OKOSVÁROS-TECHNOLÓGIÁK XVI.

IT-rendszerek és kritikus szolgáltatások teljesítménye

Kocsis Imre – Klenik Attila

SZÉCHENYI 2020

MAGYARORSZÁG KORMÁNYA

Európai Unió
Európai Szociális Alap

BEFEKTETÉS A JÖVŐBE

NEMZETI KÖZSZOLGÁLATI EGYETEM
BUDAPEST

OKOSVÁROS-TECHNOLÓGIÁK
A technológia fejlődésének irányai és hatása
XVI. kötet

Sorozatszerkesztő:

Sallai Gyula

Kocsis Imre – Klenik Attila

IT-rendszerek és kritikus szolgáltatások teljesítménye

Nemzeti Közszolgálati Egyetem
Közigazgatási Továbbképzési Intézet
Budapest, 2020

A kötet a Nemzeti Közszolgálati Egyetem **KÖFOP-2.1.2-VEKOP-15-2016-00001 „A jó kormányzást megalapozó közszolgálat-fejlesztés”** projektje keretében, a Budapesti Műszaki és Gazdaságtudományi Egyetem Villamosmérnöki és Informatikai Karán létesült **„Okos város – okos közigazgatás”** kutatóműhelyben (2017/162/BME-VIK) készült.

Szakmai lektor:
Pataricza András, egyetemi tanár, BME-VIK

A kézirat lezárásának dátuma:
2017. december 31.

© Nemzeti Közszolgálati Egyetem
Közigazgatási Továbbképzési Intézet, 2020
© Kocsis Imre, 2020
© Klenik Attila, 2020

A mű szerzői jogilag védett. Minden jog, így különösen a sokszorosítás, terjesztés és fordítás joga fenntartva. A mű a kiadó írásbeli hozzájárulása nélkül részeiben sem reprodukálható, elektronikus rendszerek felhasználásával nem dolgozható fel, azokban nem tárolható, azokkal nem sokszorosítható és nem terjeszthető.

TARTALOMJEGYZÉK

1. Bevezető ... 7

2. Okos városok ... 9

2.1. Az okos város fogalma ... 9
2.2. Okosváros-megoldások és okosváros-stratégia ... 10
2.3. Okos város és integrált településfejlesztési stratégia ... 11

3. Az okos várost lehetővé tevő IKT-képességek ... 13

3.1. Mindenhol elérhető kommunikáció ... 13
3.2. „A dolgok internete” – Internet of Things (IoT) ... 13
3.3. Mobil számítástechnika ... 14
3.4. „Nagy adat” adatelemzés ... 14
3.5. Felhő-számítástechnika ... 15
3.5.1. Szolgáltatásmodellek ... 15
3.5.2. Infrastruktúra mint szolgáltatás (IaaS) ... 15
3.5.3. Platform mint szolgáltatás (PaaS) ... 16
3.5.4. Szoftver mint szolgáltatás (SaaS) ... 16
3.5.5. A felhő-számítástechnika néhány alapvető jellemzője ... 17
3.5.6. Telepítési modellek ... 17
3.5.7. Felhőszolgáltatások és okosváros-megoldások ... 18
3.6. A műszaki innováció következő hulláma ... 18

4. Teljesítménytervezés és -menedzsment: motiváció ... 20

4.1. Teljesítményhibák okosváros-megoldásokban ... 20
4.2. Teljesítménykövetelmények és IKT-kapacitások ... 22

5. A teljesítménymenedzsment és tervezése ... 26

5.1. Az IT Infrastructure Library ... 26
5.2. Teljesítménytervezés- és menedzsment az ITIL-ben ... 26
5.3. Szolgáltatási szintek: kapacitás és teljesítmény ... 28
5.4. Skálázható kapacitás ... 29
5.5. QoS-metrikák meghatározása ... 30
5.6. Kapacitás-, terhelés- és igénymenedzsment ... 30
5.7. ITIL szemléletű okos város teljesítménymenedzsment ... 32

6. Autonóm teljesítménymenedzsment ... 34

6.1. Autonóm számítástechnika ... 34
6.2. Eljárásrend-alapú felügyeleti logika és futásidejű tervezés ... 35
6.3. Autonóm teljesítménymenedzsment ... 37
6.4. A szabályozástechnika eszközeit alkalmazó megoldások ... 42
6.5. Autonóm okos város? ... 44

7. A túlterheléskezelés automatizmusai ... 45
7.1. Elnapolható munka ... 45
7.2. Megfelelő erőforrás-megosztás ... 46
7.3. Automatikus védelmek ... 46
7.3.1. Munkamegosztás ... 47
7.3.2. Terheléseldobás ... 47
7.3.3. Alkalmazási kérdések ... 48
7.4. Automatikus kapacitáskiterjesztés ... 49
7.5. „Kemény” hibák teljesítményhibaként kezelése ... 49
8. Összefoglalás ... 51
9. Hivatkozások ... 52

1. Bevezető

Az okos városok (smart cities) magukban hordozzák annak lehetőségét, hogy az információ- és kommunikációtechnológia eszközeinek segítségével minőségi javulást érjünk el a városi élet szinte minden területén.

Az eszközökbe és tárgyakba beépített intelligenciával és távol, számítási felhőkben igénybe vett informatikai szolgáltatásokkal „okossá" tett környezet (utak, közművek, autonóm járművek, modern telekommunikációs hálózatok stb.) közvetlenül fogják befolyásolni mindennapjainkat már a közeljövőben is.

Szinte törvényszerű, hogy az okosváros-képességek elterjedésével azok egy részére mint kritikus képességre kell majd tekintenünk; pontosan úgy, ahogy a kritikus fizikai infrastruktúrák mellett ma már beszélünk kritikus informatikai infrastruktúrákról is. Ezek tágan értelmezett védelme – tehát zavartalan működésük biztosítása – nemzeti és összeurópai érdek (lásd pl. [BÁNYÁSZ et al. 2013]).

A kritikus szolgáltatások a követelményeknek megfelelő üzemeltetése több aspektusú feladat. Egyik kulcseleme a megfelelő teljesítmény biztosítása – különösen kiberfizikai rendszerek esetén, ahol a fizikai világ változásait szinte minden rendszernek valamilyen értelemben „időszerűen" kell kezelnie.

Jelen kismonográfia „A jó kormányzást megalapozó közszolgálat-fejlesztés" projekt „Okos város-okos közigazgatás" műhelyében jött létre; fő célja, hogy egy áttekintő képet adjon az okos városok számítástechnikai teljesítménymenedzsmentjének kihívásairól és lehetőségeiről. A kismonográfia szerkezete a következő:

- A 2. fejezet röviden bevezeti az okos városok fogalmát és tervezésük-üzemeltetésük egyik módszertanát. Később ez szolgál a teljesítménytervezés és menedzsment mint tevékenységek keretéül.
- A 3. fejezet áttekinti az okosváros-megoldásokat lehetővé tevő főbb IKT-rendszer-kategóriákat. Célja kettős: részben szemlélteti a rendszereket, amelyek feldolgozási/átviteli/tárolási stb. kapacitásainak menedzseléséről később általánosabb értelemben értekezünk, részben pedig bemutatja a felhő-számítástechnikát, melynek elasztikus skálázódási képessége az okos városokban is a hatékony teljesítménymenedzsment egyik alapvető eszköze.
- A 4. fejezet bemutatja, hogy az okos városban egy (rész)rendszer teljesítményhibái milyen tovaterjedő hatásokkal rendelkezhetnek, megteremtve a teljesítménymenedzsment motivációját. A fejezet továbbá az adatfolyam feldolgozás, mint IKT-megoldás példáján szemlélteti, hogy az informatikai rendszerekben a teljesítmény megfelelőségét alapvetően a rendszer terhelésének és feldolgozási kapacitásainak viszonya határozza meg.
- Az 5. fejezet bemutatja a teljesítménymenedzsment klasszikus folyamatait, és javaslatokat tesz azok az okosváros-kontextusra adaptálására.
- A 6. fejezet bemutatja az autonóm számítástechnikát és ezen belül az autonóm teljesítménymenedzsmentet. Fő üzenete, hogy a nagyvállalati informatika modern, nagyban automatizált felügyeleti megközelítései két szinten alkalmazhatók okosváros-kontextusban; mind a városi szolgáltatáshálón, mind pedig az egyes

okosváros-megoldásokban. Kiemelt elemként jelenik meg az ellenállóság fogalma; egy okos városban a szolgáltatások teljesítményének tervezett módon kell tudnia degradálódni kapacitáskimerülés és hibák esetén.

- A 7. fejezet áttekinti azokat a műszaki automatizmuskategóriákat, amelyek segítségével informatikai rendszereket a kapacitáskimerüléssel szemben védünk.

Kismonográfiánk fejezeteinek célközönsége nem egységes. Szándékunk szerint az első öt fejezetet haszonnal forgathatja a széles érdeklődő publikum, különös tekintettel azokra a döntéshozókra és közszolgálatot végző szakértőkre, akik okosváros-stratégia kialakításában vesznek részt, vagy okosváros-projektek létrehozásánál, megvalósításánál és utókövetésénél segédkeznek. Ezen fejezetek megértése műszaki előképzettséget nem igényel.

Az első öt fejezetre építve a 6. fejezet azon műszaki informatikai hátterű közszolgáknak nyújthat a legtöbbet, akik az okosváros-kezdeményezések érettségi szint emelésének kérdéskörét vizsgálják. A fejezet által bemutatott autonóm számítástechnikai megközelítések az okos városok területére átültetése még jórészt nyitott kérdéskör; mindemellett néhány ajánlást itt is megfogalmazunk.

A 7. fejezet a teljesítménybiztosítás műszaki eszközeit tárgyalja. Így szándékunk szerint fő célközönségét a rendszermérnökök és műszaki informatikusok jelentik – úgy a közszolgálatban, mint az okosváros-megoldásokat megvalósító piaci szereplőknél.

A szerzők[1]

[1] *Dr. Kocsis Imre*, okl. mérnökinformatikus, a Budapesti Műszaki Egyetem Villamosmérnöki és Informatikai Kara Méréstechnikai és Információs Rendszerek Tanszékének adjunktusa. PhD fokozatát a BME-en 2019-ben szerezte. Fő kutatási területe az elosztott megvalósítású informatikai szolgáltatások ellenállóképességre tervezése. Kutatásainak fő alkalmazási területei az „edge felhő" elemeket alkalmazó kiberfizikai rendszerek, illetve az elosztott főkönyvi rendszerek (Distributed Ledger Technology, DLT).
Klenik Attila, okl. mérnökinformatikus, a Budapesti Műszaki Egyetem Villamosmérnöki és Informatikai Kara Méréstechnikai és Információs Rendszerek Tanszékének doktorandusz hallgatója. Fő kutatási területe az elosztott informatikai infrastruktúrák megfelelő teljesítményre tervezése, illetve meglévő infrastruktúrák teljesítményének elemzése.

2. Okos városok

Az információ- és kommunikációtechnológia (IKT, Information and Communications Technology) fejlődése évtizedünkre jutott el arra a pontra, hogy a városok működését és a városok lakóinak életét alapvetően képes legyen átalakítani. A városi élet szinte minden területén folyamatosan jelennek meg olyan megoldások, amelyek IKT-eszközök újszerű, intelligens alkalmazásával teszik könnyebbé, hatékonyabbá, biztonságosabbá vagy éppen igazságosabbá a polgárok, a gazdasági szereplők és a kormányzat tevékenységeit és mindennapjait.

2.1. Az okos város fogalma

A város működésének kihívásaira így válaszoló városokat hívjuk általánosságban „okos városnak" („smart city"). A fenti gondolat általánossága miatt az okos város fogalmára számos definíciót ismerünk (lásd pl. [ALBINO–BERARDI–DANGELICO 2015]) – nem utolsósorban azért is, mert a világ különböző részeinek különböző méretű, gazdaságú és kultúrájú városai radikálisan különböző problémákkal küzdenek.

Ha leíró módon közelítjük meg az okos város fogalmát, akkor egy város „okossága" megragadható azon keresztül, hogy mennyire hatékonyan és előremutató módon képes kezelni hat városi alrendszer meghatározó tényezőit [GIFFINGER–PICHLER–MILANOVIĆ 2007] (hazánkban e kategorizálás egy változatát alkalmazza a Lechner Tudásközpont is [Lechner s. a.]):

- Okos kormányzás (smart governance): részvétel a döntéshozatalban, közszolgáltatások és szociális szolgáltatások, a kormányzás átláthatósága.
- Okos mobilitás (smart mobility): fenntartható, innovatív és biztonságos közlekedési rendszerek és elérhetőségük, IKT-infrastruktúra.
- Okos környezet (smart environment): természeti környezet vonzósága, környezetszennyezés és -védelem, fenntartható erőforrásgazdálkodás.
- Okos gazdaság (smart economy): innováció, vállalkozókedv, termelékenység, munkapiac rugalmassága, nemzetközi beágyazottság.
- Okos életkörülmények (smart living): kultúra, egészség, személyi biztonság, oktatási rendszer, turisztikai vonzóerő, társadalmi összetartás (kohézió).
- Okos emberek (smart people): képzettség, élethosszan tartó tanulás készsége, flexibilitás, kreativitás, részvétel a közügyekben.

Ezen tényezőkhöz mutatók is rendelhetők, amelyek segítségével városok összehasonlíthatóvá válnak, illetve az „okos városfejlesztések" hatása mérhetővé is válik (bővebben lásd [GIFFINGER–PICHLER–MILANOVIĆ 2007]). A fenti, dekompozíción alapú meghatározás egyúttal rámutat az okos városok fogalmának egyik alapvető tulajdonságára is: egy város „okosságát" nem az IKT-megoldások határozzák meg önmagukban. Helyesebb úgy fogalmaznunk, hogy a megfelelően tervezett és bevezetett IKT-megoldások, amelyeket a város működésének résztvevői akarnak és tudnak használni, akár radikálisan is képesek az egyes tényezők tekintetében vett „okosságot" növelni.

Ebben az értelemben az okos(abb) város felé való elmozdulást segít(het)ik elő például a következő fejlesztések ([COPENHAGEN s.a.] alapján).

- Városi wifi (municipal WiFi, Muni-Fi): az okosváros-megoldások közös infrastruktúráját adhatja, de pusztán az akár korlátozott, de mindenhol elérhető internet-hozzáférés nyújtásával is okosabbá teheti a várost.
- Okos parkolás: a parkolóhelyekbe épített szenzorokkal és megfelelő mobilalkalmazás-támogatással megvalósítható többek között a szabad parkolóhelyek gyors megtalálása, előre foglalása vagy akár a dinamikus parkolóhely-árazás.
- Forgalomoptimalizálás: a közúti forgalom folyamatos, valós idejű megfigyelése lehetőséget teremt az intelligens, a trendeket és az aktuális helyzetet is figyelembe vevő forgalomirányításra (pl. forgalmi lámpák váltási idejének felülvezérlése).
- Okos hulladék: a megfelelő szenzorokkal (és alacsony sávszélességű kommunikációs képességgel) felszerelt hulladékgyűjtő edények lehetővé teszik például az elszállítás optimalizálását és a használatalapú árazást.

Hasonló jól meghatározott, önálló célként számtalan okosváros-megoldást ismerünk ma már; ezek a közvetlen megvalósíthatóság és a létező kérdésekre válaszadás szempontjából igen széles skálán mozognak, az inkrementális innovációtól a ma még radikálisnak tűnő víziókig. Európai példák széles skáláját találhatjuk meg például Amszterdam városának interaktív okosváros-térképén [AMSTERDAM s. a.], Berlin okosváros-stratégiájának referenciaprojekt-mellékletében [BERLIN 2015], Stockholm hosszú távú okosváros víziójában [STOCKHOLM s. a.], vagy Budapest viszonylatában a Lechner Tudásközpont online példatárában (LECHNER s. a.) (lásd még Budapest okos város jövőképét is [BUDAPEST 2017]).

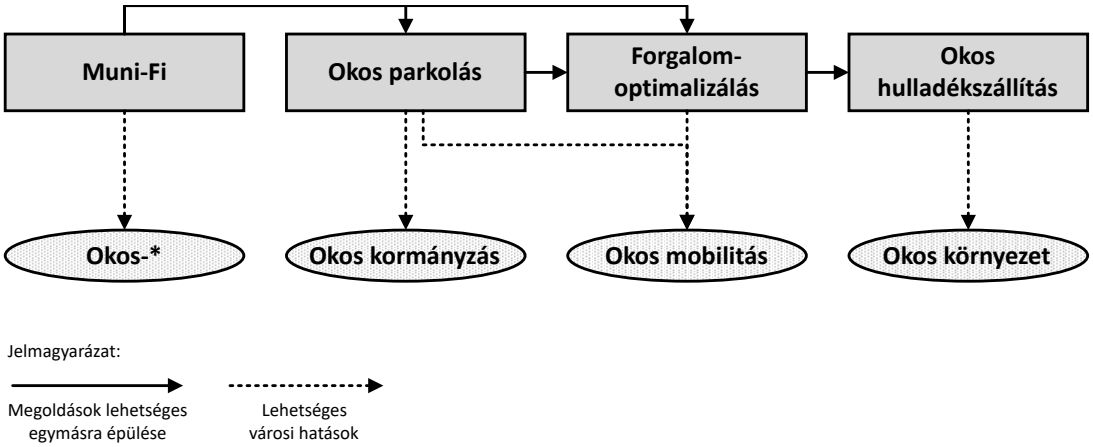
2.2. Okosváros-megoldások és okosváros-stratégia

Különálló okosváros-megoldásokat önmagukban is be lehet vezetni, inkrementális javulást elérve ezzel a városi élet valamely területén. A nemzetközi példák azonban arra engednek következtetni, hogy helyes városi szintű stratégiai célokat megfogalmazni, és az okosváros-prioritásokat és -funkciókat ezek fényében meghatározni. Ennek egyik alapvető oka, hogy az okosváros-megoldások képesek egymás hatásának felerősítésére, egyfajta „okosváros-képesség” hálózatba szerveződve.

Ez jól szemléltethető alábbi, viszonylag egyszerű példáink segítségével is.

- A nyílt és mindenütt jelen lévő kommunikációs hálózati elérés lehetővé teszi vagy elősegíti a többi megoldást.
- Az okos parkolás eszközöket adhat a forgalomoptimalizáláshoz; pl. dugócsillapítás a vezetőknek P+R lehetőségek adhoc felvetésével mobiltelefonon keresztül.
- A hulladékbegyűjtés ütemezésének optimalizálásában pedig segíthet a robosztus forgalom-előrejelzés és -figyelés.

Ezeket az erős szinergikus hatásokat szemlélteti az 1. ábra. Figyeljük meg, hogy egyes megoldások több területen is kifejthetnek hatást. Például a parkolás igazságossága és transzparenciája a magas népsűrűségű nagyvárosokban közügy, bármely megoldás, amely a szűkös és korlátozott erőforrás hozzáférésének kezelését az optimalizálás mellett „társadalmiasítja” is (lásd lentebb), legalább másodlagosan okoskormányzás-megoldásként is értelmezhető.



1. ábra. Okos város megoldások szinergiájának illusztrációja
Forrás: a szerzők szerkesztése

Azt is megfigyelhetjük, hogy a megoldások sikeressége több szereplőtől is függ; általánosságban mind a városvezetés és -üzemeltetés, mind a gazdaság szereplői, mind pedig a polgárok támogatása és cselekvő közreműködése szükséges. Ez természetesen nem kell, hogy minden egyes okosváros-megoldásra teljesüljön, de az okos város mint összetett rendszer szintjén mindenképp igaz.

Például az okos parkolás egy modellje lehet, hogy a városvezetés stratégiai, szabályozási és kommunikációs támogatása mellett a rendszer kiépítését és üzemeltetését piaci szereplők végzik; emellett azonban szükséges az is, hogy a polgárok a rendszert elfogadják és tevékenyen használják.

A város mint a parkolási erőforrások jó részének szabályozója és tulajdonosa dönthet alapvetően más innovációs modell mellett is. A mobileszközök és a blokklánc (Blockchain) technológiák elterjedésével ma már műszakilag megoldható a „közösségi parkolás" több modellje is, ahol a parkolóhelyek (előre) foglalását, a fizetést és akár a büntetések kiszabását nem egy piaci szereplő végzi, hanem a város polgárainak közössége együttesen, egy blokkláncházat felett. Mint említettük, ez a modell ráadásul a polgárok felé való átláthatóság miatt komoly hosszabb távú társadalmi előnyökkel is járhat a transzparencia és a kölcsönös bizalom kultúrájának terjesztésével. (A blokklánc-technológia robbanásszerű elterjedésével ilyen rendszerek már ma is léteznek korai szakaszban, piaci kiforratlanságuk miatt tényleges példákat azonban nem közlünk.)

Érdemes azt is megemlítenünk, hogy a megfelelő technológiai platform egyben lehetővé teheti azt is, hogy a különböző szereplők véges erőforrásai – pl. utcai parkolóhelyek, piaci alapon működő parkolóházak és akár a polgárok magántulajdonában álló parkolóhelyek – közösítve álljanak egy okosparkolás-megoldásban rendelkezésre. Egy ilyen megoldás felett számtalan olyan üzleti modell létezik, amelyben minden szereplő „nyer"; már önmagában a különböző árazású parkolóhelyek közös kereshetősége és foglalhatósága is hatékonyabbá tudja tenni a teljes piacot az információhiány radikális csökkentésével.

2.3. Okos város és integrált településfejlesztési stratégia

Már egyszerű példáink is érzékeltetik, hogy a sikeres stratégiába rendezett okosváros-megoldások átalakító ereje hatalmas lehet. Egyben igaz azonban az is, hogy a megfelelő – az adott város kihívásaira válaszoló és jövőképét támogató – stratégiát megtervezni és megvalósítani összetett feladat; hazánkban ehhez az 56/2017. (III. 20.) kormányrendelet által előírtan alkalmazandó módszertani útmutató ad segítséget [LECHNER 2017].

A magyar kormányrendeleti szabályozás és a módszertani útmutató az „okos város” tulajdonságot az integrált településfejlesztési stratégia okosváros-módszertan megfelelőségéhez köti. A településfejlesztésnek így a város „*természeti és épített környezetét, digitális infrastruktúráját, valamint a települési szolgáltatások minőségét és gazdasági hatékonyságát korszerű és innovatív információtechnológiák alkalmazásával, fenntartható módon, a lakosság fokozott bevonásával*” kell fejlesztenie ([LECHNER 2017] nyomán).

A nemzetközi szabványokat és jó példákat is figyelembe vevő módszertant teljességében itt nem áll módunkban bemutatni és jellemezni, néhány alapvető megállapítása azonban jelen munka témáját közvetlenül érinti.

Kiemelkedő jelentőségű, hogy „okos várossá válni” a településfejlesztés- és rendezés rendszerébe ágyazott, összehangolt folyamat kell hogy legyen. A településnek

- létre kell hoznia a megvalósítás keretrendszerét (szervezeti keretek, üzleti modell),
- helyzetelemzéssel meghatározni a fejlesztési lehetőségeket,
- okos város stratégiát kell létrehoznia,
- cselekvési tervet kell készítenie
- és a megvalósult fejlesztéseket monitoring folyamatokkal követnie, ellenőriznie.

Egyben kimondásra kerül az is, hogy „okos városnak lenni egy folyamat, a folyamatos fejlesztés útja”. A módszertan meghatározza az „okos város folyamat” alapvető szerepköreit is:

- Lakosság, helyi cégek, intézmények: az okos város „végfelhasználói” és haszonélvezői, de megfelelően „társadalmiasított” megoldásokkal cselekvő közreműködésükön keresztül annak erőforrásaivá is válhatnak.
- Önkormányzatok, önkormányzati vezetés: az a szereplő, akinek az okos várossá válás folyamatának elindítása, koordinációja és irányítása lehetőségében áll. Az általánosabb, a nemzetinél megengedőbb definíciókat figyelembe véve egy város válhat részlegesen okossá a többi szereplő „okos” önszerveződésén keresztül, önkormányzati szerepvállalás nélkül, egyfajta „emergens” tulajdonságként is. Az okos város teljes potenciáljának megvalósításához azonban az önkormányzat a kritikus szereplő.
- Piaci vállalkozások: részben haszonélvezői, részben pedig – megoldásszállítóként és fejlesztőként – tevékeny alakítói az okos városnak.
- Központi államigazgatás: a megfelelő nemzeti szintű szabályozói, finanszírozási, szabványosítási és intézményi keretek megalkotásával támogatja a helyi szintű okos településfejlesztést. Emellett olyan szolgáltatásokat biztosít, amelyek helyi megvalósítása kevésbé lenne gazdaságos.

A módszertan kulcseleme, hogy a teljes folyamat során – felméréstől a stratégiai tervezésen keresztül az üzemeltetésig és utókövetésig – előírja a meglévő és elképzelt városi szolgáltatások, összefüggéseik és átfedéseik azonosítását és követését (ennek előnyeit szemléltette az előző alfejezet).

A szolgáltatásokból és az azokat üzemeltető szereplőkből egy szolgáltatási mátrix is szervezendő, amely átfogó képet ad a „párhuzamos” képességekről. Ezek megléte nem feltétlenül hátrány. Mint később látni fogjuk, a szolgáltatások mennyiségi és minőségi redundanciája az okosváros-megoldások teljesítmény- és ellenállóképesség-menedzsmentjének egyik pillére is lehet – mind az IKT-erőforrások, mind a szolgáltatások szintjén.

3. Az okos várost lehetővé tevő IKT-képességek

Az információ- és kommunikációtechnológia néhány jól körülírható, az utóbbi évtizedben megjelent, illetve jelentős fejlődésen átesett területe kulcselemét képezi az okosváros-megoldásoknak. Röviden felsoroljuk és jellemezzük az általunk legfontosabbnak tekintett területeket. Érdemes megjegyeznünk, hogy annak ellenére, hogy fejlődésük mind a mai napig folyamatos, az informatikának ezek a különálló tudományágai már elég érettek ahhoz, hogy a közvetlen, alkalmazási célú innováció részét képezzék.

3.1. Mindenhol elérhető kommunikáció

Városi környezetben ma már elfogadható feltételezés, hogy szinte mindenhol létesíthető vezeték nélküli kapcsolat, amelyen keresztül elérhető az internet. Ma ez még jellemzően 3G/4G szolgáltatásokat jelent, illetve jóval kevésbé elterjedten Muni-Fit (jó közelítéssel: városi tulajdonú, széles lefedettségű wifiszolgáltatást).

A terepi eszközök és szenzorok sokszor az internettől elhatárolandó kommunikációját ezek a megoldások jellemzően viszonylag rosszul – költségesen, magas késleltetéssel és energiaigényesen – támogatják; ezeket az akadályokat azonban a néhány éven belül kiépülni kezdő 5G hálózatok szinte teljes mértékben meg fogják oldani. Emellett a szenzorok terepi kommunikációjára és a kommunikáció összegyűjtés után átjárón keresztül vezetékes/normál mobil hálózatra átirányítására már ma is léteznek kiforrott megoldások.

Az ötödik generációs hálózatok jellemzően városi környezetben fognak megjelenni, és várható, hogy elsődlegesen a városok jellemzője lesz a mindenhol elérhető szélessávú mobilkommunikáció is (lásd például [ERICSSON 2017]).

3.2. „A dolgok internete” – Internet of Things (IoT)

A mindenhol elérhető kommunikáció, valamint az elektronika és szoftvertechnológia fejlődése lehetővé tette, hogy a fizikai világ szinte minden tárgyát az internetre legyünk képesek csatlakoztatni, egyre költséghatékonyabban. Ezáltal nemcsak a „dolgokra” szerelt szenzorok által szolgáltatott adatokhoz jutunk, de képessé válhatunk beavatkozásokra is.

Nem feltétlenül egyértelmű, hogy a korábban is létező, biztonságkritikus városi infrastruktúrák (pl. közlekedési lámpák) szélesebb körű, de még mindig nem nyílt hozzáférésű hálózatba kötése tekinthető-e IoT-megoldásnak; a szerzők véleménye szerint nem. Az okos épített környezet – szenzorokkal felszerelt és részlegesen „távvezérelhető” épületek, közművek, közterületi bútorok, közterületi világítás, úttest, hirdetőoszlopok stb. – viszont mindenképp az IoT példái is.

A terepi és mobil eszközök azonban az IoT-nek csak az egyik kulcsfontosságú elemét adják. A rengeteg IoT-eszköz által folyamatosan létrehozott, óriási mennyiségű megfigyelés kezelését és gyors – időszerű beavatkozásokat lehetővé tevő – elemzését a felhő-számítástechnika és a Big Data adatelemzés teszik lehetővé (a felmerülő kihívásokkal kapcsolatban bővebben lásd pl. [LEE et al. 2014]).

Az IoT-nek mint jelenségnek van egy talán ritkábban kiemelt hajtóereje is: a szenzoripar fejlődése. A „dolgok” internetre kötéséhez nem csak apró, olcsó, „elég erős” és alacsony fogyasztású „számítógépekre” (mikroprocesszorokra és mikrokontrollerekre) van szükségünk, hanem költséghatékony és „elég jó” szenzorokra is. Ezek nemcsak hogy rendelkezésünkre állnak ma már, de jó néhány, korábban legfeljebb katonai alkalmazásokban felbukkanó szenzortípus is megjelent a tömegpiacon, egy-két évtizeddel korábbi áránál szó szerint nagyságrendekkel olcsóbban (mint például a nagyfelbontású hőkamerák és a lézeralapú távérzékelés – LIDAR).

3.3. Mobil számítástechnika

A mobil számítástechnika – legfőképp az okostelefonok – rohamosan növekvő penetrációja ma már közhely; 2016-ban például csak Magyarországon már 3,3 millióan használtak okostelefont [BBJ 2016]. A mobil számítástechnika jelentősége az okos városban alapvető. Ez teszi lehetővé, hogy a polgárok mobilalkalmazásokkal vagy weboldalakon keresztül akármikor kölcsönhatásba tudjanak lépni egy okosváros-megoldással, még akkor is, ha annak fizikai kezelőfelület nem része, vagy a város adott pontján nem elérhető. Mobil számítástechnikára épülő okosváros-megoldásként értelmezhetők olyan, ma már sokszor magától értetődőnek vett rendszerek, mint például a valós idejű tájékoztatás a tömegközlekedési eszközökről (lásd Budapesten a BKK FUTÁR alkalmazást [BKK s. a.]).

A mobil számítástechnika kulcseleme az „osztozáson alapuló gazdaságnak” (sharing economy) is; véges erőforrások és javak közvetlen („peer to peer”) bérlésének, cseréjének, ajándékozásának és adásvételének (lásd pl. [HAMARI–SJÖKLINT–UKKONEN 2016]). A sharing economy modellben nem egy harmadik félen keresztül történnek az ügyletek, bár az ügyletek lebonyolítását lehetővé tevő informatikai platformot – pl. weboldalt – adhatja egy harmadik fél. A véges városi erőforrások – például különböző közlekedési eszközök vagy a parkolóhelyek – különböző „megosztott” használati modelljei egyértelműen okosváros-megoldások.

A mobil számítástechnikában azonban bőven rejlenek még további, eddig jórészt kihasználatlan lehetőségek is, pl. a „community sensing”-en keresztül – a mobileszközök szenzoraival (ide tartozik a kamera és a mikrofon is!) a polgárok csoportjai akárhol és akármikor tudnak megfigyeléseket eszközölni (lásd például [LANE et al. 2010]). Ez lehetővé teszi, hogy a város életének aktív közreműködőivé váljanak, akár a hétköznapokban (pl. városi környezet megóvása és karbantartása), akár balesetek és katasztrófák esetén.

3.4. „Nagy adat” adatelemzés

Az informatikai ipar egyes szereplői a 90-es évek végétől kezdve kezdtek úgynevezett „Big Data” kihívásokkal szembesülni: olyan nagy mennyiségű adat tárolását és elemzését igényelték, ami a klasszikus adatbázis-kezelőkkel nem volt hatékonyan (és észszerű költségekkel) megoldható.

Eredetileg ezek az adatkészletek jellemzően informatikai forrásokból származtak: jellegzetes példák az internet teljes szöveges tartalma (keresőmotorok esetén), nagy forgalmú weboldalak felhasználói „kattintásainak” a követése, vagy éppen a közösségi oldalak kapcsolati hálóinak alakulása. Kihasználva, hogy ezek „nyugalomban lévő” („at rest”), azaz a továbbiakban módosítást már nem igénylő adatok, kialakultak azok a módszerek, amelyekkel az adattárolás és -elemzés olcsó számítógépek farmjain is megvalósítható.

Ma már kiforrott nyílt forráskódú Big Data eszközök széles palettája áll rendelkezésünkre, és ezeket akár felhőben is futtathatjuk – fizikai gépek helyett olcsó, átmenetileg bérelt virtuális gépeken. A Big Data adatfeldolgozásnak létrejött egy másik ága is: a folyamfeldolgozás (stream processing), amikor a nagy mennyiségű adatot folyamatosan, beérkezésével egy ütemben kell feldolgoznunk.

Az IoT-forradalom egyik hajtóereje, hogy az IoT-eszközök által létrehozott megfigyelések kezelése alapvetően Big Data probléma: vagy tárolni akarjuk őket a későbbi módosítások igénye nélkül, és időről időre teljességükben elemezni, vagy folyamatosan, minél kisebb késleltetéssel feldolgozni.

A Big Data szerepe az okos város IKT-támogatásában is egyértelmű, még IoT-értelmezés nélkül is: korábbi példáinkban akkor tudjuk a megoldásokat igazán „okossá” tenni, ha a releváns megfigyeléseket (parkolások, járművek mozgása, hulladékgyűjtő edények telítettsége stb.) hosszabb időtávon, nagy felbontással tároljuk, majd a megoldások működését a tárolt adatok elemzése alapján optimalizáljuk. A kritikus, azonnali beavatkozást igénylő helyzetek felismerésére pedig szintén a Big Data adatfeldolgozás adhat választ.

3.5. Felhő-számítástechnika

Szakmai körökben közismert bonmot, hogy a felhő-számítástechnika nem más, mint „valaki más számítógépének használata”. Az okos városokban – valamint az IoT- és kiberfizikai rendszerekben (lásd később) – betöltött alapvető szerepe miatt a felhő-számítástechnikát a valóban okos várost lehetővé tevő többi informatikai tudományágnál részletesebben ismertetjük.

3.5.1. Szolgáltatásmodellek

Definíció szerint a felhő-számítástechnika (cloud computing) „*egy [szolgáltatási] modell, amely lehetővé teszi a hálózaton keresztül való, kényelmes és széles körű hozzáférést konfigurálható számítási erőforrások egy megosztott halmazához*” (az NIST 800-145 szabvány [PETER–GRANCE 2011] definíciójának értelmező fordítása).

A pontos, szakmai közmegegyezésen alapuló, de nem épp szemléletes kanonikus meghatározás közérthetővé válik, ha a felhő-számítástechnika legfőbb típusait (a szabványos szóhasználatban „szolgáltatási modelljeit”) külön-külön tekintjük.

3.5.2. Infrastruktúra mint szolgáltatás (IaaS)

Az Infrastructure as a Service, az IaaS alapgondolata, hogy ahelyett, hogy saját kiszolgálókat (szervereket) szereznénk be és saját adatközpontban helyeznénk el őket, egy szolgáltató szerverfarmjában bérlünk szerver-„szeleteket”, amelyek számítástechnikai szempontból majdnem pontosan úgy viselkednek, mintha a saját számítógépeink lennének. A gépekhez az interneten keresztül férünk hozzá, és csakúgy, mint egy saját adatközpontban, telepíthetünk rájuk a felhasználók által az internen keresztül elért kiszolgálóprogramokat és alkalmazásokat is. Emellett természetesen kialakíthatunk korlátozott, nem nyilvános hozzáférésű rendszereket is – például adattárolásra és -elemzésre.

A szolgáltató szervereinek „szeletelése” az úgynevezett virtualizációs technológiák segítségével történik; ezek lehetővé teszik, hogy egy számítógép egyszerre több, egymástól független (és egymást befolyásolni nem képes) „virtuális” számítógépet (virtual machine, VM) futtasson.

A virtualizációnak köszönhetően a fizikai erőforrásokat jellegzetesen a többi „bérlővel” (tenant) megosztva használjuk. Pl. két különböző bérlő egy-egy VM-e ugyanazon a kiszolgálón osztozik a processzoridőn; azonos erőforrás-kiosztásnál mindkét VM elég jó közelítéssel azt érzékeli, mintha egy fele olyan gyors processzoron futna, mint a valódi fizikai gépbe épített processzor.

Az IaaS-szolgáltatások magját a virtuális gépek adják, de a teljes szolgáltatási ökoszisztéma tartalmazza – jellemzően virtualizálva – mindazon egyéb elemeket, amelyek egy saját adatközpontban is szükségesek (virtuális hálózat, hálózatra csatolt tárolók, terheléskiegyenlítők, bérelhető nyilvános hálózati címek stb.)

3.5.3. Platform mint szolgáltatás (PaaS)

A Platform as a Service, PaaS-modellben nem teljes virtuális gépeket bérlünk, amelyeken aztán mi alakítjuk ki az alkalmazásaink futtatásának képességét, hanem kész alkalmazásfuttató környezeteket; bérlőként mi „csak” magát az alkalmazási logikát biztosítjuk. Az alkalmazásunk mögé rendelt erőforrásokat (virtuális gépek, hálózati kapacitások stb.) a szolgáltató kezeli, legtöbbször dinamikusan, az alkalmazásunk terhelése alapján.

3.5.4. Szoftver mint szolgáltatás (SaaS)

Kész, esetlegesen konfigurálható szoftverszolgáltatásokhoz igény szerinti hozzáférés, ahol az igénybe vehető kapacitás skálázható. A Software as a Service, a SaaS klasszikusan webalkalmazásokat – böngészőből, weboldalként használható szolgáltatásokat – jelentett. Erre példa a SaaS-ként igénybe vett levelezés és egyéb irodai jellegű funkciók (kollaboratív szövegszerkesztés, telekonferencia stb.); a kapacitásskálázás itt azt jelenti, hogy szervezetünkön belüli felhasználókat könnyen tudunk felvenni és eltávolítani, felhasználószám és használati intenzitás alapján fizetve a szolgáltatásért.

Emellett azonban az SaaS-kategória alá sorolható jó néhány olyan szolgáltatástípus is, amelyek az okosváros-megoldások szempontjából jóval nagyobb horderővel bírnak. Egy SaaS-szolgáltatás jelenthet olyan összetett szoftverrendszereket, amelyeket elméletileg magunk is létrehozhatnánk (akár saját, akár az IaaS-modellben bérelt erőforrásokon), de szervezetünk erre szakértelem hiányában nem képes, vagy nem kívánjuk véges emberi erőforrásainkat ezen rendszerek létrehozására és üzemeltetésére fordítani.

Jellemző példáját adják ennek a Big Data és az IoT-t támogató megoldások: ezeket a bonyolult szoftverrendszereket használatra készen kínálják a meghatározó felhőszolgáltatók. Ezen elemeket egy összetett megoldásban legtöbbször nem is saját (minimális) felhasználói felületükön keresztül használjuk, hanem egy elosztott rendszerbe beágyazva, úgynevezett alkalmazásprogramozási csatolófelületen (application programming interface, API) keresztül elérve. Jellegzetes példa, hogy egy SaaS-ként „bérelt” Big Data megoldást mobil eszközökön futó alkalmazások látnak el adatokkal, az elemzéseket a felhasználó pedig egy erősen vizuális, üzleti analitikai szemléletű webfelületről indítja (amelynek kiszolgálómegvalósítása is futhat persze „a felhőben”).

Másrészről egyre inkább a SaaS-modellben érhetőek el olyan szolgáltatások, amelyek erősen specializált, különösen mély hozzáadott értéket képviselnek. Ennek egy példája az automatizált adatelemzés, amelynek során egy intelligens szolgáltatás önműködően deríti fel egy adatkészlet statisztikai jellemzőit és az adatkészletben megbúvó lehetséges összefüggéseket (lásd például [HOYT et al. 2016]).

3.5.5. A felhő-számítástechnika néhány alapvető jellemzője

A felhőszolgáltatásoknak van néhány jellemzője, amelyekben alapvetően különböznek a saját erőforrásokkal megvalósító rendszerektől, s amelyek kihatással vannak az okos város kontextusban alkalmazásukra is. Ezek közül röviden áttekintjük a jelen tanulmány témáját érintőket. A felhő-számítástechnika teljes körű alkalmazhatósági és összehasonlító elemzésére azonban itt nem vállalkozunk, e célból az érdeklődő olvasó figyelmébe ajánljuk Joe Weinman közismert – és közérthető – munkáját [WEINMAN 2012].

Elasztikus skálázás. A bérelt szolgáltatást – legyen az virtuális számítógépek fürtje, vagy egy Big Data platform – „elasztikusan" tudjuk skálázni: könnyen és viszonylag gyorsan igényelhetünk további erőforrásokat és szabadíthatjuk fel a feleslegessé váltakat. Míg a kezdetekben az elasztikus skálázás alkalmazásának elsődleges célja a költséghatékonyság-optimalizálás volt, mint a későbbiekben azt megmutatjuk, ez a képesség kiemelkedően alkalmas a szolgáltatások ellenállóképességének javítására is.

- **Használatalapú számlázás**: a felhőszolgáltatások igénybevétele alapvetően használat alapján történik; annyi gépidőért, tárhelyért, elemzési időért stb. fizetünk a szolgáltatónak, amennyit az adott időszakban „használtunk" (bár nem feltétlenül kihasználtunk).
- **Erőforrások megosztása**: a szolgáltató a saját erőforrásait a bérlők dinamikusan változó igényei között szintén dinamikusan, mint egy kvázi egységes „erőforráspark" (resource pool) osztja meg; ez jelentheti azt, hogy bérlők – a tudtukon kívül – számítógépeken és más fizikai erőforrásokon egyidejűleg osztoznak. Ennek előnye, hogy a szolgáltató igen magas erőforrás-kihasználtságot tud megvalósítani, ám hátránya, hogy az egyidejű használat teljesítmény- és biztonsági problémákhoz vezethet.
- **Széles körű hálózati hozzáférés**: a felhőszolgáltatások jellemzően az internetről érhetők el, a szolgáltató saját adatközpontjaiban üzemelő kiszolgálókon. A nagy felhő-szolgáltatók globálisan jellemzően több, akár tucatnyi adatközpontot üzemeltetnek. Ennek közvetlen folyománya, hogy egy informatikai megoldásba integrációjuk hálózati szinten ténylegesen pusztán az egyszerű internetelérés követelményének teljesülésétől függ. Az interneten keresztül való elérés árnyoldalai – kommunikációs hibák, kiszámíthatatlan késleltetés, biztonsági kérdések – természetesen jelentkeznek, de a legtöbb esetben ezek műszakilag kezelhető kihívások.

3.5.6. Telepítési modellek

Bár a felhőszolgáltatások tárgyalását azzal kezdtük, hogy a felhő „valaki más számítógépe", e kijelentésünkön érdemes finomítanunk. A szolgáltatási modellek mellett a felhőszolgáltatásoknak megkülönböztetjük úgynevezett telepítési (deployment) modelljeit is. Itt a három fő kategória a nyilvános (public), a magán (private) és a közösségi (community) felhők.

Az eddigi tárgyalás implicit feltételezte a nyilvánosfelhő-modellt. A nyilvános esetben a felhőszolgáltatást egy profitorientált felhőszolgáltató végzi úgy, hogy valódi használaton alapuló pénzügyi elszámolás történik. Ma már rengeteg szolgáltató nyújt így felhőszolgáltatásokat; ezek közül a két legnagyobb és leginkább reprezentatív példa az Amazon AWS (Amazon Web Services) és a Microsoft Azure platformja.

Felhőszolgáltatást azonban egy szervezet önmagának is létrehozhat, kizárólag a saját, belső felhasználói számára; ezeket hívjuk privát felhőknek. Egy ilyen rendszer kialakításának fő célja általában az, hogy növelje a számítási erőforrásokhoz való hozzáférés hatékonyságát és az erőforrások kihasználtságát.

A számítási felhők kialakítása megfelelő szoftvereszközöket igényel. A fizikai platformok és erőforrások megosztható hozzáférésének biztosítására különböző virtualizációs megoldásokat alkalmazunk. Ezekre épülnek a felhő-szoftverkeretrendszerek, mint például az OpenStack, amely jelenleg a legelterjedtebben használt nyílt forráskódú IaaS-keretrendszer.

A közösségi felhők bizonyos értelemben a nyilvános és privát felhők mint két véglet között helyezkednek el. Mint nevük is mutatja, ezek szervezetek egy (zárt) közösségét szolgálják ki, adott esetben a szervezetek számítási erőforrásait összevonva egy felhőbe.

3.5.7. Felhőszolgáltatások és okosváros-megoldások

Az egyes megoldások szemszögéből vizsgálva nyilvánvalóan az adott okosváros-megoldás tulajdonságaitól függ, hogy a felhőszolgáltatások alkalmazása indokolt vagy szükséges-e. Korábbi példáinkra mind igaz, hogy az adatbiztonsági megfontolásoktól eltekintve rendszertervezői szemszögből nehezen indokolható a felhőplatformot és -szolgáltatásokat használó kulcselemek kizárása.

Valójában a fő kérdés nem is az, hogy az okosváros-megoldások erőforrás-igényes, központosított háttérszolgáltatásait felhőplatformra helyezzük-e, hanem hogy milyen telepítési modell mellett döntsünk. Megfelelő törvényi háttér és „ügyfélméret” esetén a piaci felhőszolgáltatók képesek és hajlandók olyan, számítástechnikailag leválasztott és a megfelelő folyamatok segítségével üzemeltetett belső rendszereket kialakítani, amelyek kiemelten szigorú törvényi és műszaki adatbiztonsági követelményeket is kielégítenek. Ennek közismert példája, hogy az Amazon nyújt felhőszolgáltatást az Egyesült Államok kormányzati szervei részére is.

Az okosváros-stratégiának lehet kulcseleme egy, azt dedikáltan támogató privát felhő kialakítása is. Mint azt szemléltettük, az okos város egymást támogató okosváros-megoldások hálózataként is felfogható; amennyiben ezek irányelvként a közös, városi (vagy állami) szintű privát felhőt használják, a felhő szolgáltatói gazdaságosságához szükséges méret elérése szinte triviális (a felhő-méretgazdaságosság befolyásoló tényezőivel kapcsolatban bővebben lásd [WEINMAN 2012]).

Ebből adódóan – hasonlóan a mindenhol elérhető kommunikációhoz – önmagában a felhő kialakítása is egy okosváros-megoldás. Ennek közvetlen, működő előképei hazánkban is vannak; „okoskormányzat”-megoldásként a Kormányzati Felhő [KOF s. a.], „okos emberek” kontextusban pedig a felsőoktatás – akár új oktatási modellek mentén – felhő módszerekkel támogatása [KOCSIS et al. 2014].

3.6. A műszaki innováció következő hulláma

A jelen fejezetben eddig bemutatott IKT-képességek közös jellemzője, hogy okosváros-megoldásokban alkalmazásuk már ma is bevett gyakorlatnak tekinthető. Az okos város műszaki innováció következő hullámában várhatóan a következő IK-képességek lesznek meghatározók.

A blockchain-technológiák új, decentralizált kooperációs platformot hoznak létre, amely a széles körű társadalmi önszerveződés lehetőségét is magában hordozza [Biswas–MUTHUKKUMARASAMY 2016, SUN et al. 2016].

A felhő-számítástechnika kiterjesztéseként a rövid időn belül a megvalósuló alkalmazásokban is meg fog jelenni a „köd”-számítástechnika (fog computing); ennek lényege, hogy a terepi eszközöket, átjárókat és a „klasszikus” felhőfunkciókat képes egységes felhő szemléletben kezelni [BONOMI et al. 2012].

A neurális hálók modern alkalmazásán alapuló deep learning nem csak az „adatközponti” mesterséges intelligenciában jelent komoly előrelépést (az arcfelismeréstől a parkolók használati mintázatainak előrejelzéséig). Azt is lehetővé teszi, hogy mesterséges intelligenciát terepi eszközökön – például okoskamerákon – közvetlenül, helyben alkalmazzunk [AMATO et al. 2017].

4. Teljesítménytervezés és -menedzsment: motiváció

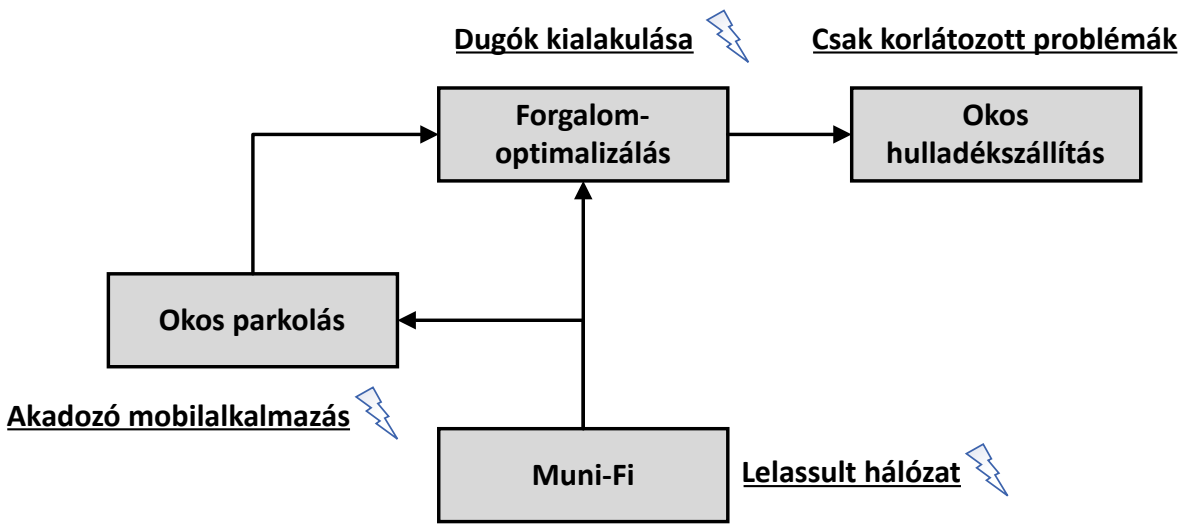
Az informatika mérnöki gyakorlatában a számítógépes rendszerekkel megvalósított szolgáltatások tulajdonságait általában kettébontjuk funkcionális és extra-funkcionális jellemzőkre.

Rendszereink ugyanis megcélzott funkcionalitásuk megvalósítása mellett egyéb jellemzőkkel is rendelkeznek, amelyek számszerűsíthetően és mérhetően rendszer-, illetve szolgáltatás-„jóságot”, -„minőséget” fejeznek ki – de legalábbis lehetővé teszik a funkcionalitáson túlmutató összehasonlíthatóságot. Klasszikusan [AVIŽIENIS et al. 2004] az alapvető rendszerjellemzők közé soroljuk a funkcionalitáson túl a teljesítményt (performance), szolgáltatásbiztonságot (dependability), adatbiztonságot (security) és a költséget (cost). Ezen jellemzők gyűjtőfogalmak; definíciójuk és értelmezésük részattribútumaik meghatározásával történik, figyelembe véve a műszaki és üzleti kontextust.

A költség, a szolgáltatásbiztonság és az adatbiztonság alapvető jelentősége az okosváros-megoldásokban talán minden olvasónk számára egyértelmű; itt most arra kísérlünk meg rámutatni, hogy az IKT-elemek teljesítményének szerepe minőségileg más lehet az okosváros-rendszerekben, mint a „szokásos” informatikai megoldásoknál.

4.1. Teljesítményhibák okosváros-megoldásokban

Tekintsük a korábban az okosváros-szolgáltatások egymásra épülésének szemléltetésére hozott példánkat, és tételezzük fel, hogy az okosváros-megoldások közös, vezeték nélküli kommunikációs rétege radikálisan „lelassul”!



2. ábra. Teljesítményhibák hatásának terjedése egy okosváros-megoldáshálóban

Forrás: a szerzők szerkesztése

A hálózati kommunikáció lelassulása a „klasszikus” informatikai szolgáltatásoknál – az egyszerű weboldalaktól a közösségi médián keresztül a vállalati rendszerekig – is probléma; közel sem véletlen, hogy a teljesítmény menedzselésének gyakorlata és irodalma több évtizedet ölel fel. A fenti esetekben azonban a teljesítmény romlása legfeljebb a felhasználói elégedettséget csökkenti, annak minden „üzleti” hatásával. Más a helyzet az okos városban.

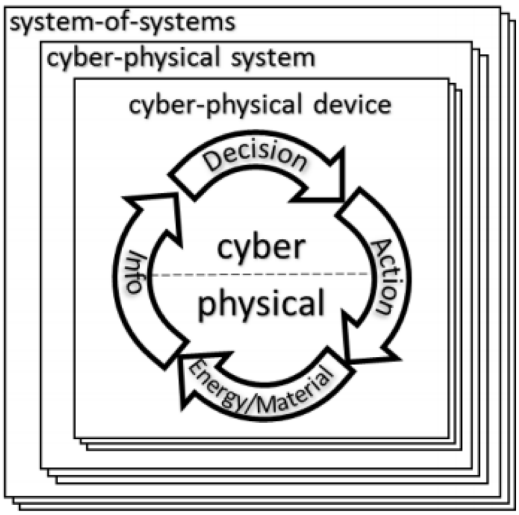
- A hálózat lelassulása mint hiba továbbterjed az okos parkolásra; az akadozó mobilalkalmazás miatt a felhasználók nem tudnak parkolóhelyet keresni és foglalni.
- A hálózat lassulása következtében a valós idejű forgalomoptimalizálás nem rendelkezik a szükséges áttekintő képpel a város forgalmi helyzetéről; legrosszabb esetben ez oda vezethet, hogy olyan közlekedési torlódások alakulnak ki, amelyeket a meglévő rendszerek el tudnának kerülni. Az okos parkolás akadozása közvetve is hozzájárulhat a probléma kialakulásához, bár jóval kisebb mértékben.
- Az okos hulladékszállítás mindeközben valószínűsíthetően csak korlátozott problémákkal szembesül. Ezen a területen a fizikai folyamatok – magyarul: a szemétgyűjtő edények „telítődése” – elég lassúak ahhoz, hogy az okosváros-megoldásra csak hosszan fennálló hálózati problémák legyenek komoly hatással.

A példa fő üzenete: egy okosváros-megoldás egy elemének informatikai teljesítményproblémái a fizikai világban okozhatnak kritikus helyzeteket; minőségileg más kockázatokat gerjesztve, mint akár a korábbi korszakok „üzletmenet-kritikus” informatikai megoldásai. Mindez pedig abból adódik, hogy az okosváros-megoldások IKT-rendszerei a fizikai világ megfigyelése alapján működnek, közvetlen hatással lehetnek a fizikai világra, és egy rendszerekből álló rendszerbe állnak össze.

Ebben az értelemben az okos város egy nagyobb, modern informatikai rendszerkategória egyik kiemelt fontosságú alesete: az úgynevezett kiberfizikai rendszereké (Cyber-Physical Systems, CPS). A kiberfizikai rendszerek „olyan okos rendszerek, melyek fizikai és számítási elemek kölcsönhatásban álló hálózatait tartalmazzák” [Cyber Physical Systems PWG 2015]. A „szabályozási kör”-jelleg mellett kulcsfontosságú jellemzőjük, hogy időérzékenyek, és nagyobb „rendszerek rendszerébe” (System of Systems, SoS) is rendeződhetnek (lásd 3. ábra). A CPS mint rendszerkategória ma kiemelt figyelmet élvez a kutatásban, a kutatásfejlesztésben és a szabványosításban; gyakran, de nem kizárólag okos város hangsúllyal. (CPS-nek tekinthetőek még többek között a modern gyárak, az autonóm repülő eszközök rajai stb.)

Példánk egy másodlagos üzenettel is rendelkezik. Okosváros-megoldások egy hálózatában lehetséges logikai következtetéssel elemezni, hogy egy megoldás teljesítményproblémái milyen hibákat okoznak a tőle közvetlenül és közvetve függő megoldások működésében.

A függőségi hálózatba szerveződő okosváros-megoldásokat felfoghatjuk klasszikus szemléletben: szolgáltatások függőségi hálózataként is. A szolgáltatásbiztos számítástechnika (dependable computing) a hibák terjedésének szolgáltatáshálózatokban, illetve komponensekből létrehozott rendszerekben kiértékelésére évtizedek óta alkalmaz úgynevezett hibaterjedés-analízis (Error Propagation Analysis, EPA) módszereket. Egy modern megközelítést ad például [PATARICZA 2007]. Ezek okosváros-környezetben alkalmazását egy későbbi kismonográfiában tervezzük tárgyalni.



3. ábra. Kiberfizikai rendszerek: kompozíciós és kölcsönhatási hierarchia
Forrás: [Cyber Physical Systems PWG 2015], 4. oldal

Példánk alapján elmondható, hogy egy okosváros-megoldásnak akár a városi élet szereplői, akár más okosváros-megoldások a felhasználói, a megoldás nem megfelelő teljesítménye komoly kockázatot jelenthet.

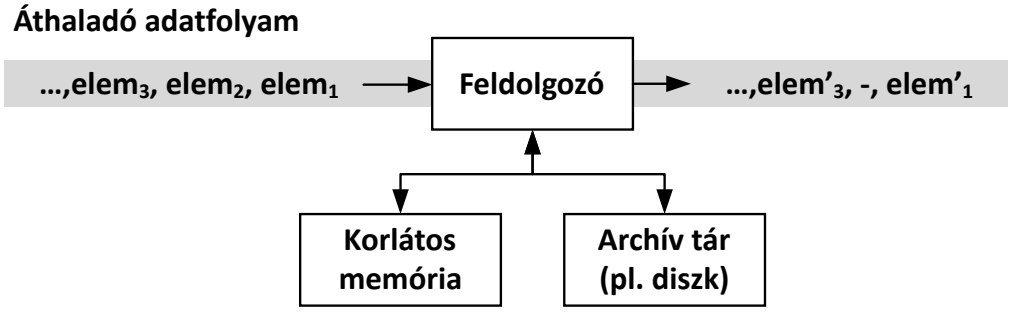
Az informatikában az ilyen jellegű kockázatok kezelésének számos megközelítése ismert. Ezekben közös, hogy a nyújtott (és igénybe vett) szolgáltatások teljesítményét egyértelműsítik, és a vállalt/igényelt követelményeket számszerűsítik.

Mielőtt ezek tárgyalására rátérnénk, egy példa segítségével megmutatjuk, hogy egy okosváros-megoldásba integrált IKT-szolgáltatás tipikusan milyen teljesítményjellemzőkkel rendelkezik, és hogyan függnek a követelmények az alkalmazástól. Szemléltetjük azt is, hogy a teljesítményhibák fő oka jellemzően a kapacitástúllépés.

4.2. Teljesítménykövetelmények és IKT-kapacitások

Szemléltető példánk az adatfolyam-feldolgozás, amelynek jelentőségét okos városban az IoT és Big Data IKT-képességeknél korábban röviden ismertettük.

Az adatfolyam-feldolgozás szokásos modelljében (lásd pl. [LESKOVEC–RAJARAMAN–ULLMAN 2014]) a feldolgozást úgynevezett feldolgozó elemek (processing elements) végzik, amelyek az adatfolyamot elemről elemre feldolgozzák; ha úgy tetszik, ezek az adatfeldolgozás logikai lépései. A feldolgozás logikáját sokszor úgy kell megvalósítani, hogy egy „legrosszabb végrehajtási idő” (worst case execution time, WCET) megadható legyen. A feldolgozási modellt a 4. ábra szemlélteti.

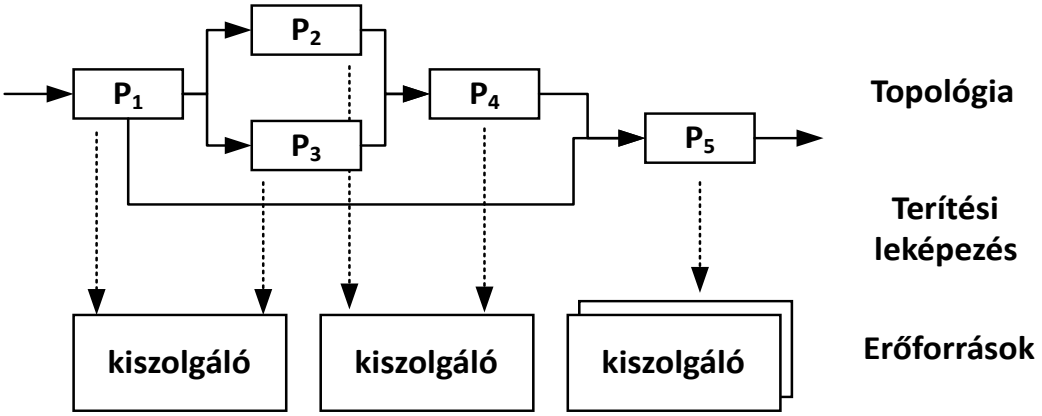


4. ábra. Adatfolyam-feldolgozóelem egyszerűsített modellje
Forrás: [LESKOVEC–RAJARAMAN–ULLMAN 2014], 132. oldal alapján a szerzők szerkesztése

Az elemi feldolgozó elemekből komponált feldolgozási logikát szokásosan topológiának hívjuk. A topológia elemi (logikai) feldolgozási lépéseinek többségét általában nem szükséges a fejlesztés során megvalósítanunk; a számítási modellt támogató keretrendszerek (vagy felhőszolgáltatások) részét szokta képezni a tipikus feladatokat elvégezni képes, konfigurálható feldolgozó elemek egy könyvtára. Néhányat bemutatunk ezek közül [BALLARD et al. 2010] alapján, egy „okos parkolóhely" példával megvilágítva, ahol feltételezzük, hogy egy központi folyamfeldolgozó szolgáltatás gyűjti az érzékelőkkel felszerelt parkolóhelyek üzeneteit, és végzi ez alapján a számlázást.

- Folyamszűrés (filtering): a feldolgozni nem kívánt elemek eldobása. Ilyen, legalábbis a számlázás szempontjából eldobandó elemek lehetnek az üres parkolóhelyek periodikus „szabad" jelzései.
- Kieső elemek (outlierek) szabályrendszer-alapú kiválogatása: a túl hosszú parkolási idő – pl. több mint egy hét – más jellegű kezelést kíván (adott esetben a gépjármű elszállíttatását), mint a normál üzem.
- Adatfolyamok dúsítása tárolt referencia, illetve kiegészítő adatokkal (supplemental data): a parkolóhelynél elhelyezett kamera által felismert rendszám alapján a megfigyelésnek a gépjármű tulajdonosának adataival való kiegészítése.
- Statisztikai modellek alkalmazása a folyamon: a tulajdonos, a gépjármű és az időszak alapján a várható fennmaradó parkolási idő előrejelzése annak meghatározására, hogy mennyi parkolókapacitást tegyen a rendszer előrefoglalásra elérhetővé.

Az ily módon összeállított logikai topológiát rendeljük a – fizikai vagy virtualizált – számítási erőforrásokhoz egy „telepítési", vagy „terítési" (deployment) jellegű leképezéssel. A leképezés során sok esetben lehetőségünk van logikai feldolgozási lépéseket többszörözötten, több erőforrásra is leképezni, ezzel növelve az adott logikai lépés kapacitását. Ezt szemlélteti az 5. ábra.



5. ábra. Adatfolyam-feldolgozás: topológiakompozíció és erőforrásokra terítés
Forrás: a szerzők szerkesztése

A terítési leképezés kulcsfontosságú fogalom az informatikában; ez teremti meg a kapcsolatot a funkcionalitás és a végrehajtás során rendelkezésre álló kapacitás között. A korábban felsorolt IKT-képességeket végső soron minden esetben fizikai számítógépek, vezetékes vagy rádiós hálózati kapcsolatok, adattároló elemek valósítják meg.

Ezek az elemek azonban véges kapacitással rendelkeznek; így korlátja van, hogy mekkora terhelést képesek kiszolgálni úgy, hogy a szolgáltatás minősége ne romoljon érdemben. A folyamfeldolgozás példáján ez különösen szemléletesen látszik: ha a beérkező adatelemek hosszú távú érkezési rátája meghaladja egy feldolgozási lépés kiszolgálóinak feldolgozási rátáját, akkor vagy elkezdünk elemeket elveszteni, vagy azok elkezdenek a feldolgozásra várva „feltorlódni" (még rosszabb eset, amikor a túlterhelés hatására a rendszer összeomlik).

Az adatfolyam-feldolgozás jó lehetőséget teremt arra is, hogy demonstráljuk a szolgáltatásminőség szempontjából meghatározó teljesítményjellemzők alkalmazás-, de legalábbis alkalmazásiosztály-függőségét. Az 1. táblázat [BALLARD et al. 2010] nyomán bemutatja az adatfolyam-feldolgozás néhány jellemző használati esetét. Ezeket mi kiegészítjük okos város példákkal is.

Alkalmazásosztály	Feldolgozás eredményének jellemző értelmezése	Meghatározó teljesítmény-attribútum
„Áteresztőképesség" (throughput)	Elemzési eredmények folyama	Feldolgozási ráta
„Késleltetés" (latency)	Felderített üzleti lehetőségek folyama	Késleltetés
Előrejelzés (Prediction)	Figyelmeztetések	Pontosság
Vezérlés (Control)	Vezérlőjel, javasolt beavatkozások	Késleltetés és stabilitása; időszerűség (timeliness)

1. táblázat. Néhány jellemző adatfolyam-feldolgozási alkalmazásosztály és meghatározó teljesítményattribútumaik

Forrás: [BALLARD et al. 2010], 80. oldala alapján a szerzők szerkesztése

Az áteresztőképesség-fókuszú alkalmazások esetén nagyon nagy mennyiségű adatelem feldolgozása szükséges úgy, hogy a tárolás és a tárolt adatok feldolgozása valamely okból nem praktikus megoldás; például mert a folyamfeldolgozás szerepe a teljes alkalmazásban az, hogy az „érdekes", de nem azonnali beavatkozást igénylő megfigyelések leválogatásával csökkentse a kezelendő adatmennyiséget. Ilyen esetekben az elsődlegesen menedzselendő teljesítmény-attribútum a feldolgozási ráta (vagy a maximális feldolgozási kapacitás); a késleltetésre, ha vonatkoznak is követelmények, azok lazák. Ebbe a kategóriába sorolhatók sokszor pl. a weboldalak látogatásait (a „kattintások" sorozatát) reprezentáló adatfolyamok. Okosváros-környezetben különösen a tér- és forgalomfigyelő kamerákat alkalmazó megoldásokban lehet szerepe.

Ezzel szemben a **késleltetésfókuszú** alkalmazásoknál az elsődlegesen menedzselendő paraméter az egyes elemek feldolgozásának topológiaszintű összkésleltetése. Fő általános példái egyes pénzügyi alkalmazások és a riasztásokat generáló SCADA (Supervisory Control and Data Acquisition, felügyeleti vezérlés és adatgyűjtés) jellegű rendszerek; a túl lassú számítás ezen esetekben a kereskedési, illetve beavatkozási lehetőségek „elszalasztását" okozhatja. Az átmenő teljesítmény és a feldolgozási kapacitás másodlagosan vezérlendő paraméterek abban az értelemben, hogy kapacitástúlterhelés hatására a tervezett, illetve garantált késleltetés be nem tarthatóvá válhat. Okosváros-környezetben példája lehet a kritikus forgalmi helyzetek, balesetek, terrortámadások felismerése a polgárok mobileszközei által szállított geolokációs adatok alapján.

Az **előrejelzési** célú alkalmazáskategóriában a jelenleg és a múltban végzett megfigyelések alapján próbálunk egy (legtöbbször statisztikai) modell segítségével a jövő eseményeire előrejelzésekkel élni (kereskedési árak, veszélyhelyzetek stb.). Érdemes megfigyelni, hogy az ilyen alkalmazások esetén a szolgáltatásminőség elsődleges

attribútuma nem műszaki, illetve platform jellegű, hanem az előrejelzés pontossága: ha úgy tetszik, egy alkalmazási szintű jellemző. Példája lehet a korábban vázolt parkolási-időtartam-előrejelzés.

Ez nem jelenti azt, hogy a műszaki jellemzőknek (feldolgozási kapacitás, válaszidő) ne lennének adott esetben meghatározott minimumkövetelményei; azt viszont igen, hogy a minőségmenedzsment céljait nem közvetlenül ezek kontextusában fogjuk megfogalmazni. Ettől függetlenül a szolgáltatásminőség-menedzsment indirekt jelentheti ezek befolyásolását. Például különösen az optimalizációs eljárások területén ma már rendelkezünk olyan eljárásokkal, amelyek az idő előrehaladtával az optimum egyre jobb közelítését adják, és ha a rendelkezésre álló számítási erőforrások mennyiségét növeljük, akkor egyszerűen „gyorsabban” tartanak az optimumhoz.

A **vezérlési célú** alkalmazások esetén a folyamfeldolgozás eredményét közvetlenül alkalmazzuk egy számítógépes vagy fizikai rendszeren végzett beavatkozások meghatározására. A szabályozástechnikából ismert, hogy a késő, illetve változékony késleltetésű („jitter-es”) szabályozójelek komoly problémákat okozhatnak a rendszer „félrevezérlésével”. A kiberfizikai rendszerek terminológiájában ennek megfelelően – többek között a beágyazott rendszerek diszciplínája nyomán, amely az egyik közvetlen előképnek tekinthető – a rendszer időszerű viselkedésének (timeliness) követelményéről beszélünk. Ezekben az esetekben a számítások időkövetelményeknek megfelelő végrehajtása már nem is tisztán teljesítményattribútum, hanem legalább részlegesen a funkcionális megfelelőség része.

5. A teljesítménymenedzsment és tervezése

Az előző, motivációs fejezetünk érzékeltette, hogy az okosváros-megoldások teljesítménymegfelelősége kulcskövetelmény. Így mind a teljesítmény, mind pedig a különböző terheléseket elfogadható teljesítménnyel kiszolgáló kapacitások tervezése és üzemeltetési idejű menedzselése cél kell hogy legyenek. Jelen fejezetben először bemutatjuk ennek egy klasszikus megközelítését, majd annak okosváros-adaptációs lehetőségeit vizsgáljuk meg.

5.1. Az IT Infrastructure Library

A teljesítmény a szolgáltatásmegfelelőség egyik alapvető tulajdonsága; mint ilyen, menedzselésének szükségességét a ma már klasszikusnak tekinthető informatikai rendszertípusok – mint a vállalati és döntéstámogató rendszerek vagy a telekommunikációs szolgáltatások – kontextusában is felismerték már.

A vonatkozó iparágakban az évtizedek során összegyűlt rendszerüzemeltetési és tágabb értelemben szolgáltatásmenedzsment „jó gyakorlatok” (best practices) összefoglalásaként rendelkezésünkre áll több rendszerezett, minta- és ajánlásgyűjteményként közvetlenül felhasználható keretrendszer. Ezek közül a legismertebbek és legszélesebb körben alkalmazottak az Information Technology Infrastructure Library (ITIL; a teljes anyag több kötetre tagozódik, „magja” az Office of Government Commerce 2007a, mely az ISO/IEC 20000 [ISO–IEC 2011] szabvány alapját is adja és a COBIT [ISACA s. a.], eredetileg a „Control Objectives for Information and Related Technologies” rövidítéseként).

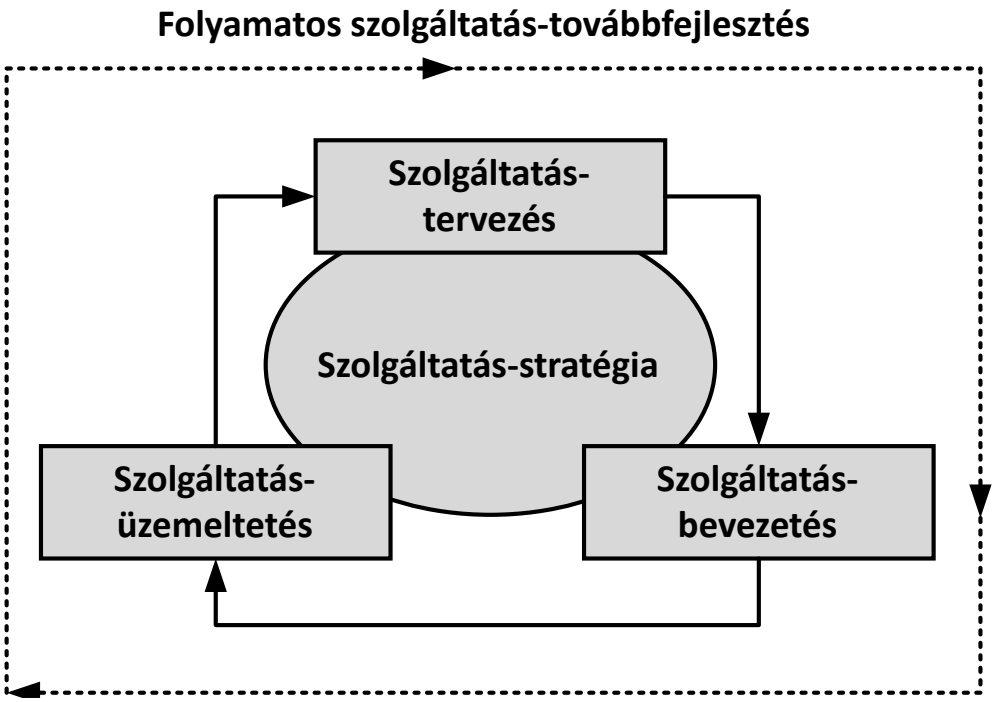
Az ITIL fő célja olyan közvetlenül testre szabható folyamatminták és gyakorlatok megadása, amelyek tapasztalati úton bizonyított módon segítenek az IT-szolgáltatások és az üzleti igények szinkronban tartásában – úgy funkcionális, mint extrafunkcionális szempontból. Ezzel szemben a COBIT felfogható egy még általánosabb, „vezérlési cél” szemléletű keretrendszerként, amely képes célvezérelten, absztrakt módon harmonizálni a különböző területeket célzó ajánlásgyűjteményeket (pl. folyamat szempontból: ITIL; a képességek javításának szemszögéből: CMMI [CMMI 2010], alkalmazott architektúrák tekintetében: TOGAF [HAREN 2011], projektmenedzsment oldalról: PMBOK [PMI s. a.]).

5.2. Teljesítménytervezés- és menedzsment az ITIL-ben

A teljesítménymenedzsmentet mint tevékenységet és folyamatot ezek közül az ITIL taglalja a talán a legteljesebb módon. Ebből kifolyólag itt alapvetően az ITIL v3-ra támaszkodva mutatjuk meg, hogy a teljesítménymenedzsment egy összetett rendszerben nem egyszerűen egy megoldandó műszaki probléma, de még csak nem is egyetlen (műszaki) folyamat, hanem egy több absztrakciós szinten, aspektusból és időfelbontással végzett folytonos tevékenység, amelynek alapját ideális esetben a rendszer, a rendszermenedzsment és az általa megvalósított szolgáltatások holisztikus nézete adja.

Bár az ITIL mint keretrendszer alapját az autonóm számítástechnika, valamint a kiberfizikai és felhőrendszerek megjelenése előtti évtizedek tudása és tapasztalata adja,

meglátásai és ajánlásai időtállók és adaptálhatók a modern rendszerkategóriákra is. Megjegyzendő, hogy az ITIL teljesítménymenedzsment-ajánlásait csak erősen kivonatolva áll módunkban közölni, egyben interpretálva és kommentálva ezeket.



6. ábra. Az ITIL szolgáltatásmenedzsment megközelítése
Forrás: a szerzők szerkesztése

Az ITIL az informatikai szolgáltatások teljes életciklusára megfogalmaz ajánlásokat, az életciklust a tervezés-bevezetés-üzemeltetés fő állomásokba rendezett folyamatokkal modellezve. Az ITIL által ajánlásokkal lefedett területek összességében a következők. (A területek kapcsolatait a 6. ábra szemlélteti.)

- Szolgáltatásstratégia (Service strategy): a szolgáltatásstratégia vezérli a folyamatokat; kiemelten fontos részei a szolgáltatáskatalógus, a belső és külső „piacok” fejlesztése és a stratégiai kockázatok azonosítása.
- Szolgáltatástervezés (Service design): szolgáltatások és szolgáltatásmenedzsment-folyamatok tervezése és fejlesztése.
- Szolgáltatásbevezetés (Service transition): új és módosított szolgáltatások bevezetésének képessége.
- Szolgáltatásüzemeltetés (Service operation): azon folyamatok és képességek, amelyek a megvalósított szolgáltatások hatékony és a követelményeknek megfelelő nyújtását teszik lehetővé.
- Folyamatos szolgáltatás-továbbfejlesztés (Continual service improvement): azon gyakorlatok, amelyek lehetővé teszik a szolgáltatások minőségének változó körülmények között is fenntartását és a szolgáltatásokkal létrehozott érték folyamatos növelését.

A szolgáltatásteljesítmény biztosításának két tevékenység adja az alapját ebben a modellben. Egyrészről a szükséges kapacitásokat tervezési időben (szolgáltatástervezés)

azonosítani kell, a rövid, közép- és hosszú távú felhasználói igényeknek megfelelően. Érdemes hangsúlyoznunk, hogy a szolgáltatás alapját adó IKT-megoldásokat és azok alkalmazásba integrációjának logikáját ez a tevékenység már adottnak veszi; a tervezési tevékenység teljesítmény szempontból itt az adott megoldás erőforrásokkal („kapacitásokkal") ellátására vonatkozik.

Másrészről a szolgáltatásteljesítményt tevékenyen szükséges menedzselni is. Az ITIL definíciója szerint a teljesítménymenedzsment a folyamatos szolgáltatás-továbbfejlesztés (Continual Service Improvement, CSI) része; ezen belül az a szolgáltatásmenedzsment-folyamat, mely a „napi szintű" kapacitásmenedzsment-tevékenységekért felelős. Részét képezi a monitorozás, határértékátlépés-detektálás, teljesítményelemzés- és hangolás, valamint a teljesítménnyel és kapacitással kapcsolatos módosítások megvalósítása ([Office of Government Commerce 2007b], 204. oldal alapján).

A következőkben bemutatunk néhány alapfogalmat, majd ismertetjük a teljesítménymenedzsment főbb folyamatait is. A tervezési idejű tevékenységeket az általános esetben külön nem tárgyaljuk; jelen anyag céljaira az egyszerűsíthető úgy, mint a következőkben bemutatott igényvezérelt szolgáltatásmenedzsment-megközelítés által megfogalmazott igények IKT-kapacitásokkal kielégítése.

5.3. Szolgáltatási szintek: kapacitás és teljesítmény

Egy menedzselt, IT-rendszerek segítségével megvalósított szolgáltatással szemben végfelhasználói igényekkel élnek; egy jól menedzselt rendszerben ezeket úgynevezett szolgáltatási szint követelmények (Service Level Requirement, SLR) írják le, melyek alapján a szolgáltatást nyújtó fél vállalásokat is tehet szolgáltatásiszint-szerződések (Service Level Agreement, SLA) formájában.

A követelmények strukturáltak; a funkcionális igényeken túl jellemzően meghatározzák a rendelkezésre állási és teljesítménykövetelményeket, ahol a teljesítményre vonatkozóan jellemzően megegyezés történik

1. a szolgáltató oldaláról a (szolgáltatási) kapacitásról,
2. a felhasználó oldaláról a megengedett/feltételezett munkaterhelésről és használati mintákról,
3. egyéb teljesítményjellemzőkről – kiemelten a válaszidőről, amennyiben az értelmezhető, és
4. a teljesítményt érintő incidensek esetleges elhárításának módjáról és minőségi jellemzőiről.

Ezeket az aspektusokat és jellemző szerepeltetésük okait érdemes közelebbről is megvizsgálnunk.

A kapacitás kulcsfogalom bármely rendszerben, amely véges erőforrások segítségével nyújt használatában skálázható szolgáltatást egy szolgáltatást nyújtó fél–szolgáltatást használó fél/felek viszonylatban. A szolgáltatáshasználat mértékének növekedésével a rendelkezésre álló erőforrások kihasználtsága is nő, mely folyamat a „végtelenségig" értelemszerűen nem skálázódhat. Igaz ez gyakorlatilag minden szolgáltatási kontextusban;

- alacsony szintű informatikai erőforrásokra: processzor, memória, hálózati sávszélesség stb.,

- logikai erőforrásokra: komponensek meghatározott végrehajtóelem-halmazokkal, szolgáltatás részeként igénybe vett további szolgáltatások stb.,
- és az emberi erőforrások tekintetében is.

A szolgáltatást nyújtó fél elsődlegesen a megegyezett kapacitásra tervezi a szolgáltatást nyújtó rendszereket és folyamatokat.

A kulcsfontosságú üzenet itt az, hogy a megegyezett kapacitás alatti használat esetén mutatott és a kapacitás szempontjából nézve túlterhelés esetén jelentkező szolgáltatásjellemzők a szolgáltatásnak két élesen elkülönülő aspektusát adják. Ezek külön specifikálandók és kezelendők; összhangjuk megteremtése egy magasabb szinten, a szolgáltatási igények és szolgáltatásnyújtási képességek összehangolása során történik meg. A vállalt kapacitás meghatározása mellett, szinte azonos megfontolások mentén a „vállalt" kapacitáskihasználás mintázatait, azaz a munkaterhelés jellemzőit is helyes szabályozni előre. Egy példa annak egyértelmű lefektetése, hogy egy átmeneti, „tüskeszerű" (burst) terhelést a szolgáltatásnak „bírnia kell-e" a szolgáltatásminőségi jellemzők megtartása mellett (és ha igen, mik jellemzik a „még vállalt" tüskéket).

Okosváros-környezetben például a város dönthet úgy, hogy az innovációt elősegítendő a városi közlekedési vállalatot arra ösztönzi, hogy a tömegközlekedési eszközök mozgásának valós idejű adatait szolgáltatásként „nyissa ki" más okosváros-megoldások számára (például autómegosztással összekötött multimodális közösségi utazástervező). Ezt a közlekedési vállalat megteheti mindenféle garancia nélkül is, bár ez a korábban tárgyalt kockázatokat rejti magában. Ezzel szemben az ITIL jó gyakorlatát képezi le a következő egyszerű stratégia.

a. A vállalat díjazásért, vagy pusztán állampolgári/cégazonosításhoz kötve SLA-ban vállalja, hogy az ügyfél 5 percenként egyszer kevesebb mint 5 másodperc késleltetéssel lekérdezheti legfeljebb 20 jármű helyzetét egy API-n keresztül. A túl gyakori lekérdezések az ügyfél kizárását vonják maguk után.
b. A magukat nem azonosító ügyfelek kiszolgálását a vállalat explicit kizárja.
c. A vállalat ügyfeleit prioritási kategóriákba rendezi és ezt minden féllel tudatja. Túlterhelés – túl sok aktív ügyfél, kibertámadás vagy infrastruktúra-hibák – esetén a magas prioritású ügyfelek (rendőrség, tűzoltóság, mentők) kiszolgálása elsőbbséget élvez.

Az utolsó pont esetén nyilvánvaló, hogy számos egyéb stratégia is elképzelhető, mint ahogy az első pont esetében is végezhetjük máshogy a „kapacitásvédelmet". A 6. és 7. fejezet célja, hogy az ellenállóképesség stratégiai tervezését és a szokásosan használt védelmi mechanizmusokat bemutassa.

5.4. Skálázható kapacitás

A felhő- és felhő jellegű szolgáltatások és platformok megjelenésével azt gondolhatnánk, hogy a kapacitásmegállapodások – legalábbis a felhőalapú rendszerek kontextusában – meghaladottá válnak, hiszen a felhőben – definíció szerint – elasztikusan skálázható módon áll rendelkezésünkre a kapacitás. Így fix kapacitások helyett alapvetően árazási és elszámolási kérdésekben szükséges megállapodni.

Az erőforrások és szolgáltatások elasztikussága azonban önmagában nem elég – alkalmazási- és folyamatszintű skálázhatóság is szükséges kihasználásukhoz; a kifejezetten felhő környezetre tervezett alkalmazások kivételével pedig a létező alkalmazások

„felhősítése” (cloudification) jellemzően komoly műszaki kihívás. Emellett az új kapacitások beállítása jellemzően nem elhanyagolható késleltetéssel rendelkezik.

Mindezeknek megfelelően helyesebb úgy fogalmaznunk, hogy skálázható kapacitású szolgáltatásnál az SLR-alapú megállapodások sémája ugyan módosításra szorulhat, de minimálisan is az javasolható, hogy a kapacitásskálázás jellemzői képezzék az SLR részét. Ilyen karakterisztikák például a skálázási kvantumok, a skálázás „sebességi” jellemzői, vagy az alkalmazandó skálázási logika egyéb, például terhelésmetrikák függvényében. (A korábban is tárgyaltakon túl ezeknek egy bővebb áttekintését adja pl. [KOCSIS–SALÁNKI–PATARICZA 2016])

Előző példánkkal élve: a szolgáltatásnak képezheti részét az, hogy az ügyfél a megengedett lekérdezési gyakoriságot és/vagy a késleltetést skálázza, a gyakrabban végzett, illetve kisebb késleltetésű lekérdezésért többet fizetve. Ekkor a közlekedési vállalat valójában már egy SaaS-szolgáltatást nyújt. A felskálázott használat kiszolgálásához azonban neki is többleterőforrásra van szüksége. Ezt vagy saját rendszere kapacitástartalékaiból fedezi, vagy, ha ő maga is felhőt használ, többleterőforrásokat igényel. Ennek viszont késleltetése van, így a szolgáltatáskiterjesztésre mint műveletre indokolt lehet egy maximális késleltetésvállalás szerepeltetése az SLA-ban.

5.5. QoS-metrikák meghatározása

Tekintsük most csak azt az esetet, amikor a felhasználó a kapacitáskorláton belül használ egy szolgáltatást. Mint korábban már szemléltetettük, ekkor a szolgáltatás felhasználása szempontjából is fontos teljesítményjellemzők és ezek megengedett határértékei erősen szolgáltatás- és esetfüggők. Mint arra példát is mutattunk, általános ajánlások tehetők, mindemellett a meghatározó Quality of Service (QoS) metrikák és határértékek meghatározása részben szakterület-specifikus tudáson, részben pedig szolgáltatásdekompozíció mentén végzett elemzésen kell hogy alapuljon. Utóbbi egy praktikus, félformális megközelítése pl. a Goal-Question-Metric (GQM) módszer [BASILI–CALDIERA–ROMBACH 1994]. Következésképp egy általános célú gyakorlatgyűjtemény, mint az ITIL, a metrikák szintjén konkrét javaslatokat nem tud megfogalmazni. Egy további oka is van annak, hogy a korábban összegyűjtött IKT-területek jellemző QoS-metrikáit nem tekintjük át: ezek szisztematikus tárgyalásához szükséges lenne informatikai előképzettség feltételeznünk. (Az érdeklődő olvasó [ORIOL–MARCO–FRANCH 2014]-ben talál egy jó áttekintést a webszolgáltatásokra; a felhőszolgáltatások teljesítménymetrikáival kapcsolatban javasoljuk [KOCSIS et al. 2016]-ot.)

Az ITIL javasolja egy sor olyan támogató folyamat megvalósítását a szolgáltatásüzemeltetés részeként, amelyek általában biztosítani képesek az elvárásoknak megfelelő működést. Ezekre a folyamatokra nemcsak hogy mintákat javasol az ITIL, de jellemzően a folyamat minőségének leírására alkalmas metrikák gyűjteményei is elérhetők [BROOKS 2006]. Ilyen folyamat például a szolgáltatászavarok, ITIL-terminológiában: incidensek kezelésének folyamata (Incident Management). A teljesítményincidensek elhárításával kapcsolatos minőségi jellemzők (pl. szolgáltatás helyreállítása „lelassulás” észlelése és bejelentése) esetén javasolható, hogy az SLR részét képezzék. Hasonlóan a szolgáltatásminőségi metrikákhoz, a szolgáltatáson értelmezett teljesítményincidens fogalmának meghatározását az ITIL nem tekinti a hatókörébe eső feladatnak.

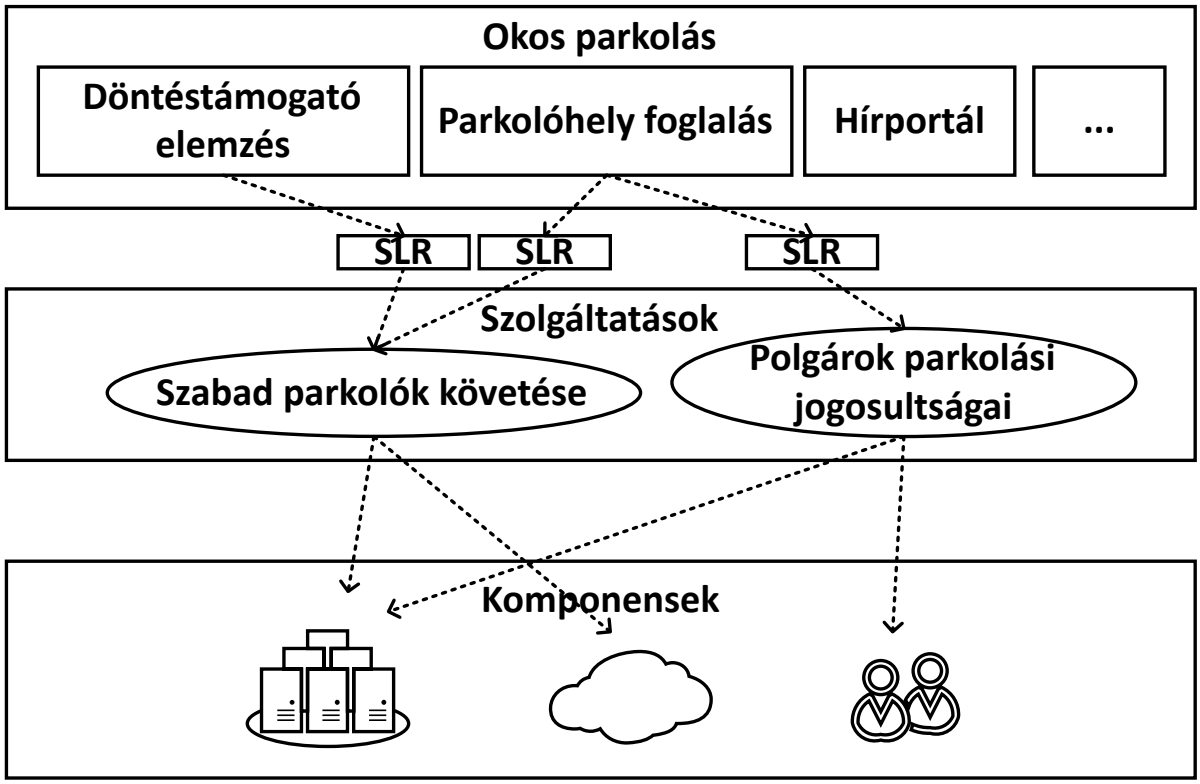
5.6. Kapacitás-, terhelés- és igénymenedzsment

Ezen megfontolások alapján érthető, hogy miért tekint az ITIL a teljesítménymenedzsmentre úgy, mint a folyamatos szolgáltatásjavítás kapacitásmenedzsment jellegű komponensére. A teljesítménymegfelelőség biztosításnak a javasolt módja az, hogy

- legyen meghatározva a rendelkezésre bocsátandó szolgáltatási kapacitás, szükség esetén a „túlhasználat” esetén mutatott viselkedést is szabályozva,
- legyenek mechanizmusok az incidensek kezelésére és
- a folyamatos felülvizsgálat és az általános értelemben javítás részét képezze a megvalósítási kapacitások folytonos felülvizsgálata.

Ez utóbbi alapvető célja a teljesítményhibák proaktív megelőzése. A felülvizsgálat optimális esetben a *megoldásszintű kapacitás → részszolgáltatás-kapacitások → komponens-kapacitások* dekompozíciós hierarchia mentén, felülről lefelé haladva történik. (Figyeljük meg, hogy okos város esetében még egy legfelső hierarchiaszintre is szükség lehet az egymásra épülő okosváros megoldások miatt.)

Elhanyagolásokkal élve, de az ITIL filozófiájához hűnek maradva mondhatjuk azt, hogy szolgáltatások céljai és az azokkal kapcsolatos stratégiák segítenek előre jelezni a szükséges szolgáltatáskapacitásokat; a szolgáltatáskapacitás-igények növekedési modelljei alapján pedig jóval nagyobb biztonsággal értékelhetők ki a komponenskapacitások tartalékai és tervezhetők szükséges bővítései, mint pusztán a műszaki kihasználtságjellemzők és műszaki szolgáltatásminőségi metrikák alapján. A hierarchiát a 7. ábra szemlélteti, egy (részleges) okosváros-példával.



7. ábra. A hierarchikus CSI kapacitás-menedzsment szintjei
és függőségei egy részleges okosváros-példán

Forrás: [Office of Government Commerce 2007b], 5.6 alfejezet ábrái alapján a szerzők szerkesztése

Ez a holisztikus nézet teremti meg a munkaterhelés-menendzsment (Workload Management) és az igénymenedzsment (Demand Management) tervezhetőségét is. Erősen változó mértékben, de általában mind egy rendszer összterhelésének ütemezése, mind pedig a rendszer egyes klienseinek a viselkedése (terhelés- és erőforrásigény generálása) alakíthatók.

A terhelési tüskék mérete sok esetben radikálisan csökkenthető „több tüskére bontással"; pl. egy felsőoktatási, laborgyakorlatokat kiszolgáló felhő maximális terhelése könnyen minimalizálható a különböző tárgyak beadási határidőinek és a bemutatási időpontjainak optimalizált, egymáshoz képest vett eltolásával [SALÁNKI et al. s. a.]. A felhasználói igények befolyásolását általában óvatosan kell kezelni, mivel költsége az elégedettségromlásban könnyen túl magas lehet; a felsőoktatási felhő esetén lehet példa erre, hogy a magas terhelésű időszakokban előre kommunikált módon csak korlátozott erőforrásokkal felszerelt virtuális gépet kapnak azok, akiknek a feladatleadása az adott héten még nem aktuális, ezzel sarkallva a hallgatókat arra, hogy a feladat komoly számítási igényű aspektusát (pl. teljes adatkészleten modelltanítás) a kívánt időszakban végezzék.

A CSI részeként végzett folyamatos kapacitás-felülvizsgálat mint iteratív kapacitásmenedzsment tevékenységre megközelítéseket javasolt az ITIL. Ily módon a jó gyakorlatok részét képezi a trendelemzés és a modellezés; utóbbi esetén az ITIL kimondja, hogy – az empirikus, „black box" módszereken túl – mind az analitikus, mind a szimulációs modellezés jelenthetnek opciót. Mindemellett azonban specifikus modellezési megközelítéseket az ITIL nem javasol. A modellezés stílusát, matematikai és műszaki értelemben vett kifinomultságát számos esetfüggő faktor is széles skálán befolyásolja, így egy „menedzsment jó gyakorlat" gyűjteménynek nem is lehet célja ezek összegyűjtése.

5.7. ITIL szemléletű okos város teljesítménymenedzsment

Mint azt [LECHNER 2017] is megjegyzi, az okos városok szolgáltatáséletciklus-modelljei még csak részlegesen alakultak ki; irodalomkutatásunk arra enged következtetni, hogy jelenleg még nem léteznek a teljesítménymenedzsment okosváros-szolgáltatásmenedzsmentbe integrált megközelítései.

Állításunk, hogy az ITIL által megalapozott jó gyakorlatok alkalmazhatók okos város kontextusban is; különösen a hazánkban alkalmazandó (a Brit Szabványügyi Hivatal keretrendszerét adaptáló) okos városfejlesztési-módszertan követése esetén. A korábban bemutatott módszertan és az ITIL között a következő természetes kompatibilitások figyelhetők meg.

- **Szolgáltatásorientáltság**: az életciklus különböző fázisaiban végzett tevékenységek tárgya mindkét esetben a szolgáltatás, attól függetlenül, hogy az ITIL jellemzően klasszikus IKT-módszerekkel, az okos város pedig CPS szemléletben megvalósított szolgáltatásokat céloz meg.
- **Szolgáltatáskatalógus és szolgáltatásfüggőségek**: mindkét keretrendszer felismeri, hogy a szolgáltatások összessége egy szolgáltatáskatalógusba rendezendő, és a szolgáltatások minden életcikluslépésénél figyelembe kell venni a szolgáltatások közötti függőségeket is.
- **Szerepkörök szétválasztása**: mindkét keretrendszer egyértelműen azonosít szerepköröket.

Ezen tulajdonságoknak köszönhetően az okosváros-szolgáltatások teljesítménytervezése és teljesítménymenedzsmentje megvalósítható SLA/SLR szemléletben, ITIL-alapú folyamatokkal.

Az ITIL és az okosváros-módszertan azonban egy lényeges ponton különbözik is. Az ITIL szervezetfókuszú; alapvetően egy adott szervezet által belső és külső ügyfelek számára nyújtott szolgáltatások menedzsmentjéhez ad ajánlásokat. Ezzel szemben az okos városban több, szervezetileg és szerepkörben is különböző szereplő nyújtja az egymást is támogató szolgáltatásokat (lásd az okos város szolgáltatási mátrix fogalmát).

Az okos város több szinten is hierarchikus és függőségi gráfba szerveződő szolgáltatáshálója várhatóan komplexebb és CPS-jellege miatt „sérülékenyebb" is, mint a vállalati rendszerek esetében. Az egyes okosváros-megoldások különböző IKT-megoldások integrációjára épülnek (lásd ezek rövid ismertetését korábban); egy absztrakciós szinttel magasabban az okosváros-megoldások pedig más megoldásokat használhatnak szolgáltatásként. Mindeközben az okosváros-megoldások által használt IKT-kapacitások között átfedés is lehet; pl. közös kommunikációs infrastruktúra, közösítve használt (akár városi) felhőszolgáltatások stb.

Hogyan kerülhetjük el mégis, hogy helyi teljesítményproblémák rendszerszintű hatással legyenek? Kutatásaink jelen pontján a következő ajánlásokkal tudunk élni az ITIL adaptálására (ezek azonban az ITIL-lel szemben még nem bizonyított „jó gyakorlatok").

1. Az önkormányzat koordinációs szerepének a többi szereplőt összefogó módon ki kell terjednie a teljesítményre is; egyfajta „ITIL-metaszervezetként" koordinálva a szereplők között a kritikus teljesítmény- és kapacitásmegállapodásokat.
    a. A stratégiai tervezés részeként létrejövő szolgáltatáshálózatban kockázat- és hibaterjedés-elemzéssel azonosítania kell azokat a kritikus szolgáltatáskapcsolatokat, ahol a szolgáltatást nyújtó fél számára formális SLR-ek írandók elő. (Lásd korábban az okos város szerepköreit.)
    b. A stratégiai tervezés részévé kell tennie az SLR-ek megfogalmazását, majd a megvalósítás során a rendelkezésére álló eszközökkel hozzá kell járulnia ahhoz, hogy ezeket teljesítő okosváros-szolgáltatások jöjjenek létre.
    c. A stratégiai tervezés részévé kell tennie az előzetes rendszerszintű kapacitástervezést – ha úgy tetszik, az okos város IKT-kapacitásmenedzsment-stratégiájának kialakítását. A későbbiekben bemutatunk egy sor automatikus kapacitásmenedzsment-megoldást; ezek azonban csak akkor alkalmazhatók okos városban valóban hatékonyan, ha a város mint szabályozó és koordináló szereplő lehetővé teszi és koordinálja a különböző szereplők kapacitásai közötti átjárást.
    d. Az üzemeltetés során követnie kell, hogy az okos város működése szempontjából kritikus teljesítményfüggőségekhez kapcsolódó megállapodásokat a felek betartják-e. A megállapodások megsértése esetén lehetősége van az SLA-ban lefektetett büntetések érvényesítésében segédkezni.
2. A rendszerszintű kapacitástervezési stratégiához kapcsolódóan az önkormányzatnak:
    a. lehetősége van javasolni vagy előírni, hogy az egyes okosváros-megoldások milyen teljesítménymenedzsment-stratégiát kövessenek, amikor belső vagy külső hibákból (ideértve az előre nem látott túlterhelést is) nem képesek vállalt SLA-ikat teljesíteni;
    b. lehetősége van olyan, városi szintű teljesítmény- és kapacitásmenedzsment-stratégiák megfogalmazására, amelyek kihasználják az okos város szolgáltatási katalógus diverzitását és a teljes támogató infrastruktúra méretét.

6. Autonóm teljesítménymenedzsment

A 2000-es évekre a nagyvállalati számítástechnika egyfajta üzemeltetési válságba került; a komplex számítógépfürtöket igénylő, összetett rendszerekkel megvalósított szolgáltatások a különböző extra-funkcionális követelményeknek megfelelő üzemeltetése egyre komolyabb emberierőforrás-ráfordítást igényelt. A jelenségre reakcióként az IBM 2001-ben elindította az „autonóm számítástechnika" (autonomic computing) kezdeményezését. Ezt több más óriáscég is követte a saját programjával; ma már kijelenthetjük, hogy az informatika fejlődése szempontjából történelmi jelentősége az IBM által definiált autonóm számítástechnikának volt, amely önálló tudományterületté fejlődött.

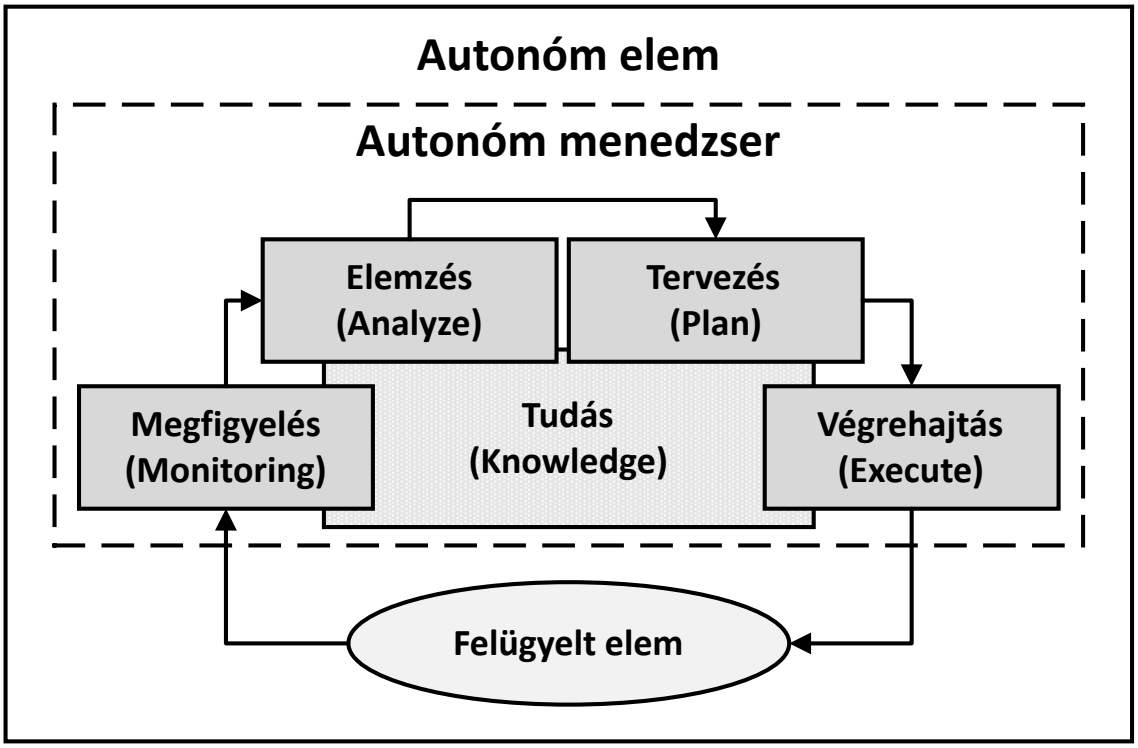
6.1. Autonóm számítástechnika

Informálisan az autonóm számítástechnika olyan számítástechnikai rendszereket jelöl, amelyek nagymértékben képesek magas szinten megfogalmazott eljárásrendek alapján „önmagukat menedzselni" – éles ellentétben az ezredfordulóra jellemző folyamatos és szoros szakértői felügyelettel és „állítgatással". Az autonóm számítástechnika pontosabb definíciói leíróak és tulajdonságdekompozíción alapulnak. A négy fő „ön-*" (self-*) tulajdonság, amelyek együtt az autonóm viselkedés meghatározói, az önkonfiguráció (self-configuration), az önoptimalizáció (self-optimization), az öngyógyítás (self-healing) és az önvédelem (self-protection) (lásd a 2. táblázatot).

Fogalom	Autonóm számítástechnika előtt	Autonóm számítástechnika
Önkonfiguráció	A komplex, több beszállítótól származó szoftverek szakértői telepítése, konfigurálása és integrálása időigényes; gyakori hibák	A komponens- és rendszerszintű konfiguráció automatikus és magas szintű eljárásrendeket (policy) követ. A rendszerszintű kontextus automatikusan adaptálja magát a változáshoz.
Önoptimalizáció	A rendszerkomponensek több száz nemlineáris hangolási paraméterét szakértők kezelik	A komponensek és a rendszerek folyamatosan saját teljesítményük és hatékonyságuk javítására törekednek.
Öngyógyítás	A hibadiagnosztika és –javítás sokszor heteket vesz igénybe	A rendszer a lokalizált szoftver- és hardverhibákat automatikusan detektálja, diagnosztizálja és javítja.
Önvédelem	Mind a támadások, mind a hibaterjedési láncok felismerése és a védekezés/helyreállítás nagyban manuális, helyi megoldásokat alkalmaz és erősen esetleges	A rendszer automatikusan védekezik a rosszindulatú támadásokkal szemben és hatékonyan korlátozza a belső hibaterjedést. A rendszerszintű hatásokat előrejelző detektálási mechanizmusokra támaszkodva, proaktív módon elkerüli.

2. táblázat. Az autonóm számítástechnikai rendszerek négy fő önmenedzselési képessége Forrás: [KEPHART–CHESS 2003], 3. oldal alapján a szerzők szerkesztése

Ezen tulajdonságok értelmezhetők egyfajta követelményrendszerként. Az autonóm számítástechnika fogalmához hozzátartoznak azok az alapvető architekturális és viselkedési minták is, amelyekkel a követelmények megvalósítása „jó gyakorlatként” javasolt (lásd pl. [MURCH 2004]); ezek közül a legfontosabb és leginkább időtálló az „autonóm elem” (autonomic element) fogalma.



8. ábra. Az autonóm menedzser és a MAPE-K szabályozási kör

Forrás: a szerzők szerkesztése

Az architekturális minta alapgondolata, hogy konzisztens módon eljuthatunk rendszerszinten autonóm viselkedésig, ha a rendszer nem autonóm elemeit dedikált „autonóm menedzser” elemek felügyelete alá helyezzük és a rendszert ezen autonóm elemek – akár dinamikus – hálózataként szervezzük. Az autonóm menedzser folyamatos tevékenysége tudás által vezérelt megfigyelés, (helyzet)elemzés, tervezés és végrehajtás fázisokra bontható; a fázisok kanonikus angol nevei alapján MAPE-K szabályozási körként (MAPE-K control loop) is hivatkozunk az autonóm menedzserre (8. ábra). Mint később azt részletezzük, a szabályozástechnika terminológiájának kölcsönvétele közel sem véletlen.

6.2. Eljárásrend-alapú felügyeleti logika és futásidejű tervezés

Az autonóm számítástechnika jelentős eredményei közé tartozik az eljárásrend-alapú (policy-based, illetve policy-driven) rendszerfelügyelet kodifikálása és szisztematikus tervezési módszereinek megalapozása is. A magas szintű célok alapján az automatikus felügyeleti beavatkozástervezés mint igény kielégítése természetesen azt hozza magával, hogy a felügyeleti logika a korábbinál jóval mélyebb és tudásintenzívebb módszereket kell hogy használjon.

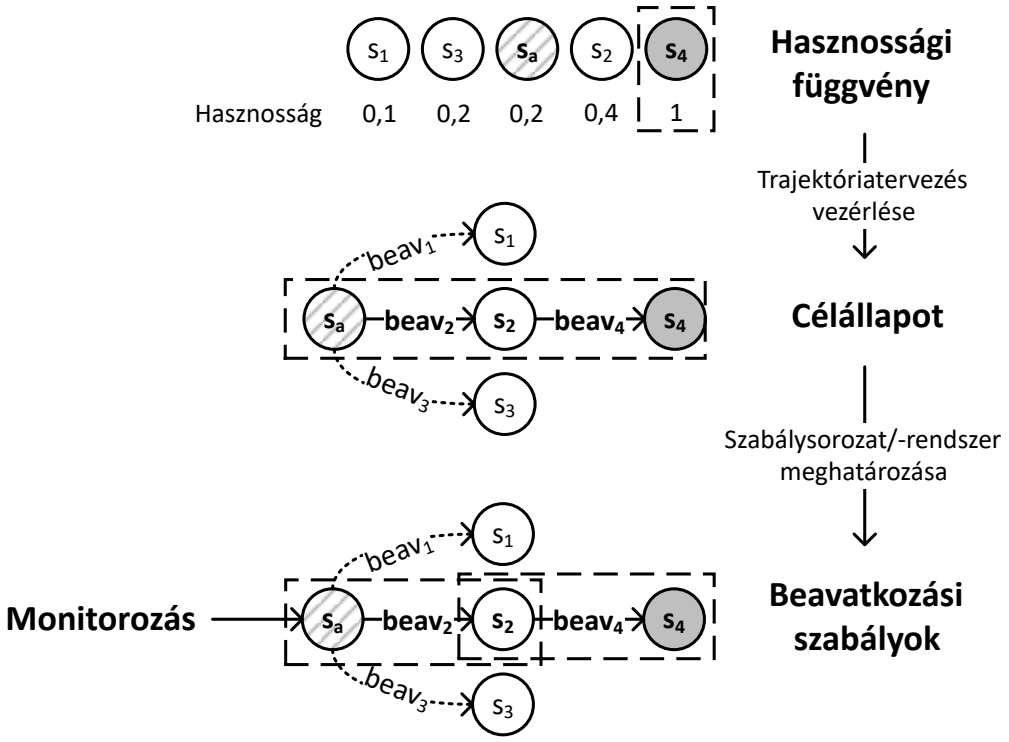
Ezek esetlegesen elfedhetők a rendszerintegrátor és a rendszerfelügyelet elől, amennyiben a szoftver- vagy rendszergyártó a felügyelt elemet az autonóm menedzserrel együtt, „fekete dobozként" tudja szállítani – ilyenek például az autonóm jellemzőket felmutató levelezőrendszerek [PAREKH et al. 2001] és adatbázisok [LIGHTSTONE–LOHMAN–ZILIO 2002], amelyek eljárásrend-alapú beléptetésmenedzsmentet valósítanak meg, a lekérdezések futásidejét adaptívan kezelik, vagy köztestárakat hangolnak automatikusan.

Azon esetekben azonban, amikor ez nem teljesül, az intelligens módszerek alkalmazásának szükségessége jellemzően kemény korlátokat állít az autonóm funkcionalitás megvalósítása elé. A testre szabott autonóm funkciókat megvalósító logika kifejlesztése ugyanis csak nagyméretű rendszerekre, elégségesen homogén rendszeraspektusokra lesz gazdaságos. Ennek kiemelt jelentőségű példája a QoS- és hibatűrés-vezérelt erőforrás-skálázás felhő (jellemzően IaaS) alkalmazásokban [LORIDO-BOTRAN–MIGUEL-ALONSO–LOZANO 2014].

Az autonóm rendszerekbe beépítésre kerülő intelligencia három meghatározó kategóriája a szabályozástechnikai értelemben vett szabályozási körök; a mesterséges intelligencia és az operációkutatás eszközeit alkalmazó, futásidőben is megvalósítható konfiguráció- és trajektória-tervezési módszerek; és az automatikus statisztikai modellépítés. Minél magasabb szintű eljárásrendek támogatását tűzzük ki célul – azaz a rendszermenedzsmentnek „minél kevesebb dologgal kell foglalkoznia" –, annál komolyabb apparátust igényel az eljárásrendben megfogalmazott célok infrastruktúra- és szoftverkomponens szintű felügyeleti és vezérlési akciókra leképezése. Az eljárásrendeket szokásosan a következő módon kategorizáljuk ([KEPHART–WALSH 2004] alapján):

- Beavatkozás-eljárásrendek (action policies): az autonóm elem a monitorozási képessége segítségével egy állapotképet (state) tart karban a rendszer állapotáról. A beavatkozás-eljárásrendek ezen állapotkép felett megfogalmazott HA(predikátum az állapot felett) → AKKOR(beavatkozás) szerkezetű szabályokból komponált szabályrendszerek. Azt, hogy a feltétel teljesülése esetén a beavatkozás-végrehajtásra a rendszer a kívánt állapotba kerüljön, a szabályrendszert megfogalmazó szakértőnek kell biztosítania.
- Célállapot-eljárásrendek (goal policies): az eljárásrend pusztán azt adja meg, hogy mi a rendszertől aktuálisan megkívánt állapot (vagy mik az elfogadható állapotok jellemzői); az eljárásrendet értelmező autonóm elem feladata, hogy a jelenlegi állapotból a trajektóriatervezést (állapotsorozat az aktuális állapot és a megkívánt között, valamint az állapotátmeneteket biztosító felügyeleti/vezérlési akciók sorozata) megvalósítsa. Mind a klasszikus szabályozások, mind pedig a rendszerszintű tulajdonságok alapján végzett, kombinatorikus keresést és folytonos optimalizálást alkalmazó strukturális átkonfigurálások ebbe a kategóriába esnek (kritikus alkalmazásokat futtató clou-környezetekre lásd példaként [KOCSIS–MANN–ZILAHI s. a.]).
- Hasznosságifüggvény-alapú eljárásrendek (utility function policies): az üzemeltetés feladata egy hasznossági függvény megadása a rendszerállapotok felett; ez elméletben lehetővé teszi, hogy a megfelelő intelligenciával ellátott autonóm menedzser az aktuálisan elérhető állapotok teréből tetszőleges pillanatban kiválassza a „legjobbat", és amennyiben a rendszer ebben az értelemben szuboptimális állapotban van, azt adaptálja.

A három absztrakciós szint között nemcsak logikai hierarchia áll fent, de megvalósításuk is történhet rétegzett módon. A célállapot-eljárásrendek tervezési aspektusának kimenete értelmezhető egy „tervezett” beavatkozás-eljárásrendként; a hasznossági-függvény-alapú eljárásrendek pedig generálhatnak célállapot-eljárásrendeket. Ennek megfelelően a tervezés sokszor történhet legalább részlegesen offline módon is. Az eljárásrendtípusok hierarchiáját a 9. ábra szemlélteti.



9. ábra. Az autonóm számítástechnika eljárásrendjeinek absztrakciós hierarchiája és egymásra épülése
Forrás: a szerzők szerkesztése

6.3. Autonóm teljesítménymenedzsment

Az autonóm működés általános célrendszere és működési sémája nagyban hasonlít az ITIL-hez és más „jó gyakorlat” gyűjteményekhez abból a szempontból, hogy egy menedzsment – műszaki szemszögből: vezérlési – sémát ad meg, de a vezérelt rendszerjellemzőt, rendszerjellemzőket nem, vagy csak nagyon lazán köti meg. Így az autonóm működés célja több különböző rendszerjellemző hatékony, erősen önálló szabályozása is lehet; ezek között kiemelt fontosságú a teljesítmény.

A korábbi fejezetek bemutatták a teljesítménymenedzsment klasszikus – és klasszikusan jórészt emberi erőforrásokkal megvalósított – folyamataspektusait; és ennek rövid időhorizontú komplementerét, a teljesítmény menedzselésére a túlterheléshibák kontextusában klasszikusan javasolt, jellemzően beépített automatizmusként megvalósított futásidejű mintákat. A két „jó gyakorlat” gyűjtemény alapvető közös pontjai, hogy

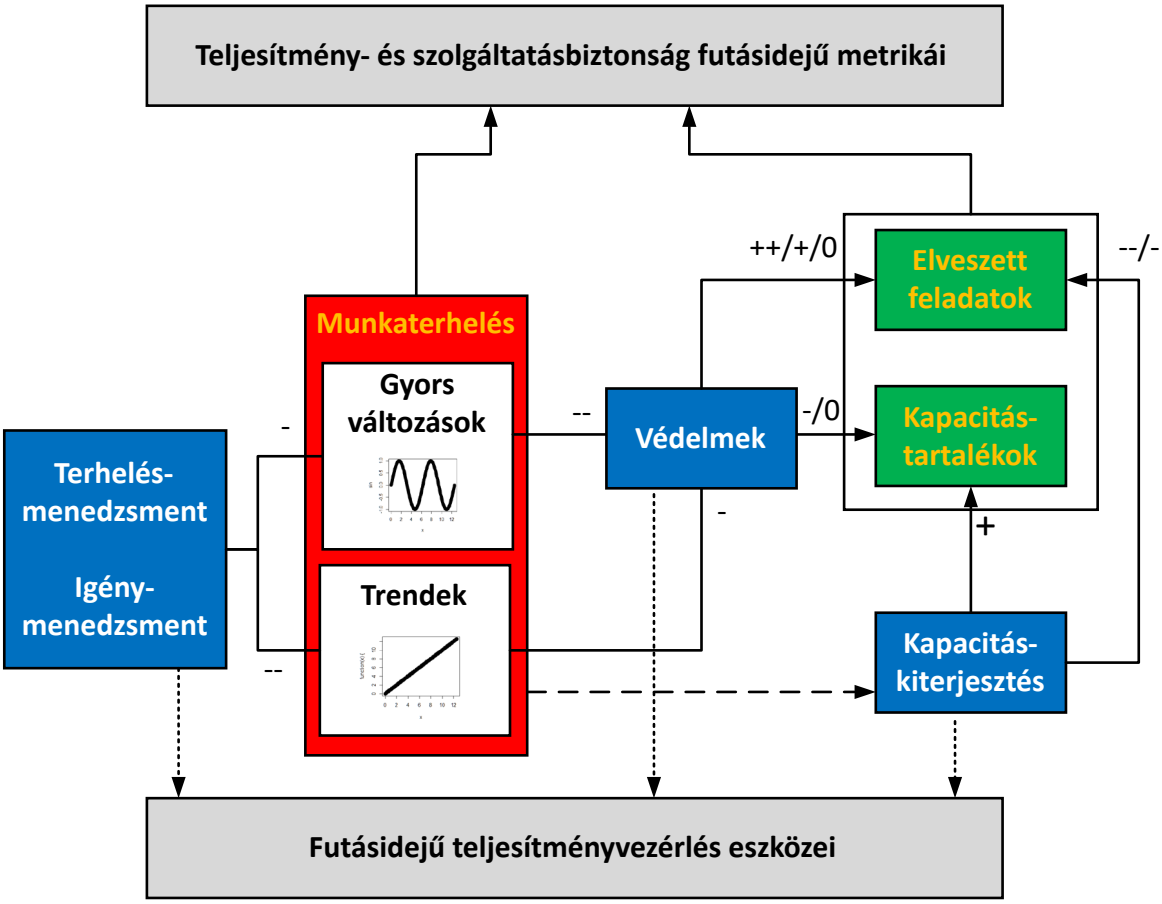
mindkét esetben egy eszközkészlet kerül meghatározásra, amelyekből monitorozás alapú menedzsment, illetve visszacsatolt vezérlés alakítható ki.

Az autonóm teljesítménymenedzsmentre tekinthetünk úgy, mint ezen tevékenységek és tervezési aspektusaik egy részének futásidejű, eljárásrend-vezérelt megvalósítására. Közel sem véletlen, hogy az autonóm számítástechnika és a mesterséges intelligencia közötti kapcsolatot tudományos igénnyel először megalapozó, kiemelt jelentőségű cikkükben Kephart és Walsh [KEPHART–WALSH 2004] az autonóm eljárásrendek szintjeit és azok kapcsolatát egy teljesítménymenedzsment példával szemléltetik.

Érdemes látnunk, hogy az autonóm teljesítésmenedzsment valódi komplexitását nem kizárólag az intelligens módszerek (kiemelt jelentőséggel: futásidejű trajektóriatervezés és parametrikus szabályzójel-számítás) bevezetése adják. Hanmer [HANMER 2013] több helyen kiemeli, hogy az előre figyelembe nem vett hibák miatt a védelmek hatékonysága sokszor esetleges, így egy komplex rendszer hibatűrése nemcsak hogy architekturális értelemben rétegzett kell hogy legyen, de jellemzően eszkalációs mechanizmusokat is kell hogy tartalmazzon.

Ha egy hibás állapotot egy lokális védelem nem tud korlátozni abban, hogy a rendszer nagyobb részére (legrosszabb esetben a szolgáltatásra) hatással legyen, akkor eszkalációra van szükség: a hibás állapotot egy invazívabb megközelítéssel próbáljuk kezelni továbbra is lokálisan, vagy egy következő rendszerfelbontási szintre engedjük tovább a kezelését. Emellett bizonyos megoldások időhorizontjukban egymást kiegészítők; a gyors, de képességeikben gyorsan „kimerülő” reakciók elég időt biztosíthatnak a hosszabb végrehajtási idejű változások végrehajtásához. Ily módon egy autonóm megoldás esetén nemcsak az a feladat például, hogy a foglalt felhőerőforrások skálázási logikáját (adott esetben klasszikus értelemben vett szabályozási körét) megtervezzük, de az is, hogy a beléptetés jellegű védelmeket – amik önmagukban hordozhatnak eszkalációt – létrehozzuk és parametrizáljuk, terhelés- és igénymenedzsmentet vezessünk be, vagy a hosszabb távú pénzügyi hatékonyságot biztosítsuk a védelmek fenntartása mellett (lásd pl. [KOCSIS–MANN–ZILAHI s. a.]).

Bár ezek a mechanizmusok egy-egy adott „hibamód” kezelését vagy elkerülését célozzák meg, egymásra is hatással vannak, így a teljes autonóm viselkedés megfelelőségvizsgálata során sokszor igen összetett kölcsönhatásaikat is vizsgálnunk szükséges. Így igaz az, hogy az autonóm teljesítménymenedzsment tervezése valós feladatok esetén nem pusztán különálló mechanizmusok alkalmazásának tervezését jelenti, hanem határozott stratégiai tervezési aspektussal is rendelkezik. Ezt demonstrálja a 10. ábra a teljesítmény- és igénymenedzsment, a védelmek és kapacitáskiterjesztések a terhelésre, az elveszett feladatokra (lost load) és a kapacitástartalékokra gyakorolt közvetlen és közvetett hatásainak megvilágításán keresztül. Az ábra fogalmai és üzenetei szándékoltan önmagukban is érthetők; megjegyezzük azonban, hogy a következő fejezet részletesen is tárgyalja a szokásosan alkalmazott védelmi és kapacitáskiterjesztési módszereket.



10. ábra. A teljesítménymenedzsment-mechanizmusok interakciója a terhelés, az elveszett feladatok és a kapacitástartalékok közös befolyásolásán keresztül

Forrás: a szerzők szerkesztése

Itt mélyebben nem tárgyaljuk, de a stratégiai tervezést alapvetően befolyásolja a teljesítményhibákat elégségesen előrejelző futásidejű metrikák kiválasztása és megfigyelhetősége. Ezt a problémát nehezíti, hogy ezen jellemzők működési módonként (pl. a túlterhelési állapot függvényében) változnak is [PALJAK et al. 2009; 2010].

Általánosabb értelemben is igaz az, hogy összetett, több rendszer- és minőségi aspektust is kezelő, rétegzett stratégiákra van szükség ahhoz, hogy egy informatikai rendszer a változó terheléshez, hibaterheléshez és környezethez alkalmazkodni tudjon szolgáltatásnyújtásának legfeljebb korlátozott minőségromlásával. Ezt a tulajdonságot hívjuk ellenálló képességnek (resilience; lásd például [LAPRIE 2008]).

Így modern megközelítésben a teljesítménymenedzsment stratégiai tervezése túlmutat az SLR-ek/SLA-k betartásának követésén és a megfelelőség lehetőleg proaktív kikényszerítésén. El kell fogadnunk, hogy egy szolgáltatást érhetnek olyan változások és behatások, amelyek kezelésére – vagy hatásuk teljes elfedésére – az optimális szolgáltatásminőség fenntartása mellett a szolgáltatás nincs felkészítve, vagy amelyek kezelése érzékelhető szolgáltatásromlás nélkül elméletileg is lehetetlen a szolgáltatás számára rendelkezésre álló erőforrásokkal.

A névleges működési tartománytól eltérő szolgáltatásminőségi metrikaértékek nyilvánvalóan nem azonos mértékben problematikusak. Hétköznapi példával megvilágít-

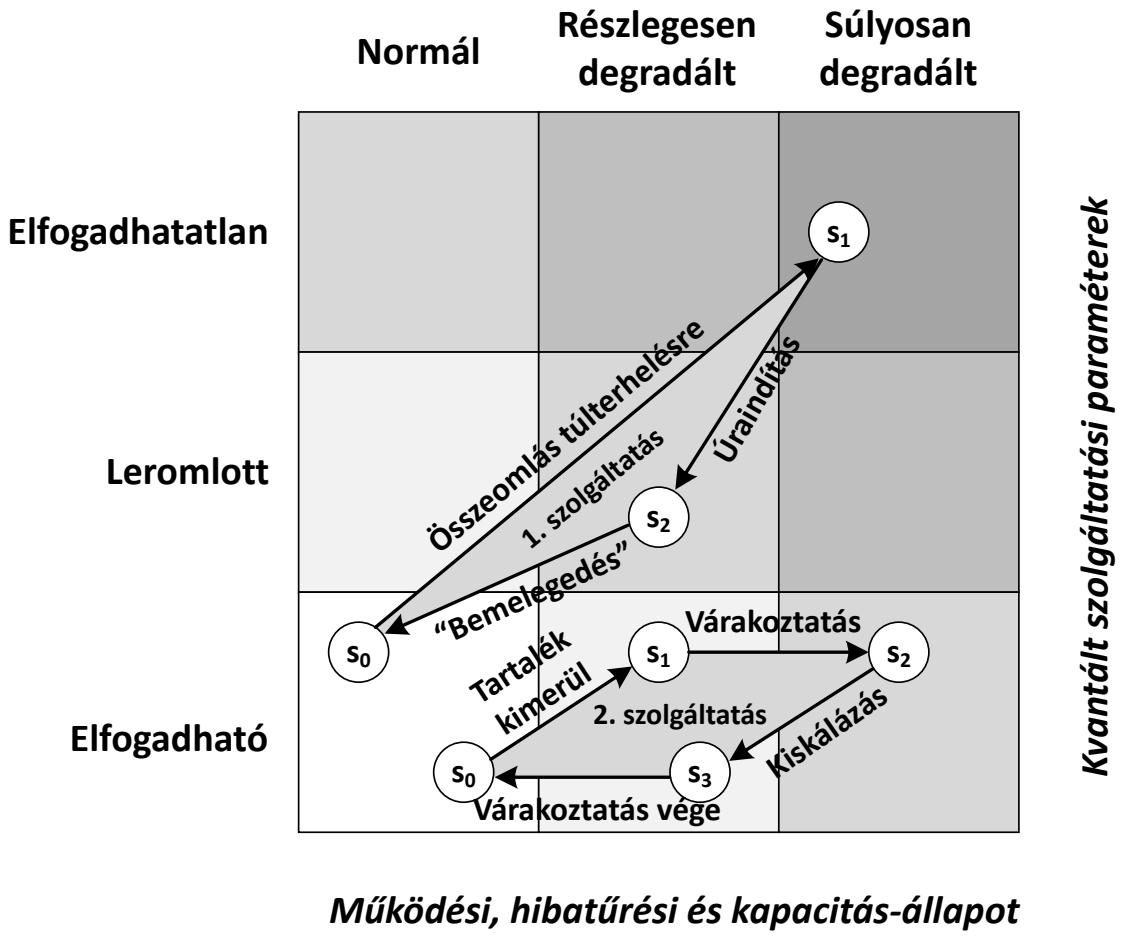
va: egy, a felhasználó számára a szokásosnál lassabban betöltődő weboldal megnőtt válaszideje jelenthet SLA-sértést, de közel sem jelent akkora problémát, mint a 10-20 másodperces, vagy még ennél is magasabb kiszolgálási idők. Különösen igaz ez, ha a probléma akár a hibaok megszűnése, akár a gyors felügyeleti beavatkozás miatt átmeneti (tranziens). Így a nemmegfelelőség sok esetben alkalmasan kategóriákba sorolható – optimális esetben a hibajelenség súlyossága szerint. A rendszer működési, hibatűrési és tartalékkapacitás-viszonyai is levetíthetők egy állapotkészletre.

A nem tökéletesen, de legalább részlegesen ellenálló rendszer működése, vagy a működésével kapcsolatos elvárásaink e két állapotváltozó-kategória mentén reprezentálhatók automata-szemléletben, az úgynevezett ellenállósági állapottérben (resilience state space) leírt trajektóriákkal. Ezt szemlélteti a 11. ábra két, túlterhelés esetén különbözőképpen viselkedő rendszer példáján.

Az 1. szolgáltatás a rosszul védett szolgáltatás példája: túlterhelés esetén összeomlik és a rendszerfelügyeleti mechanizmusoknak (pl. autonóm ágens) kell újraindítania. Az újraindítás után a névleges szolgáltatásminőség és belső erőforráshasználat ismételt eléréséhez a rendszernek túl kell még esnie a bekapcsolási tranziensen és a „bemelegedésen” is (utóbbi jellemzően a különböző belső köztestárak feltöltődését jelenti). Míg ez megtörténik, a rendszer leromlott minőséggel (pl. az elvártnál jóval magasabb válaszidő) nyújt csak szolgáltatást.

A 2. szolgáltatás esetén figyeljük meg, hogy a rendszer nem hagyja el az elfogadható szolgáltatásminőség kategóriáját, hanem belső állapota degradálódik csupán. (További túlterhelés persze okozhatná az elfogadható kategóriából kilépést – a bemutatott modell nem teljes, csupán szemléltető példa.) Ezt úgy éri el, hogy a tartalék erőforrások kimerülése után elkezdi várakoztatni a kéréseket és egyben el is kezdi az alkalmazás erőforrásainak kiskálázását.

Mint korábban említettük, ez mind nyilvános vagy privát felhő, mind pedig klasszikus adatközponti környezetben lehetséges általában. A kiskálázás befejeződésével a belső állapot kritikussága megszűnik, a várakoztatási eljárásrend feloldható; amennyiben a várakoztatás nem vezet SLA-sértéshez az új erőforrások beállítása előtt, úgy a váratlan terhelésnövekményt a felhasználók által érzékelt negatív hatás nélkül kezeltük.



11. ábra. Ellenállósági állapottér és két, túlterhelés esetén
különbözőképpen viselkedő rendszer reprezentációja.
Forrás: a szerzők szerkesztése

Figyeljük meg, hogy egy ellenállósági stratégia megfogalmazására alkalmas eszközt adnak az autonóm számítástechnika különböző eljárásrendjei.

- A beavatkozási szabályok alkalmasak a különböző, a stratégiába foglalt megoldások előfeltétel-vezérelt érvénybe léptetésére vagy semlegesítésére.
- A célállapot-eljárásrendek alkalmazhatósága egyértelmű, különösen akkor, ha az ellenállósági állapottér és átmenetei a gyakorlatban is kezelhető módon megragadhatók.
- A hasznossági függvényre jórészt ugyanazok a megfontolások igazak, mint amiket korábban ismertettünk. Az ellenállósági stratégia tervezésének jellemzője azonban, hogy a hasznossági függvény felállítása jóval egyszerűbb feladat, mint az egyes ellenállósági állapotok „részét képező" teljesítményszabályozási mechanizmusok esetén, különösen, ha az ellenállósági állapottér már előállt.

Vegyük észre, hogy a 11. ábrán színezéssel jelölt demonstratív „izokritikussági" régiók a mellékátlóval azonos irányú vonalak mentén önmagukban egy kvalitatív inverz hasznossági függvényt – azaz költség-, vagy büntetőfüggvényt (cost function, penalty function) – adnak. Ennek folytonos finomításai lehetségesek, de nem feltétlenül szükségesek. Az üzleti igényeknek megfelelően egyéb (adott esetben részleges) sorrendezések is elképzelhetők.

Általánosan problémát a különböző aspektusok közötti – például szolgáltatásminőségi állapot és működési állapot –, az üzleti igényeket hűen tükröző sorrendezés jelenthet, különösen, ha a modellezés a példánkban használt két dimenziónál finomabb. Az ehhez hasonló problémákra azonban az operációkutatás többkritériumos döntéselemzés (multiple-criteria decision analysis, MCDA; lásd pl. [GUITOUNI–MARTEL 1998]) ága számos eljárással rendelkezik.

6.4. A szabályozástechnika eszközeit alkalmazó megoldások

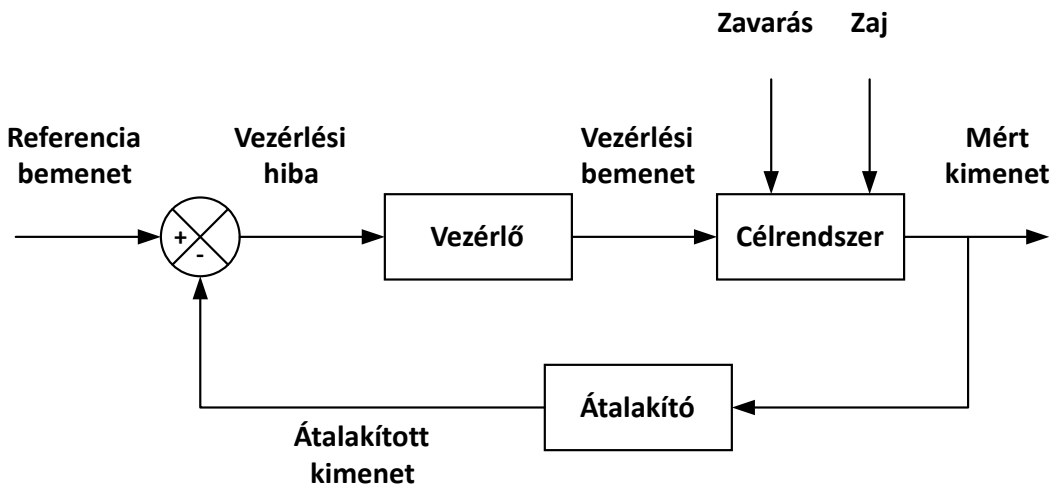
Az IT-rendszerekkel megvalósított szolgáltatások a szabályozástechnika eszközeivel tervezett vezérlése mint paradigma jórészt a 90-es években és a 2000-es évek első felében alakult ki, bár távolról sem előzmények nélkül. Az időszak kiemelt jelentőségű, szintetizáló munkájának minősül [HELLERSTEIN et al. 2004].

A szakterület alapötlete – visszatekintve – szinte kézenfekvő: a fizikai, elektromos és elektronikus rendszerek vezérlésének sémái és ezek tervezési módszerei elméletben alkalmazhatók számítógépes rendszerekre is, ha a megfelelő futásidejű beavatkozókat azonosítani és használni tudjuk, valamint a rendszer viselkedésére megfelelő, akár közelítő modelltípust tudunk találni, és annak paramétereit alkalmas megfigyelések alapján identifikálni.

A 12. ábra a visszacsatolt szabályozás (feedback control) egy lehetséges blokkmodelljét ábrázolja. A modellben néhány elemhez érdemes rövid magyarázatot fűznünk:

- A referenciabemenet az az érték, amit a mért kimenet szándékunk szerint fel kell hogy vegyen.
- A vezérlő definíció szerint az az elem, ami a vezérlési hiba – a referenciabemenet és az átalakított kimenet különbsége – jelenlegi és múltbeli értékei alapján meghatározza a vezérlési bemenetet.
- Zavarásnak (disturbance) minősül minden olyan változás, ami befolyásolja a vezérlési bemenet és a mért kimenet közötti kapcsolatot.
- A zaj a visszacsatolt szabályozás ezen interpretációjában elsődlegesen a mérési hibára és a „szenzorzajra” vonatkozik.
- Az átalakító (transducer) szerepe, hogy a mért kimenetet a referencia bemenettel közvetlenül összevethető formára alakítsa.

A blokkmodell (illetve az előrecsatolt szabályozás blokkmodellje, amelyet itt nem tárgyalunk) alkalmazásával számos autonóm vezérlési probléma megoldása került publikálásra. Ezekről 2004-ig [HELLERSTEIN et al. 2004] tudományos igényességű, a szabályozástervezési módszerek alkalmazását is tárgyaló áttekintést ad. A séma alkalmazásának egy klasszikus és közérthető példáját adja [PAREKH et al. 2001], amelyet [HELLERSTEIN et al. 2004] kivonatol és didaktikus magyarázatokkal lát el.

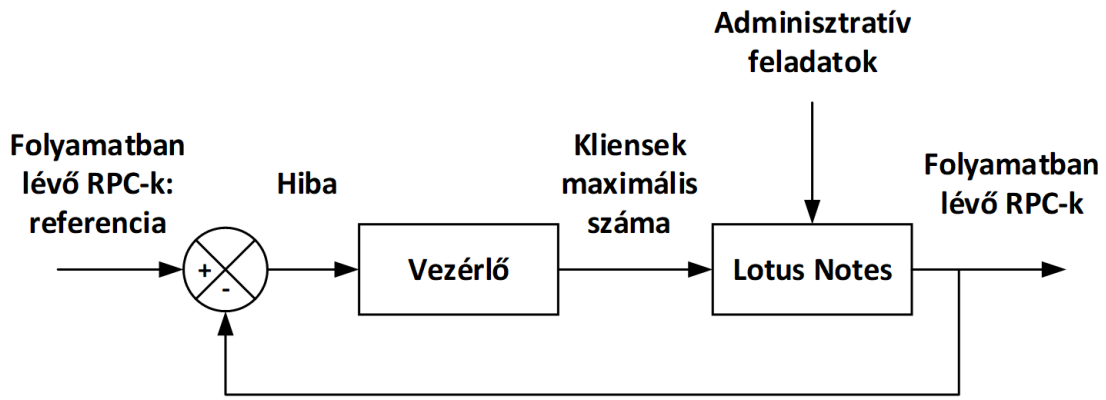


12. ábra. A visszacsatolt szabályozás egy lehetséges blokkdiagramja.
Forrás: [HELLERSTEIN et al. 2004], 5. oldal alapján a szerzők szerkesztése

A Lotus Notes levelezés- és csoportmunka-kiszolgáló aktuális erőforrásigényét a kiszolgáló kliensalkalmazásai által indított és párhuzamosan feldolgozott távoli eljárás-hívások (remote procedure call, RPC) száma határozza meg alapvetően. A kiszolgáló erőforrás-kapacitások kimerülése kerülendő, mivel az lassulásokhoz és egyéb hibákhoz vezet. A kliensek és az RPC-feladathalmaz azonban nem reprezentálhatók egy-egyértelmű leképezéssel; egyrészt egy kliens működésében meghatározó a „gondolkodási idő", másrészt időről időre adminisztratív tevékenységek futtatása szükséges, amelyek RPC-terhelést generálnak, de kliens hez nem köthetők. Az RPC-k számát így leginkább a rendszerbe „beengedett" kliensek számával alkalmas szabályozni. Ezt demonstrálja a visszacsatolt szabályozási sémán a 13. ábra. [PAREKH et al. 2001] a rendszer működését ezek után egy autoregresszív mozgóátlagos (autoregressive moving average, ARMA) modellel közelíti, definiál egy integráló (I) szabályozót, vizsgálja a szabályozás minőségi paramétereit, különös tekintettel a stabilitásra, és kísérleti úton validálja a megközelítést.

A 13. ábrán bemutatott blokkséma egyben azt is demonstrálja, hogy a különböző teljesítménymenedzsment-technikákhoz kialakítható lenne egy szabályozási minta és modellezési-paraméteridentifikációs jó gyakorlat gyűjtemény, nagyban hasonlóan [HANMER 2013]-hoz. Ez a szerzők tudomása szerint még nem történt meg.

A [HELLERSTEIN et al. 2004] kiadása óta eltelt másfél évtized fő eredményeiről pl. [FILIERI et al. 2017] és [SHEVTSOV et al. 2017] ad áttekintést. Ezek alapján elmondható, hogy a szabályozástechnika autonóm számítástechnikai alkalmazása továbbra is aktív kutatások tárgyát képezi, de messze nem vált a hétköznapi mérnöki vagy akár kutatásfejlesztési gyakorlat részévé.



13. ábra. Visszacsatolt szabályozási séma alkalmazása Lotus Notes kiszolgáló folyamatban lévő konkurrens távoli eljáráshívásainak referenciaértékre szabályozására

Forrás: [HELLERSTEIN et al. 2004], 14. oldal alapján a szerzők szerkesztése

6.5. Autonóm okos város?

Az előző fejezet fő üzenete az volt, hogy egy okos város szolgáltatási hálójában a teljesítmény tervezése és menedzsmentje megvalósítható klasszikus megoldások adaptálásával is. Jelen fejezet azt mutatta be, hogy a főleg emberi folyamatokat alkalmazó rendszermenedzsment-megközelítés intelligens automatizálására, illetve eljárásrendekkel vezérlésére milyen megközelítéseket dolgozott ki az informatika.

Egyszerű lenne azt mondanunk, hogy az okos város szolgáltatásait leképezve autonóm elemekre egy összességében nemcsak okos, de egyben „autonóm várost" is kapunk. Ez azonban nem realisztikus; az okosváros-szolgáltatást nyújtó szereplőitől nem várható el általánosságban, hogy megoldásaikat autonóm működésre készítsék fel.

Az itt bemutatott megoldások lehetséges szerepe az okos városokban meglátásunk szerint a következő lehet.

1. Az okos város stratégiai tervezése során az önkormányzatnak a kritikusnak ítélt szolgáltatásokra érdemes egy ellenállósági állapotteret és azon belül a szolgáltatások igényelt viselkedését meghatároznia. Ez a stratégia részeként egyértelműsíti, hogy az egyes szolgáltatások esetén minőségileg milyen viselkedést vár el a város a szolgáltatóktól annak érdekében, hogy a szolgáltatászavarok a lehető legkisebb hatással legyenek összességében a város életére.
2. A szolgáltatást nyújtó szereplőkkel közösen az önkormányzat kialakítja a konkrét eljárásrendeket, amelyek az ellenállósági állapottérben az átmeneteket vezérlik. Ehhez minimálisan modellezési szinten segítséget nyújtanak az autonóm számítástechnika ismert megközelítései.
3. A szolgáltatást nyújtó fél a vállalt SLA-knak és a vállalt ellenállósági eljárásrendnek megfelelően kialakítja belső teljesítménymenedzsment-mechanizmusait, melynek során alkalmazhat autonóm számítástechnikai módszereket is.

Fontos azonban kiemelnünk, hogy míg véleményünk szerint a klasszikus teljesítménymenedzsment elemeinek beépítése az okos város életciklusába elkerülhetetlen lesz az okosváros-megoldások széles körű elterjedésével, addig az autonóm működés jellemzően az „érett" okos városok sajátja lesz.

7. A túlterheléskezelés automatizmusai

A szolgáltatásbiztos számítástechnika (dependable computing) régi felismerése, hogy a szolgáltatásteljesítményt és a szolgáltatásbiztonság klasszikus attribútumait [AVIŽIENIS et al. 2004] – főként a megbízhatóságot, rendelkezésre állást és konzisztenciát – befolyásoló tényezők a legtöbb rendszerben komoly átfedésben vannak. A „kemény” hibák, mint például fizikai számítási elemek kiesése vagy a különböző szoftverhibák, hajlamosak csökkenteni a teljesítményt; és fordítva, a túlterhelés nemcsak a teljesítményt degradálja, de sokszor a szűken vett szolgáltatásbiztonság értelmében is hibahatásokhoz vezet.

A szolgáltatásbiztonság elemzése során azt vizsgáljuk, hogy a hibaokok (faults) milyen hibás állapotokhoz (errors) vezethetnek, amik aztán szolgáltatás szintjén milyen hibahatásban (failure) manifesztálódnak; ebben a fogalmi rendszerben az elérhető kapacitásokat meghaladó terhelés egyfajta – jellemzően tranziens, átmeneti – hibaokként fogható fel, amely túlterhelt állapothoz vezet.

A szolgáltatásbiztos számítástechnika mint diszciplína céljai közé tartozik, hogy a különböző jellegű hibák elfedésére, de legalábbis az általuk okozott hibás állapotból visszaállásra megoldásokat adjon. Ez alól nem képeznek kivételt a túlterhelés jellegű hibák sem. Hasonlóan a szoftvertervezéshez, az általánosított problémákra adott, „jó gyakorlat” jellegű válaszokat itt is mintanyelvként (pattern language) rendszerezi és alkalmazza a szakma, szolgáltatásbiztonsági és hibatűrési mintaként hivatkozva ezekre (dependability and fault tolerance patterns). A klasszikus, újrakonfigurálható (felhő) számítástechnikát nem alkalmazó és nem kiberfizikai jellegű szolgáltatásokra pl. [HANMER 2013] ad egy jó áttekintést a hibatűrő számítástechnika alapvető mintáiról. Jelen fejezet [HANMER 2013] a „hibás állapot áthidalása” (error mitigation) mintanyelv részeként definiált, a túlterhelések tűrését célzó módszereit mutatja be. A mintákat itt csupán kivonatoljuk, részletesebb tárgyalásukat lásd [HANMER 2013]-ban és az ott hivatkozott további forrásmunkákban.

Megjegyzendő, hogy a minták alkalmazása feltételezi, hogy a hibatűrő működés részeként a túlterhelés detektálása előzetesen megtörténik, és az alapvetően viselkedési – tehát „elindítandó”, nem strukturális – módszerek alkalmazását megelőzi a megfelelő diagnózis.

7.1. Elnapolható munka

A tipikus informatikai rendszerek fő tevékenységük mellett időszakos jelleggel, ütemezett módon karbantartási jellegű tevékenységeket is végeznek. Ez a rendszer jellegétől függően többek között az archiválástól a konzisztenciaellenőrzésen keresztül a különböző kötegelt, nem időkritikus információcserékig terjedhet. Az elnapolható munka (deferrable work) minta alapgondolata, hogy amennyiben a rendszer a „hasznos”, időkorláton belül kiszolgálandó terhelés tekintetében még nem merítette ki kapacitásait, úgy ezen tevékenységek elnapolásával felszabadítható annyi kapacitás, ami a túlterhelt állapotot normalizálja, és időt biztosít a további mechanizmusok aktiválására.

7.2. Megfelelő erőforrás-megosztás

A „megfelelő erőforrás-megosztás” (equitable resource allocation) egy alapvető minta az informatikai infrastruktúra-menedzsmentben. Meghatározó szerepű például a hálózati szolgáltatásminőség-menedzsmentben [EVANS–FILSFILS 2007] és a virtualizált rendszerek erőforrás-allokációja során (pl. másodpercenkénti CPU ciklusok arányos megosztása virtuális gépek között; lásd pl. [CHERKASOVA–GUPTA–VAHDAT 2007]). A minta által megfogalmazott jó gyakorlat lényege, hogy érdemes az informatikai rendszer által „elvégzendő feladatokat” kategóriákba (*pools*) rendezni és a véges erőforrásokat ezen kategóriák között osztani meg, sokszor dinamikusan (például a kategóriákhoz rendelt szolgáltatási szintek fontossága alapján).

A mintát alkalmazási, illetve szolgáltatás szinten is alkalmazzuk; jellemzően akkor, amikor meg kívánjuk előzni, hogy nagyszámú, nem kritikus feladat megakadályozza a kritikus feladatok időszerű végrehajtását. Megjegyzendő, hogy a megfelelő erőforrás-megosztás tervezése esetén külön figyelmet kell fordítani a prioritásinverzió [TANENBAUM–BOS 2014] elkerülésére.

A megfelelő erőforrás-megosztás minta alkalmazása tipikusan az „erőforrás várakozási sorok” (queue for resources) mintával együtt történik; amennyiben az erőforráskorlátok miatt az összes beérkező feladat végrehajtását nem tudjuk a beérkezéssel egyidejűleg megkezdeni, úgy a kéréseket sorban (queue) várakoztatjuk. Megjegyzendő, hogy a várakozási sorok emellett egy alapvető hibatűrés-támogató rendszerintegrációs minta is; a küldő és a fogadó szereplő hibáinak hatása üzenetsorok alkalmazásával sokkal korlátozottabb, mint a szinkron kommunikáció esetén.

7.3. Automatikus védelmek

A komplex rendszerek viselkedése túlterhelés – azaz a véges feldolgozási kapacitást meghaladó terhelés – esetén sokféle lehet. A megvalósítás belső logikájától függően a túlterhelés összeomláshoz vezethet; ez okozhat akár adatvesztést vagy a tárolt adatok konzisztenciájának sérülését is. Túlterhelés miatti összeomlásnál a rendszer újraindítása sem jelent önmagában kielégítő megoldást, ha a túlterhelés nem pillanatnyi tüske, hanem trendszerű; a helyreállás után – miközben az összeomlás és a helyreállás között a rendszer nem volt elérhető, így rövid és középtávú rendelkezésre állása (availability) jelentősen csökkenhetett – a továbbra is fennálló túlterhelés miatt ismét összeomlás várható.

Más rendszerek az átmeneti túlterheléseket beépített várakozási sorokkal és intelligens erőforrás-átütemezéssel képesek lehetnek ugyan összeomlás nélkül áthidalni, ám a szolgáltatás összteljesítménye (pl. az átlagos késleltetés) ezekben az esetekben is sérülhet, akár a szolgáltatásiszint-szerződések megsértéséhez vezetve. Általánosságban igaz az (bár nem kivételek nélkül), hogy minél közelebb kerül egy erőforrás a telítettséghez, annál nagyobb az erőforrás-menedzsment összköltsége a hasznos terhelés kiszolgálásán felül. Rendszerfelügyeleti szempontból pedig nem hagyhatjuk figyelmen kívül a tényt, hogy a legtöbb szoftverrendszert „normál” terhelésre tervezik – a rendszer működése túlterhelés esetén sokszor kiszámíthatatlan, nem dokumentált és a gyártó által nem feltétlenül tesztelt.

Megjegyzendő, hogy ez alól kivételt jelentenek a klasszikus értelemben kritikus, például telekommunikációs rendszerek és a hálózati alapszolgáltatások; az utóbbiak ipari minőségű megvalósításai pontosan az alapszolgáltatás tulajdonság miatt jellemzően legalább kvalitatíve ismert viselkedéssel rendelkeznek túlterhelés esetén is. Mindemellett

ezen rendszerek sem terhelhetők „a végtelenségig”; egy digitális telefonközpont specifikációja jellemzően a vállalt kapacitás (telekommunikációs terminológiában: engineered capacity, [BAUER–ADAMS 2013]) párszorosáig írja elő (illetve garantálja) a viselkedést szolgáltatásbiztonság és teljesítmény szempontjából.

Mindezekből levonható a következtetés, hogy a komplex rendszereket érdemes a végrehajtott feladatok, kérések mennyiségi korlátozása képességével felruházni. Ez az „automatikus védelmek” (protective automatic controls) minta alapelve: abban az esetben, amikor a rendszerben valamely véges erőforrás szaturációközeli állapotba kerül vagy már kimerült, a korlátozások életbe léptetésével a rendszert teljesítmény (és erőforráshasználat) szempontjából az ismert és a minőségi követelményeket kielégítő tartományban lehet tartani.

7.3.1. Munkamegosztás

Az automatikus védelmek tervezésének és megvalósításának számos mintája ismert. A „munkamegosztás” (share the load) minta a rendszer más részeiben még esetlegesen megtalálható, eredetileg egyéb feladatokhoz rendelt, szabad kapacitások bevonását javasolja. Alkalmazhatóságát korlátozza, hogy jellemzően csak alacsony fokú szinkronizációt alkalmazó feladatok esetén hatékony. Napjaink szoftverdefiniált infrastruktúráinak képességeit figyelembe véve (felhők, konténerkörnyezetek [MERKEL 2014], szoftverdefiniált hálózatok [NUNES et al. 2014]) pedig sokszor egyértelműen kifizetődőbb egy rendszert eleve közösített szabad kapacitások kihasználására képessé tenni (lásd az automatikus kapacitáskiterjesztést a következő pontban), mint a munkamegosztás támogatására.

7.3.2. Terheléseldobás

A munkamegosztást kiegészítő, napjainkra jóval nagyobb szerepű megoldás a „terheléseldobás” (shed the load). Alapötlete, hogy az el nem végzett munkának nincs erőforrásigénye; így feladatok eldobásával is normál terheléstartományban tartható egy rendszer. Általában jó gyakorlat a terhelést a rendszer perifériáján eldobni (shed work at periphery); így nemcsak hogy minimalizáljuk a feleslegesen elvégzett munkát, de aktív visszautasítás esetén a feladat végrehajtását igénylő kliensek is a lehető leggyorsabban értesülnek a túlterhelésről. Így nemcsak az „újrapróbálkozásokból” adódó további terhelésnövekményt tudjuk csökkenteni, de alkalmazási szinten használva a mintát a felhasználói elégedettség romlása (Quality of Experience, QoE csökkenése) is jelentősen kisebb lehet – egy e-kereskedelmi szolgáltatás felhasználóját egy összeválogatott termékkosár elvesztése érzékenyebben fogja érinteni, mintha a vásárlási folyamatot csak az oldalhoz később visszatérve tudja megkezdeni. A rendszer perifériáján végzett munkaterhelés-menedzsmentre szokás „beléptetésmenedzsment”-ként (admission control) is hivatkozni.

Ha a beléptetésmenedzsmentet egyszerű munkamennyiség-korlátokkal (thresholding) vezéreljük, akkor könnyű belátni, hogy trendszerű túlterhelésnél a kéréseket sokszor „löketekben” engedjük be a rendszerbe. Ennél alacsonyabb felhasználói elégedettségromlással járó stratégia lehet lépcsőzetesen egyre komolyabb beléptetési korlátozásokat bevezetni, egyre jobban „lassítva” a túlterhelt állapot felé elmozdulást (a minta neve Slow it down, amit magyarul talán a „terhelésnövekmény-lassítás” ad leginkább vissza).

Web-alkalmazások (pl. híroldalak) esetén ennek a stratégiának egy manifesztációja, ha az oldal komoly túlterhelés esetén először csak a képekre vonatkozó kéréseket

tagadja meg (esetleges további priorizálással a képek között és/vagy először azoknak csak a kis felbontású változatát kiszolgálva), majd tovább növekvő terhelés esetén elkezd valamilyen prioritási logika alapján szelektálni a weboldalszövegekre vonatkozó kérések között is. A terhelés csökkenése esetén a korlátozások visszavonását jó gyakorlat bevezetésükhöz képest egy hiszterézis-görbe mentén végezni.

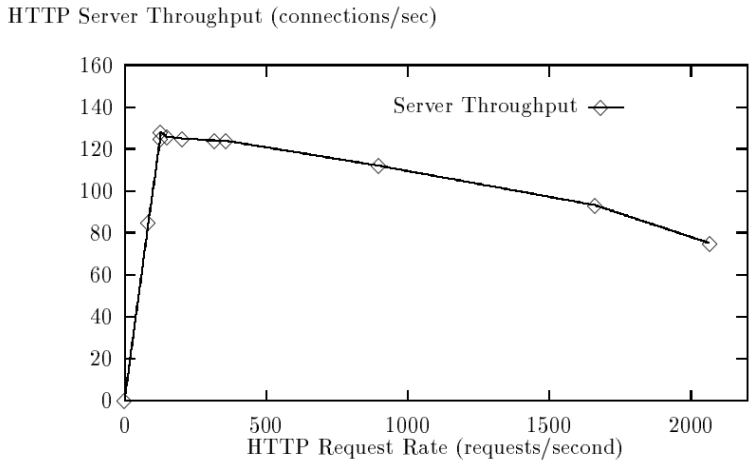
A terheléseldobás minta további kiegészítése, hogy a belépő kérések visszautasítása mellett jellemzően érdemes a már végrehajtás alatt álló feladatok befejezésére is kifejezetten törekedni (Finish work in progress).

7.3.3. Alkalmazási kérdések

Az automatikus védelmek nagy előnye, hogy vezérlési késleltetésük jellemzően alacsony. Triviális hátrányuk emellett, hogy a terhelés növelésével a véges erőforrások „védelme” miatt a ki nem szolgált kérések mennyisége monoton növekszik, ami önmagában SLA-sértést okozhat. Ennek megfelelően alkalmazásuk napjainkban leginkább a kapacitáskiterjesztéssel párosítva történik, különösen olyan szoftverdefiniált infrastruktúrákban, ahol tartalékkapacitások hozzáférhetőek, de a tartalékkapacitások beállítása viszonylag időigényes művelet.

Ennek egy példáját adják az Infrastructure-as-a-Service (IaaS) felhők, ahol egy új virtuális gép elindítása és a már működő virtuális gépekkel közös klaszterbe állítása percekig is tarthat. Az automatikus védelmek gyors bekapcsolással és nagyon magas megbízhatósággal képesek az ismert és megfelelő viselkedésű tartományban tartani a rendszert, amíg a kapacitáskiterjesztés megtörténik.

Lényeges tulajdonsága az automatikus védelmeknek, hogy hétköznapian fogalmazva „nincsenek ingyen”. Tekintsük példaként a 14. ábrán szereplő, klasszikus kísérletet, ahol egy tipikus webkiszolgáló feldolgozási rátája látható a kérések beérkezési rátájának függvényében [BANGA–DRUSCHEL 1999]. A terhelés és az áteresztőképesség a szaturációig megegyezik; ezután a kiszolgáló tovább üzemel, de (CPU) erőforrásainak egyre nagyobb részét emészti fel a ki nem szolgált kérések eldobásának kezelése, ami összességében azt eredményezi, hogy még a kapacitásnak megfelelő maximális áteresztőképességet sem tudja tartani a túlterhelés növekedésével. Ennek megfelelően az automatikus védelmek tervezése során figyelmet kell fordítani arra, hogy a védelem számára megfelelő dedikált kapacitások álljanak rendelkezésre.



14. ábra. Tipikus webkiszolgáló áteresztőképessége a kérések beérkezési rátájának függvényében
Forrás: [BANGA–DRUSCHEL 1999], 10. oldal

7.4. Automatikus kapacitáskiterjesztés

Az automatikus kapacitáskiterjesztés (expansive automatic controls) alapgondolatában az automatikus védelmek ellentéte: a véges erőforrások védelme helyett a munkába állított erőforrások kiterjesztését javasolja a tervezett kapacitást meghaladó terhelés kiszolgálására. Ez történhet dedikált tartalék erőforrásokkal vagy a terhelés kiszolgálására alkalmas alternatív (egyben potenciálisan szuboptimális) megoldások átmeneti engedélyezésével.

Az első lehetőség beépített, vagy viszonylag gyorsan „beszerezhető", a tervezett terheléshez viszonyítva redundáns tartalék erőforrásokat feltételez. A felhő és felhő jellegű (konténer-„rajok", edge és fog számítástechnika) környezetekben az új erőforrások igény szerinti „beszerzése" alapfunkcionalitás – a magán- és nyilvános felhők között ebből a szempontból legfeljebb annyi a különbség, hogy magánfelhőben nem feltétlenül „praktikusan végtelen" a lefoglalható új erőforrások mennyisége.

Fontos rámutatnunk, hogy nem felhőalkalmazásoknál sem azonos a kapacitáskiterjesztés a specifikusan a maximális túlterhelésre tervezéssel; az automatikus kapacitáskiterjesztésre allokált rendundanciák használata akkor a leghatékonyabb, ha több szolgáltatás számára egy közös tartalék kontingens áll rendelkezésre.

Amennyiben a túlterhelési jelenségek valószínűségi értelemben elégségesen függetlenek, úgy a statisztikai multiplexálás szokásos logikája alapján a közösített tartalék költséghatékony és megbízható védelmet tud nyújtani. Az érdeklődő olvasó figyelmébe ajánljuk [WEINMAN 2012] 15. fejezetét, ahol a párhuzamos igények statisztikai multiplexálásának hatását az összes erőforrásigény variációs koefficiensére – a szórás osztva a várható értékkel – a centrális határeloszlás tétel alkalmazása nélkül, elemi eszközökkel mutatják meg. Historikus érdekesség, hogy a közösített kapacitáskiterjesztési tartalékot mint megoldást már a felhőplatformok alkalmazásának elterjedése előtt széles körben alkalmazták; már 2005-ben léteztek kiforrott rendszerek adatközponti kapacitások ily módon kezelésére (lásd például a [KEPHART–DAS 2007]-ben eszközként alkalmazott nagyvállalati adatközpont-automatizációs és irányítási technológiákat).

Az alternatív kiszolgálási módszerek mint elv alkalmazhatósága sokkal esetlegesebb, mint a tartalékbekapcsolásé és nagyban alkalmazásfüggő. Ide sorolható például különböző hálózatokban (telefónia, IP, de például terepi szövedékhálózatok – mesh networks – is) az optimális, de telítődő útvonal mellett további, hosszabb/lassabb/stb. útvonalak engedélyezése.

7.5. „Kemény" hibák teljesítményhibaként kezelése

Üzemeltetési szempontból mind a nyújtott, mind az igénybe vett szolgáltatások teljesítményének megfelelőségét két fő tényező veszélyezteti: a (pillanatnyi) kapacitást meghaladó igények és azok a hibaokok (faults) és hibaállapotok (errors), amelyek kihatással vannak a teljesítményre. Utóbbira közismert alacsony szintű példa a „memóriaszivárgás" (memory leak), amely ugyan egy idő után általában összeomlás jellegű hibákhoz vezet, de ennek bekövetkeztéig is radikálisan tudja csökkenteni egy rendszer egészének teljesítményét.

A teljesítményhatással járó „kemény" – logikai, fizikai és konfigurációs – hibák (hard faults) kezelésére és azok elkerülésére (pl. a „software rejuvenation" technika segítségével [HANMER 2010]) általános jellegű útmutatást az üzemeltetési stratégia szintjén nehézkes lenne megfogalmazni. Ezeket a hibákat a szolgáltatást megvalósító rendszer-

nek képesnek kell lennie tűrni, vagy ezekből képesnek kell lennünk azt helyreállítani – pl. egy ITIL-alapú folyamattal.

Modern, megfelelően tervezett rendszerekben a detektálás-diagnosztika-tervezés-helyreállítás folyamatát a rendszer önállóan is képes lehet végezni; erről a lehetőségről az autonóm számítástechnikát taglaló részben ejtettünk bővebben szót. Itt a következőket emeljük ki.

A kapacitástúllépési hibák hatása futásidejű automatizmusok segítségével részlegesen tűrhető, de legalábbis csökkenthető. Mind a teljesítményhiba-tűrésre, mind pedig a teljesítmény hibahatás-csökkentésre kodifikált tervezési minták állnak rendelkezésünkre.

A teljesítményhatással rendelkező kemény hibák aktiválódása sok esetben jól interpretálható klasszikus túlterhelésként is – a különbség pusztán annyi, hogy a kapacitás csökkent, és nem a terhelés nőtt meg. Így mindazon megoldások, amelyek alkalmasak a terhelésnövekményből adódó kapacitástúllépés automatikus kezelésére, jelenthetnek átmeneti megoldást a kemény hibák esetén is. Abban az értelemben persze óvatosnak kell lennünk, hogy a klasszikus túlterhelést jellemzően vagy átmeneti hibaokként értelmezzük és kezeljük, vagy pedig olyan mechanizmust alkalmazunk a kezelésére, amely kapacitásátrendezés és -növelés segítségével egyben meg is szünteti a hibaokot. Ezzel szemben a kemény hibák esetén ezek a mechanizmusok a hibaok eltávolítására nem képesek, sokszor még a hibás állapot megszüntetésére sem.

8. Összefoglalás

Munkánk áttekintette az okosváros-szolgáltatások IT-teljesítménymenedzsmentjének főbb kérdéseit és alkalmazható megoldásait. Igyekeztünk áttekintő képet adni, amely az okosváros-módszertantól a városi szintű tervezésen és irányításon keresztül az egyes okosváros-megoldásokban alkalmazható mechanizmusokig terjed.

Kismonográfiánk számos releváns területet még csak nem is érint. Az alacsony szintű műszaki kérdések (pl. felműszerezés és monitorozás) esetén ez kifejezetten szándékunk is volt; széles célközönség számára ezeket a kérdéseket nehézkes és koncepcionális szempontból nem is lényeges bemutatni.

Más területek tárgyalásáról – ilyenek a kockázatvezérelt tervezés, a teljesítménymenedzsment Big Data adatelemzéssel támogatása és a terhelés-túlterhelések előrejelzése és kezelése (lásd pl. [BOZÓKI–KORONKA–PATARICZA 2015]) – terjedelmi okokból kellett lemondanunk.

Kismonográfiánkat nem csak az üzemeltetéssel és fejlesztéssel foglalkozó informatikai szakembereknek szántuk. Reményeink szerint haszonnal forgathatja mindenki, aki okos város informatikai megoldások életre hívásán dolgozik, akár a közszolgálatban, akár az iparban.

9. Hivatkozások

ALBINO, Vito – BERARDI, Umberto – DANGELICO, Rosa Maria (2015): Smart Cities: Definitions, Dimensions, Performance, and Initiatives. *Journal of Urban Technology*, vol. 22. no. 1. 3–21. Taylor & Francis

AMATO, Giuseppe – CARRARA, Fabio – FALCHI, Fabrizio – GENNARO, Claudio – MEGHINI, Carlo – VAIRO, Claudio (2017): Deep Learning for Decentralized Parking Lot Occupancy Detection. *Expert Systems with Applications,* vol. 72, Apr.. 327–334. Elsevier

Amsterdam Smart City (s. a.), forrás: https://amsterdamsmartcity.com/ (A letöltés dátuma: 2018. 01. 30.)

AVIŽIENIS, Algirdas – LAPRIE, Jean-Claude – RANDELL, Brian – LANDWEHR, Carl (2004): Basic concepts and taxonomy of dependable and secure computing. *IEEE Transactions on Dependable and Secure Computing,* vol. 1. no. 1. 11–33.

BALLARD, Chuck – FARRELL, Daniel M. – LEE, Mark – STONE, Paul D. – THIBAULT, Scott – TUCKER, Sandra (2010): *IBM InfoSphere Streams Harnessing Data in Motion.* IBM Redbooks.

BANGA, Gaurav – DRUSCHEL, Peter (1999): Measuring The Capacity Of A Web Server Under Realistic Loads. *World Wide Web,* vol. 2. no. 1–2. 69–83. Springer

BÁNYÁSZ Péter – BUKOVICS István – DERS Csaba – ZÁGON Csaba – DOMBORÓCZKY Zoltán – ENDRŐDI István – FÁBOS Róbert – FÖLDI László – GYARMATI József – HORVÁTH Attila – MOLNÁR Ferenc – NÉMETH József Lajos – PINTÉR István – SZABOLCSI Róbert – SZÁSZI Gábor – TÓTH Bálint (2013): *Fejezetek a kritikus infrastruktúra védelemből.* Tanulmánykötet. Magyar Hadtudományi Társaság, Budapest.

BASILI, Victor R. – CALDIERA, Gianluigi – ROMBACH, H. Dieter (1994): Experience Factory. In MARCINIAK, John J. ed.: *Encyclopedia Of Software Engineering*, 528–532. John Wiley & Sons

BAUER, Eric – ADAMS, Randee (2013): *Service Quality of Cloud-Based Applications.* Wiley-IEEE Press.

BBJ, Budapest Business Journal (2016): Researchers say Hungary has 3.3 mln smartphone users. Forrás: https://bbj.hu/economy/researchers-say-hungary-has-33-mln-smartphone-users_109296 (A letöltés dátuma: 2018. 01. 30.)

Berlin Smart City (2015): Smart City Strategie Berlin. Forrás: http://www.almanac-project.eu/downloads/M2M_Workshop_Presentations/Session%204/Mia_Copenhagen_smart_city_2015.pdf (A letöltés dátuma: 2018. 01. 30.)

BISWAS, Kamanashis – MUTHUKKUMARASAMY, Vallipuram (2016): *Securing Smart Cities Using Blockchain Technology.* Proceedings of the 2016 IEEE 18th International Conference on High Performance Computing and Communications; IEEE 14th International Conference on Smart City; IEEE 2nd International Conference on Data Science and Systems (HPCC/SmartCity/DSS). IEEE

BKK Futár (s. a.): Android alkalmazás. Forrás: https://play.google.com/store/apps/details?id=hu.webvalto.bkkfutar&hl=hu (A letöltés dátuma: 2018. 01. 30.)

BONOMI, Flavio – MILITO, Rodolfo – ZHU, Jiang – ADDEPALLI, Sateesh (2012): *Fog Computing and Its Role in the Internet of Things.* Proceedings of the First Edition of the MCC Workshop on Mobile Cloud Computing.

BOZÓKI, Szilárd – KORONKA, Gábor – PATARICZA, András (2015): Risk Assessment Based Cloudification. *Software Engineering for Resilient Systems, Lecture Notes in Computer Science,* vol. 9274. 71–81. Springer, Cham

BROOKS, Peter (2006): *Metrics for IT Service Management.* Van Haren, Zaltbommel.

Budapest Smart City (2017): Smart Budapest: Budapest Okos Város Jövőképe. Forrás: http://budapest.hu/Documents/V%C3%A1ros%C3%A9p%C3%ADt%C3%A9si%20F%C5%91oszt%C3%A1ly/Smart_Budapest_%C3%B6sszefoglal%C3%B3_HUN.pdf (A letöltés dátuma: 2018. 01. 30.)

CHERKASOVA, Ludmila – GUPTA, Diwaker – VAHDAT, Vahdat (2007): Comparison of the three CPU schedulers in Xen. *ACM SIGMETRICS Performance Evaluation Review,* vol. 35. no. 2. 42–51.

CMMI Product Team (2010): CMMI for Services, Version 1.3., Technical Report CMU/SEI-2010-TR-034. Software Engineering Institute, Carnegie Mellon University, Pittsburgh, Pennsylvania

Copenhagen Smart City (s. a.), forrás: http://www.almanac-project.eu/downloads/M2M_Workshop_Presentations/Session%204/Mia_Copenhagen_smart_city_2015.pdf (A letöltés dátuma: 2018. 01. 30.)

Cyber Physical Systems PWG (2015): CPS PWG Framework V0.8. Forrás: https://pages.nist.gov/cpspwg/library/ (A letöltés dátuma: 2018. 01. 30.)

Ericsson (2017): Ericsson Mobility Report. Forrás: https://www.ericsson.com/assets/local/mobility-report/documents/2017/ericsson-mobility-report-june-2017.pdf (A letöltés dátuma: 2018. 01. 30.)

EVANS, John William – FILSFILS, Clarence (2007): *Deploying IP and MPLS QoS for Multiservice Networks: Theory & Practice.* Morgan Kaufmann Publishers Inc., San Francisco, CA, USA

FILIERI, Antonio – MAGGIO, Martina – ANGELOPOULOS, Konstantinos – D'IPPOLITO, Nicolás – HEMPEL, Andreas Berndt – HOFFMAN, Henry – JAMSHIDI, Pooyan – KA-

LYVIANAKI, Evangelia – KLEIN, Cristian – KRIKAVA, Filip – MISAILOVIC, Sasa – PAPADOPOULOS, Alessandro V. – RAY, Suprio – SHARIFLOO, Amir M. – SHEVTSOV, Stepan – UJMA, Mateusz – VOGEL, Thomas (2017): Control Strategies For Self-Adaptive Software Systems. *ACM Transactions on Autonomous and Adaptive Systems,* vol. 11. no. 4. article 24. ACM, New York, NY, USA

GIFFINGER, Rudolf – PICHLER-MILANOVIĆ, Nataša (2007): *Smart cities: Ranking of European Medium-sized Cities.* Centre of Regional Science, Vienna University of Technology. Forrás: http://www.smart-cities.eu/download/smart_cities_final_report.pdf (A letöltés dátuma: 2018. 01. 30.)

GÖNCZY, László – MAJZIK, István – BOZÓKI, Szilárd – PATARICZA, András (2016): MDD-Based Design, Configuration and Monitoring of Resilient CPS. In ROMANOVSKY, Alexander – ISHIKAWA, Fuyuki eds.: *Trustworthy Cyber-Physical Systems Engineering.* 395–420. Chapman and Hall/CRC, New York, NY, USA

GREGG, Brendan (2013): *Systems Performance: Enterprise and the Cloud*. Pearson Education.

GUITOUNI, Adel – MARTEL, Jean-Marc (1998): Tentative Guidelines To Help Choosing An Appropriate MCDA Method. *European Journal of Operational Research,* vol. 109. no. 2. 50–521. Elsevier

HAMARI, Juho – SJÖKLINT, Mimmi – UKKONEN, Antti (2016): The sharing economy: Why people participate in collaborative consumption. Journal of the Association for Information Science and Technology, vol. 67. no. 9. 2047–2059. Wiley Online Library

HANMER, Robert (2010): *Software Rejuvenation.* Proceedings of the 17th Conference on Pattern Languages of Programs, Article 21.

HANMER, Robert (2013): *Patterns for Fault Tolerant Software.* John Wiley & Sons.

HAREN, Van (2011): *TOGAF Version 9.1.* Van Haren Publishing.

HELLERSTEIN, Joseph L. – DIAO, Yixin – PAREKH, Sujay – TILLBURY, Dawn M. (2004): *Feedback Control of Computing Systems.* John Wiley & Sons

HOYT, Robert Eugene – SNIDER, Dallas – THOMPSON, Carla – MANTRAVADI, Sarita (2016): IBM Watson Analytics: Automating Visualization, Descriptive, and Predictive Statistics. *JMIR Public Health Surveillance,* vol. 2. no. 2. JMIR Publications Inc., Toronto, Canada

INTERNATIONAL ORGANIZATION FOR STANDARDIZATION (2011): Information technology - Service management - Part 1: Service management system requirements. ISO/IEC 20000-1:2011

ISACA, Information Systems Audit and Control Association (s.a.): COBIT 5. Forrás: http://www.isaca.org/cobit/pages/default.aspx (A letöltés dátuma: 2018. 01. 30.)

KEPHART, Jeffrey O. – CHESS, David M. (2003): The Vision Of Autonomic Computing. *Computer*, vol. 36. no. 1. 41–50. IEEE

KEPHART, Jeffrey O. – DAS, Rajarshi (2007): Achieving Self-Management via Utility Functions. *IEEE Internet Computing,* vol. 11. no. 1. 40–48. IEEE

KEPHART, Jeffrey O. – WALSH, William E. (2004): *An artificial intelligence perspective on autonomic computing policies.* Fifth IEEE International Workshop on Policies for Distributed Systems and Network. 3–12. IEEE

KOCSIS Imre – PATARICZA András – HUSZERL Gábor – IZSÓ Benedek – SZATMÁRI Zoltán – TÓTH Áron – VARRÓ Dániel – VÖRÖS András (2014): Informatikaoktatás felhőben: egy új oktatási modell bevezetése. In RÉVAI Nóra szerk.: *BeleSTEM - Felsőoktatási jó gyakorlatok a tudomány, a technológia, a műszaki tudományok és a matematika szolgálatában,* 36–41. Tempus Közalapítvány, Budapest, Magyarország

KOCSIS, Imre – MANN, Zoltán Ádám – ZILAHI, Dávid (s. a.): Optimized deployment of critical applications in Infrastructure-as-a-Service clouds. *International Journal of Cloud Computing,* vol. 6. p. 342. *Inderscience.*

KOCSIS, Imre – SALÁNKI, Ágnes – PATARICZA, András (2016): Measurement-based identification of infrastructures for TCPS. In ROMANOVSKY, Alexander – ISHIKAWA, Fuyuki eds.: *Trustworthy Cyber-Physical Systems Engineering.* p. 369–392. Chapman and Hall/CRC.

KOF (s. a.): Kormányzati Felhő. Forrás: http://kof.hu/ (A letöltés dátuma: 2018. 01. 30.)

LANE, Nicholas D. – MILUZZO, Emiliano – LU, Hong – PEEBLES, Daniel – CHOUDHURY, Tanzeem – CAMPBELL, Andrew T. (2010): A survey of mobile phone sensing. *IEEE Communications Magazine*, vol. 48. no. 9. 140–150. IEEE

LAPRIE, Jean-Claude (2008): *From Dependability To Resilience.* 38th IEEE/IFIP International Conference On Dependable Systems and Networks, G8-G9.

Lechner Központ (2017): Okos város fejlesztési modell, módszertani útmutató. Forrás: http://lechnerkozpont.hu/doc/okos-varos/okos-varos-fejlesztesi-modell-modszertani-utmutato-20171130.pdf (A letöltés dátuma: 2018. 01. 30.)

Lechner Központ (s. a.): Okos város. Forrás: http://okosvaros.lechnerkozpont.hu/hu (A letöltés dátuma: 2018. 01. 30.)

LEE, Edward A. – RABAEY, Jan – BLAAUW, David – DUTTA, Prabal – FU, Kevin – GUESTRIN, Carlos – HARTMANN, Björn – JAFARI, Roozbeh – JONES, Doug – KUBIATOWICZ, John – KUMAR, Vijay – MANGHARAM, Rahul – MURRAY, Richard M. – PAPPAS, George – PISTER, Kris – ROWE, Anthony – SANGIOVANNI-VINCENTELLI, Alberto – SESHIA, Sanjit A. – ROSING, Tajana Simunic – TASKAR, Ben – WAWRZY-

NEK, John – WESSEL, David (2014): The Swarm at the Edge of the Cloud. *IEEE Design & Test,* vol. 31. no. 3. 8–20. IEEE

LESKOVEC, Jure – RAJARAMAN, Anand – ULLMAN, Jeffrey David (2014): *Mining of Massive Datasets.* Cambridge University Press.

LIGHTSTONE, Sam S. – LOHMAN, Guy – ZILIO, Danny (2002): Toward Autonomic Computing With Db2 Universal Database. *Acm Sigmod Record,* vol. 31. no. 3. 55–61.

LORIDO-BOTRAN, Tania – MIGUEL-ALONSO, Jose – LOZANO, Jose A. (2014): A Review Of Auto-Scaling Techniques For Elastic Applications In Cloud Environments. *Journal of Grid Computing,* vol. 12. no. 4. 559–592. Springer

MERKEL, Dirk (2014): Docker: Lightweight Linux Containers for Consistent Development and Deployment. *Linux Journal 2014*, no. 239. article 2.

MURCH, Richard (2004): *Autonomic Computing*. IBM Press.

NUNES, Bruno Astuto A. – MENDONCA, Marc – NGUYEN, Xuan-Nam – OBRACZKA, Katia – TURLETTI, Thierry (2014): A Survey Of Software-Defined Networking: Past, Present, And Future Of Programmable Networks. *IEEE Communications Surveys & Tutorials,* vol. 16. no. 3. Third Quarter, 1617–1634. IEEE

OFFICE OF GOVERNMENT COMMERCE (2007A): ITIL - Service Strategy. The Stationary Office, London, UK.

OFFICE OF GOVERNMENT COMMERCE (2007B): ITIL – Continual Service Improvement. The Stationary Office, London, UK.

ORIOL, Marc – MARCO, Jordi – FRANCH, Xavier (2014): Quality models for web services: A systematic mapping. *Information and Software Technology,* vol. 56. no. 10. 1167–1182. Elsevier

PALJAK, Gergely János – ÉGEL, Zoltán – TÓTH, Dániel – KOCSIS, Imre – KOVÁCS-HÁZY, Tamás – PATARICZA, András (2010): *Qualitative performance control in supervised IT infrastructures.* International Conference on Dependable Systems and Networks Workshops. 59–65. IEEE

PALJAK, Gergely János – KOCSIS, Imre – ÉGEL, Zoltán – TÓTH, Dániel – PATARICZA, András (2009): Sensor Selection for IT Infrastructure Monitoring. In *International Conference on Autonomic Computing and Communications Systems*, p. 130–143. Springer, Berlin, Heidelberg

PAREKH, Sujay – GANDHI, Neha – HELLERSTEIN, Joseph – TILBURY, Dawn – JAYRAM, Thathachar – BIGUS, Joe (2001): *Using control theory to achieve service level objectives in performance management.* IEEE/IFIP International Symposium on Integrated Network. IEEE

PATARICZA, András (2007): Systematic generation of dependability cases from functional models. In *Proceedings of Formal Methods for Automation and Safety in Railway and Automotive Systems* (FORMAT 2008), Budapest, Hungary.

Mell, Peter – GRANCE, Tim (2011): The NIST Definition of Cloud Computing. *NIST Special Publication.* 800–145.

PMI, Project Management Institute (s. a.): PMBOK Guide and Standards. Forrás: https://www.pmi.org/pmbok-guide-standards (A letöltés dátuma: 2018. 01. 30.)

SALÁNKI, Ágnes – KINCSES, Gergő – GÖNCZY, László – KOCSIS, Imre (2017): Data Analysis Based Capacity Planning of VCL Clouds. *International Journal of Cloud Computing,* vol. 6. num. 4. pp. 370-383. Inderscience Publisher

SHEVTSOV, Stepan – BEREKMÉRI, Mihály – WEYNS, Danny – MAGGIO, Martina (2017): Control-Theoretical Software Adaptation: A Systematic Literature Review. *IEEE Transactions on Software Engineering,* vol. 44. no. 8. p. 784-810. IEEE

STERBENZ, James P. G. – HUTCHINSON, David – ÇETINKAYA, Egemen K. – JABBAR, Abdul – ROHRER, Justin P. – SCHÖLLER, Marcus – SMITH, Paul (2010): Resilience And Survivability In Communication Networks: Strategies, Principles, And Survey Of Disciplines. *Computer Networks,* vol. 54. no. 8. 1245–1265. Elsevier

Stockholm Smart City (s. a.): Welcome to the smartest city in the world. Forrás: http://international.stockholm.se/globalassets/ovriga-bilder-och-filer/smart-city/welcome-to-the-smartest-city-in-the-world-english-designfiction-sthlm-stad.pdf (A letöltés dátuma: 2018. 01. 30.)

SUN, Jianjun – YAN, Jiaqi – ZHANG, Kem Z. K. (2016): Blockchain-Based Sharing Services: What Blockchain Technology Can Contribute To Smart Cities. *Financial Innovation,* vol 2, issue 1. SpringerOpen

TANENBAUM, Andrew S. – BOS, Herbert (2014): *Modern Operating Systems,* 4th edition. Prentice Hall Press, Upper Saddle River, NJ, USA

TM Forum (2013): Multi-Cloud Service Management Pack: Service Level Agreement (SLA) Business Blueprint, TR 197, V1.4. Forrás: https://joinup.ec.europa.eu/solution/tr197-multi-cloud-service-management-pack-sla-business-blueprint-frameworx-multi-cloud (A letöltés dátuma: 2018. 01. 30.)

WEINMAN, Joe (2012): Cloudonomics: *The Business Value of Cloud Computing.* John Wiley & Sons.

A Nemzeti Közszolgálati Egyetem kiadványa

Kiadó:

Nemzeti Közszolgálati Egyetem
Közigazgatási Továbbképzési Intézet
www.uni-nke.hu

Felelős kiadó:
Prof. Dr. Kis Norbert rektorhelyettes
Címe: 1083 Budapest, Üllői út 82.

Olvasószerkesztő:
Dorogi Katalin

Tördelőszerkesztő:
Friebert Máté

ISBN 978-963-498-394-1 (elektronikus)