208
 - folyamatok 351–353
 - gyors szoftverfejlesztés 398
 - követelmények 475
 - követelménytervezési folyamat 164–170
 - teszt 475
- kezelőmodell 260
- kihelyezési idejű konfigurálás 438
- kiterjesztések 791, 792, 796–797
- kivételkezelés 480–483
- kivételek 765
- kliens-szerver
- architektúra 274–279
 - modell 253–254
- kockázat
- azonosítása 109–110
 - becslése 75
 - csökkentése 201–202, 204–205
 - elemzése 110–112
 - figyelése 113
 - kezelése 730–735
 - kiértékelése 730–731
 - kritikus rendszerek specifikációja 198–205
 - megszüntetése 204
 - osztályozása 200–202
 - projektmenedzsment 107–113
 - rendszer újratervezése 509
 - tervezése 112–113
- kockázatkértékelés az életciklusban 731, 733–735
- komponensalapú szoftvertervezés 9, 66, 70–72, 445–448
- folyamatok 457–460
- komponens kompozíció 460–467
- modellek 448–456
- szoftverfejlesztés 756
- komponensek
- elemzés 71
 - interfészek 450
 - módszerek 12
 - ősrendszerek 40
 - tervezés 79
 - tesztelés 82
 - újrafelhasználás 420
 - újrafelhasználható komponensek fejlesztése 454–457
- kompozíció 460–467
- tervezése 798
- koncepcionális újrafelhasználás 420
- konfliktuselemzés és -feloldás 798
- konkurens objektumok 323–324
- konstruktorműveletek 229
- koordinációs szolgáltatások 762
- korai aspektusok 792
- korrektív karbantartás 501, 506
- költség 10–11
- formális specifikáció 224
 - projektmenedzsment 97
 - rendszer újratervezés 509–512
 - üzembiztonság 51
- környezetek 91
- környezeti modellek 175–177
- köteget feldolgozórendszerek 298–299
- követelmények *lásd még* szoftverkövetelmények
- elemzés 67
 - felderítés 152–160
 - feltárás és elemzés 77, 149–162
 - felülvizsgálata 163
 - kezelés 87, 475
 - módosítás 71
 - specifikáció 78
 - validáció 78
 - változtatások kezelése 169–170
 - védettség 210
- követelmény-felosztás 28
- követelménymódosítás 71
- követelménytervezési folyamat 119, 145–147
- elemzés 149–162
 - feltárás 149–162
 - kezelés 164–170
 - megvalósíthatósági tanulmányok 148–149
 - validálás 162–164
- központosított vezérlés 260–261
- köztes termék 282, 446
- integrációs keretrendszerek 432
- közvetett nézőpontok 153
- kritikus meghibásodási pont 741
- kritikus rendszerek 44–46, 51–56, 211–217
- biztonságosság 56–59
 - fejlesztés 470–474
 - hibatűrés 483–490
 - hibatűrő architektúrák 490–494
 - megbízható programozás 475–483
 - megbízható szoftverfejlesztési eljárások 474–475
 - megbízhatóság 51–56

rendelkezésre állás 51–56
 specifikáció 196–197
 biztonságosság 206–208
 kockázaton alapuló specifikáció 198–205
 szoftver megbízhatósága 211–217
 védettség 208–211
 típusai 46–48
 üzembiztonság 48–51
 védettség 59–61
 kulcsfigurák 149
 kommunikáció 246
 kulcsrakész 33
 külső követelmények 125

L

Leave művelet 231
 lebegőpontos számok 478
 Lehman-törvények 498–500
 lekérdezések 278
 lekérdező műveletek 229
 létrehozás 85
 LIBSYS 122, 125
 adatmodellek 184
 architektúra 305–306
 felhasználói felület 373
 követelmények 154
 objektumviselkedési modellek 190
 szakterületi követelmények 128
 számlázórendszer 130
 újrafelhasználás 429
 XP (extrém programozás) 403
 logikai interfész tervezése 765
 Lookup művelet 231

M

magrendszer tervezése 798
 másodlagosan biztonságosságkritikus szoftverek 57
 MDA 318
 megbízható programozás 475–483
 megbízható szoftverfejlesztési eljárások 474–475
 megbízhatóság 49
 meghatározás, követelmények elemzése és felbontása 67
 meghibásodás átlagos ideje 212
 meglévő rendszerek, struktúrált módszerek 191–193
 megszakítás 479
 megszakításvezérelt modellek 262

megvalósíthatósági ellenőrzések 162
 megvalósíthatósági tanulmányok 77
 követelménytervezési folyamat 148–149
 mérőföldkövek 100–101
 meta-CASE-eszközök 313
 metrikák 212–214
 felületek értékelése 388–390
 megbízhatóság 212–214
 minimalizációs stratégiák 112
 minták
 Observer 428, 429
 tervezési 426–429
 mobil kód 277
 modellek

architektúrális terv 250
 CBSE 445–456
 ellenőrzés 475
 folyamatok 9
 MDA 318
 osztott objektummodell 280
 rendszermodellek *lásd* rendszermodellek
 rendszertervezés 30–32
 strukturált módszerek 80
 szoftverfolyamatok 66–72
 tervezési modellek 332–337
 módszerek 12
 agilis *lásd* agilis módszerek
 komponensek 12
 V & V (verifikáció és validáció) 538–543

moduláris felbontás 256–259
 modulokra bontás 510
 Move művelet 231, 233
 MTTF 212
 MTTR 212
 munkafolyamatok
 modellek 9
 RUP 86
 mutatók 478
 mutatókkal realizált adatstruktúrák redundanciája 489
 művelet 67
 absztrakt adattípusok 229
 inkompatibilitás 462
 nem teljessége 462
 MVC 373 433

N, NY

naplózó szolgáltatás 770
 nem preemptív ütemezés 352
 nemfunkcionális eredendő tulajdonságok 23
 nemfunkcionális szoftverkövetelmények 121–129, 214–217

névterek 759
 névtervezés 798
 nézőpont-orientált szemléletek, követelménytervezés 153–154, 793–794
 nyelvek
 formális specifikáció 225–226
 IDL 284
 Java *lásd* Java
 minták 426–429
 nyelvfeldolgozó rendszerek 312–314
 OCL 464
 SQL 278
 UML 12, 158
 vizuális programozási nyelvek 387
 WSDL 292
 nyomkövetés 81, 87
 nyomon követhetőség 164, 167

O, Ö

objektummodellek 185–191
 objektumok
 aggregáció 189–190
 azonosítás 330–332
 konkurens 323–324
 osztott rendszerek 279–287
 specifikáció 227
 újrafelhasználás 420
 viselkedési modellek 190–191
 objektumorientált felbontás 256, 256–257
 objektumorientált tervezés 317–320
 folyamat 325–338, 339–340
 osztályok 320–324
 Observer minta 428, 429
 OCL 464
 OMG 283
 OO (objektumorientált) 12
 operációs rendszerek 350–353
 operátormegbízhatóság 24, 211
 ORB 263, 284
 oszlopdiagramok 103–106 *lásd* még projektmenedzsment
 osztályok
 hibák 215
 hierarchiák 187–188
 objektumok 320–324
 osztott rendszerek
 kliens-szerver architektúra 274–279
 osztott objektumarchitektúrák 279–287
 szervezeti közti osztott rendszerek 287–294

- többprocesszoros architektúrák 273–274
 OWL-S 768
 öröklődés 479
 modellek 187–189
 őrendszerek 39–41
 evolúció 512–517
 felmérése 512
 szolgáltatások 769–770
- P**
 p2p (peer-to-peer) rendszerek 287–290
 paraméter inkompatibilitás 462
 párhuzamosság 478
 páros programozás 408–409
 perfekтив karbantartás 501
 periodikus ingerek 344
 platform, szoftver újrafelhasználás 426
 platformszintű védelem 737
 POFOD 213
 precizitás 49
 preemptív ütemezés 352
 preventív hibadetektálás 484
 procedurális interfészek 138
 programátvizsgálás 475, 531–535
 programok megrongálása 60
 programstruktúra-fejlesztés 510
 projektmenedzsment 94–95
 tervezés 97–101
 tevékenységek 96–97
 ütemezés 101–106
 prototípus-készítés
 felhasználói felületek 386–387
 gyors szoftverfejlesztés 413–417
 követelmények 163
 rendszerek 381
- R**
 rácseszközök 130
 RAD 409–413
 redundancia 488
 biztonsági tervezés esetén 742–743
 TMR 491
 referenciaarchitektúrák 264–267
 rekordszintű védelem 738
 rekurzió 479
 rendelkezésre állás 49
 kritikus rendszerek 51–56
 metrika 212
 rendelkezésre állást érintő fenyegetés 728
 tervezése 246–247
 rendszercélok 126
 rendszer felépítése 251–255
 rendszer infrastruktúra keretrendszerek 432
 rendszer újratervezés 509–512
 rendszerelemzés 246
 rendszerevolúció 34
 rendszerhibák 45 *lásd még*
 kritikus rendszerek
 rendszerkörnyezet objektumorientált tervezésnél 327–329
 rendszerkövetelmények 132–137
 rendszermodellek 173–174
 adatmodellek 183–185
 környezeti modellek 175–177
 objektummodellek 185–191
 strukturált módszerek 191–193
 viselkedési modellek 177–182
 rendszertervezés 25–26
 alrendszer fejlesztése 32–33
 és szoftvertervezés 8
 evolúció 34
 követelmények meghatározása 26–28
 modellek 30–32
 tervezés 28–30
 üzemen kívül helyezés 35
 vállalkozók 38
 reprezentációk 174
 részeredmények 100
 rétegek
 alkalmazásrétegek 275
 rétegzett architektúra 308
 rétegzett modell 254–255
 őrendszerek 41
 retrospektív hibadetektálás 484
 riasztás behatolás miatt 354
 ROCOF 213
 RTOS 350–353
 RUN séma 237
 RUP 66, 85–87
 megvalósíthatósági tanulmányok 148
- S**
 segédszolgáltatások 761–762
 sémák
 RUN 237
 Z 234
 semlegesítés 61
 sérülékenységi
 elkerülése 61
 és biztonság 728–729, 733–735, 736
 SOAP 292, 753, 754
 specializáció szoftver újrafelhasználásnál 438–442
 specifikáció 8
 absztrakt 79
 formális *lásd* formális specifikáció
 interfészek 137–138, 337–338
 követelmények 78
 kritikus rendszerek 196–197
 biztonságosság 206–208
 kockázaton alapuló 198–205
 szoftver megbízhatósága 211–217
 védetség 208–211
 objektumok 227
 strukturált nyelv 134–137
 szoftver 77–78
 szoftverkövetelmény 139
 természetes nyelv 133
 védetség 208–211
 spirális fejlesztés 74–76
 SQL 278
 adatbázisok 409–410
 statikus elemzés 475
 V & V (verifikáció és validáció) 535–538
 statikus modellek 333
 strukturált módszerek 12, 80
 rendszermodellek 191–193
 strukturált nyelvű specifikációk 134–137
- SZ**
 szabványok, komponens 446
 szakmai felelősség 14–18
 szakterületek
 alkalmazás 425
 nézőpontok 153
 szoftverkövetelmények 128–129
 szakterület-specifikus architektúrák 264
 szállítás kihívása 14
 számítás táblázatos megadása 136
 számítógépes visszaélés 15
 számítógép-rendszerek 35–36
 számítógép-tudomány 7
 szekvenciális kompozíció 461
 szekvenciamodellek 333

- széles körű újrafelhasználhatóság 246
szellemi tulajdonjogok 15
szemantikus adatmodellek 183
szerepkör-/cselekvésmo­dell 9
szerkesztőrács eszközök 131
szerkesztőrendszerek 309–310
szerverek
 kliens-szerver architektúra 274–279
 konkurens objektumok 323
szervezetek 35–36
szervezeti követelmények 125
szervezetközi osztott rendszerek 287–294
szerződések
 gyors szoftverfejlesztés 398
 karbantartási 503
szétszóródás 785–788
színek, felhasználói felületek 377–378
szkriptvezérelt prototípus­késztés 387
szociális tervezés 742
szociotechnikai rendszerek 21
szoftver élettartama 425
szoftver újrafelhasználás 420–423
 alkalmazás keretrendszerek 432–433
 alkalmazásrendszer újrafelhasználása 434–442
 generátor alapú újrafelhasználás 429–431
 technikák 423–425
szoftverek tulajdonságai 13
szoftverfolyamatok 64–65
 CASE 87–91
 iteráció 72–76
 modellek 66–72
 RUP 85–87
 tevékenységek 76–84
szoftverkövetelmények 119–121
 dokumentum 139–142
 felhasználói 129–132
 funkcionális és nem funkcionális 121–129
 interfész-specifikáció 137–138
 rendszerkövetelmények 132–137
 szakterületek 128–129
 szoftverkövetelmény-specifikáció 139
szoftvermegbízhatóság 24, 211–217
szoftvertervezés 4
 gyakran ismétlődő kérdések 5–14
 kihívások 14
 költségek 10–11
 módszerek 12
 szolgáltatásorientált 756
szoftver-újratervezés 510
szolgáltatás megtagadása 60
szolgáltatásjelöltek azonosítása 761–764
szolgáltatások
 felületek 266
 komponens modellek 454
 modellek 290
szolgáltatások tervezése 760–770
 implementáció és üzembe helyezés 767–768
 interfész tervezése 764–767
 jelöltek azonosítása 761–764
 szolgáltatások őrsz­re­szere­ből 769–770
szolgáltatásorientált rendszer-architektúra 290–294, 753
szolgáltatásorientált szoftvertervezés 752–756
 fejlesztés 770–778
 munkafolyamat-tervezés és -implementáció 773–776
szolgáltatások tervezése 760–770
 tesztelés 776–778
 újrafelhasználható komponensek 756–760
szükséges követelmények 132
T
támadás felderítése 61, 729
támogató szoftver 39–40
tanács 787
tárgymutatók 140
tárolási modell 251–252
tárolók 454
tartós követelmények 165–166
taszk *lásd* tevékenység
technikai számítógép-alapú rendszerek 21
teljesítmény 23–24
 tervezés 246
teljesség 123
 ellenőrzések 162
termék 7
termékfejlesztési költségek 11
termékre vonatkozó követelmények 125
természetes nyelv 133
tervezés 67, 76, 78–81
 adatszerkezet 79
 algoritmikus költségmodellek 644–646
 algoritmus 79
 architekturális *lásd* architektu­rális tervezés
 felhasználói felületek 79, 365–369
 értékelés 388–390
 folyamat 380–381
 prototípus-készítés 386–387
 gyors szoftverfejlesztés *lásd* gyors szoftverfejlesztés
 kockázat 112–113
 komponens 79
 követelmények kezelése 167–169
 objektumorientált 317–320
 folyamat 325–338, 339–340
 osztályok 320–324
 projektmenedzsment 97–101
 specifikáció 225
 szoftver újrafelhasználás 420–423, 426–429
 tervezési modellek 332–337
 tesztelés 475
 újrafelhasználás 71
 üzenetek 379
 V & V (verifikáció és validáció) 527–529
 valós idejű szoftverek 343–316
 vizsgálatok 475
tervezés a rendszer bevezetésé­nek figyelembevételével 744–746
tervezési idejű konfigurálás 439
teszt 67, 82
 tervezés 475
 XP (extrém programozás) 404–408
tesztet generálása 163
tevékenységek
 időtartama 103
 oszlopdiagramok 105
 projektmenedzsment 96–97
 szoftverfolyamatok 76–84
 tevékenység­hálók 100–104
 lásd még projektmenedzsment
tevékenységmodell 9
TMR 491

- többprocesszoros architektúrák 273–274
- többszörös öröklődés 189
- többszeri programozás 492
- tranzakciófeldolgozó rendszerek 302–309
- tulajdonságok 27
- túlélési képesség 50
- túlélőképes rendszerek
- ellenállás 747
 - felismerés 747
 - helyreállítás 747
 - túlélőképes rendszerelemzés 747–749
- U, Ü
- UD (Universal Design) 369
- UDDI 292, 753, 754–755, 768, 773
- újrafelhasználás
- komponens fejlesztése 454–457
 - széles körű 246
 - szoftver *lásd* szoftver-újrafelhasználás
 - szolgáltatásorientált szoftvertervezés során 756–760
 - tervezés 71
 - vizuális programozás 411
- újratervezés 509–512
- UML 12, 158
- Unified Software Development Process 85
- úrlapok
- gyors szoftverfejlesztés 410
 - strukturált nyelvű specifikációk 134–135
- ütemezések
- nem preemptív/preemptív 352
 - projektmenedzsment 101–106
 - szoftver újrafelhasználás 424
- üzembiztonság 48–51
- üzemeltetési folyamatok 37
- üzenetek
- tervezése 378
 - üznetszolgáltatások 266
- üzleti folyamatok 40
- üzleti irányelvek 40
- üzleti objektumok 281
- üzleti szolgáltatások 762
- V
- V & V (verifikáció és validáció) 523–527
- átvizsgálások 530–535
 - automatizált statikus elemzés 535–538
 - formális módszerek 538–543
 - tervezés 527–529
- vágási pontok 786, 788–790
- vagyon
- elemzés 732
 - és biztonság 728–729
 - tagolás 743
- validáció, validálás 8, 82–84
- aspektusorientált rendszerekben 800–803
 - fejlesztés és validálás 76
 - gyors szoftverfejlesztés 398
 - követelmények 78, 162–164
- validitás-ellenőrzések 162
- vállalati alkalmazás keretrendszerek 432
- vállalatirányítási rendszerek 6, 298, 439
- valós idejű operációs rendszer 350–353
- valós idejű szoftverek tervezése 343–346
- adatgyűjtő rendszerek 358–361
 - figyelő- és vezérlőrendszerek 353–358
 - rendszertervezés 346–349
 - RTOS 350–353
- vastag kliens modell 275
- védelem 736–738
- védettség 49
- követelmények 210–211
- kritikus rendszerek 59–61
- kritikus rendszerek specifikációja 208–211
- tervezés 246
- vékony kliens modell 275
- verifikálás
- aspektusorientált rendszerekben 800–803
 - követelménytervezés 164
 - nem funkcionális szoftver-követelmények 126
- verifikálhatóság 163
- veszély 201 *lásd még* kockázat
- elkerülés 58
 - felderítés és eltávolítás 58
- vészhelyzeti tervek 112
- veszteségfelmérés 484–489
- vezérlési stílusok 259–264
- vezérlőrendszerek 353–358
- biztonságos 728–729
- viselkedés
- formális specifikáció 233–239
 - objektumok 190–191
- viselkedési modellek 177–182
- vízéses megközelítési mód 9
- vízésesmodell 66, 67–69
- vizuális programozás
- nyelvek 387
 - újrafelhasználás 411
- VOLERE követelménytervezési módszer 132
- vonatkozások 791–792
- szétválasztása 430, 782–786
- vonatkozásorientált követelmény-feltárás 792–796
- W, X, Y, Z
- webalapú felületek 372 *lásd még* felhasználói felületek
- WSDL 292, 754, 757–758, 767
- XP (Extrém programozás) 403–409
- YAWL 773
- Z séma 234
- példák 236–238