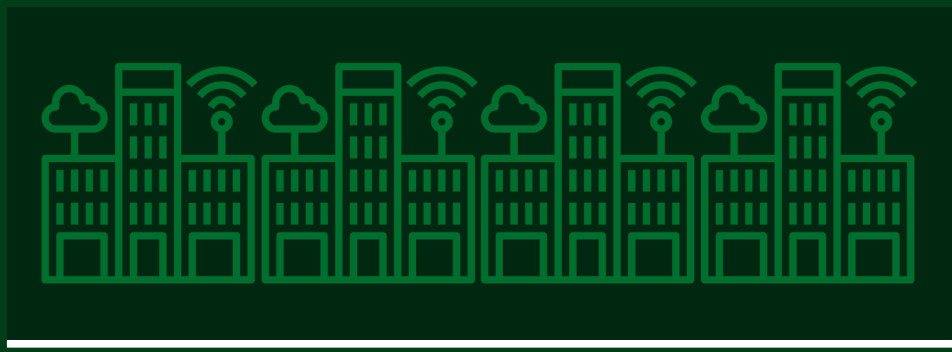
OKOSVÁROS-TECHNOLÓGIÁK XVIII.

Okos szolgáltatások teljesítménybenchmarkolása és értékelése



Klenik Attila

NEMZETI KÖZSZOLGÁLATI EGYETEM
BUDAPEST

SZÉCHENYI 2020
MAGYARORSZÁG KORMÁNYA
Európai Unió
Európai Szociális Alap
BEFEKTETÉS A JÖVŐBE

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

Sorozatszerkesztő:
Sallai Gyula

Klenik Attila

OKOS SZOLGÁLTATÁSOK TELJESÍTMÉNYBENCHMARKOLÁSA ÉS ÉRTÉKELÉSE

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:
Sallai Gyula, professor emeritus, BME-VIK

A kézirat lezárásának dátuma:
2018. január 31.

TARTALOMJEGYZÉK

1. BEVEZETÉS

A tárgyak vagy dolgok internete (Internet of Things, IoT) egy rohamosan fejlődő terület, melynek célja, hogy egyesítse a fizikai környezetből származó adatokat a kibervilágban elérhető módszerekkel és információkkal. A „fogyasztótól üzletig” jellegű (Consumer to Business, C2B) alkalmazások teszik ki az IoT-piac egy jelentős részét, ahol a fogyasztók állítanak elő értéket az üzletek számára, ellentétben a hagyományos üzleti modellekkel. Az ilyen jellegű alkalmazások a fogyasztók széles körében végezhetnek adatgyűjtést.
Az adatok felhasználásának egyik iránya az egyénenkénti feldolgozás, annak érdekében, hogy a hagyományos üzleti folyamatok automatizálhatóak legyenek. Például az egyes fogyasztók havi energiafelhasználásának automatikus leolvasása egy okosmérőóra segítségével egy teljesen automatizált számlázási folyamat alapvető előfeltételét képezi. Az ilyen jellegű üzleti automatizációk a gyakorlatban jelentősen csökkenthetik az adminisztratív költségeket, amelyek adott esetben a mérőórák személyes leolvasásával járnak. Az adatok felhasználásának másik iránya az aggregált feldolgozás, amely lehetővé teheti az energiafogyasztási trendek rövid és hosszú távú megfigyelését, megalapozva ezzel például a regionális erőforrás-kihasználtság becslését.
Azonban a nagyszámú adatforrás kezeléséhez, az adatok megszerzéséhez és feldolgozásához folyamatosan és megbízható módon rendelkezésre álló adatok és azok feldolgozására képes megfelelő számítási kapacitás szükséges (Byun – Park, 2011, Balakrishna, 2012). Emiatt az IoT-rendszerek megfelelő méretezése egy fontos tényező a megfelelő szolgáltatásminőség nyújtásában.
Az IoT-rendszerek kritikus természete szükségessé teszi robusztus és általános megoldások (Romanovsky – Ishikawa, 2016) alkalmazását a tervezési, telepítési és karbantartási fázisokban. Kiemelkedően fontos egy olyan architektúra összeállítása, amely teljesíti az IoT-rendszerek komplex követelményeit, mint skálázhatóság, robusztusság, biztonságosság és titkossági kényszerek.
Az információtechnológia (IT) területén különböző módszerek léteznek arra, hogy általános célú komponenseket a kívánt cél érdekében egy komplex rendszerré komponáljunk. Az egyik leginkább alkalmazott megközelítés a benchmarkolás, amelynek célja a különböző megoldásverziók objektív összehasonlítása teljesítmény és ár/teljesítmény szempontok alapján. A publikusan elérhető benchmarkeredmények rendkívül hasznosak egy bizonyos integrálandó részkomponens kiválasztásakor. Egyrészt összevetve a benchmarkeredményeket a tervezés alatt álló rendszer elvárt jellemzőivel, a tervezési tér jelentősen szűkíthető realisztikus komponensjelöltekre. Másrészt a benchmarkeredmények gazdasági szempontból is összehasonlítási alapként szolgálnak a különböző megoldás verziókhoz.
A hosszú hagyományokkal és kifinomult módszerekkel rendelkező benchmarkolás méréstudományi oldalról nézve megfelelő mérési folyamatok segítségével garantálja a mérések reprodukálhatóságát, megismételhetőségét. Habár ez egy alapvető előfeltétele az eredmények általános célú felhasználásának, a tényleges hasznosságot a vizsgált probléma valósághűsége határozza meg, valamint az, hogy egy adott kísérleti alkalmazás során szerzett eredmények mennyire általánosíthatók szélesebb körben.
A következő fejezetek egy komplex IoT-rendszer tervezésének és megfelelő

teljesítményre való méretezésének alapvetői kihívásait mutatják be, amelyek a következők:

- Milyen követelmények alapján kell megtervezni egy földrajzilag nagy kiterjedésű kritikus rendszert?
- A követelmények alapján hogyan tervezzük meg a rendszer architektúráját?
- Az architektúra konkrét hardver- és szoftverelemeit hogyan válasszuk meg?
- Miként tudjuk biztosítani egy megépített rendszer megfelelő teljesítményét?

A 2. fejezet bemutatja a nagy kiterjedésű, elosztott infrastruktúrák kiépítésé során felmerülő kihívásokat. Egységes követelményrendszer és architektúra ajánlások alapján egy szisztematikus módszert adunk okosváros-szolgáltatások kritikus infrastruktúráinak megtervezéséhez.
A 3. fejezet a bemutatott tervezési folyamat alapján előállt infrastruktúraterv teljesítményaspektusaival foglalkozik. Megmutatjuk, hogy a teljesítmény elemzéshez leggyakrabban használt benchmarkolás folyamata alapos megfontolást igényel, illetve hivatalos benchmarkokat mutatunk be a megtervezett architektúra különböző komponenseinek teljesítményelemzéséhez.
A 4. fejezet bemutatja a követelményeknek való megfelelőség kihívásait, különös tekintettel az extrafunkcionális követelményekre, mint például a teljesítmény. A fejezet során bemutatunk egy módszertant, amellyel szisztematikus, könnyen követhető módon tudjuk felmérni egy komplex rendszer teljesítményét. A módszer során, származtatott eredmények alapján, a teljesítmény szempontjából kritikus rendszerkomponensek teljesítményhangolására is kitérünk.
Az 5. fejezet összefoglalja a monográfiában bemutatott témákat, melyeknek mélyebb elsajátításához egy részletes irodalomjegyzék is rendelkezésre áll.

A szerző[1]

[1] *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. OKOSSZOLGÁLTATÁSOK INFRASTRUKTÚRÁJA

Egy okosváros (Zygiaris, 2013, Albino – Berardi – Dangelico, 2015) alapját adó infrastruktúra eleget tesz a kritikus rendszer definíciójának (Knight, 2002). Szolgáltatások meghibásodása, illetve kiesése súlyos következményekkel járhat, kezdve a jelentős anyagi károktól egészen az emberélet veszélyeztetéséig. A város-, vagy akár országméretű kiterjedéssel rendelkező rendszerek megfelelő szolgáltatásminőségre tervezése – tehát mind a funkcionális, mind az extrafunkcionális követelményeknek való megfelelése – kihívásokkal teli feladat. Egy ilyen tervezési folyamat teljes spektrumának lefedése – amely akár a különböző irányító rendszerekbe épített vezérlők időzítéseinek matematikai vizsgálatáig is terjedhet (Ostroff, 1992) – túlmutat a mű keretein. Jelen mű kritikus rendszerek és szolgáltatások architekturális aspektusát taglalja, különös figyelmet fordítva azok teljesítményére.

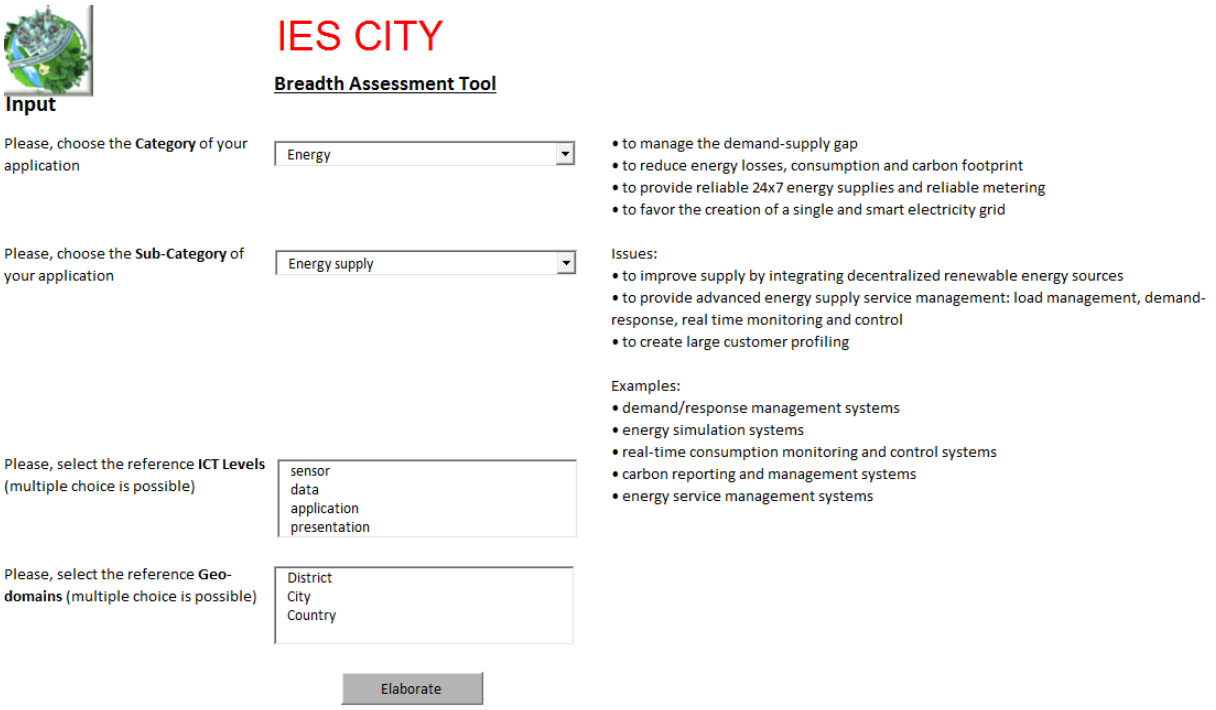
2.1. Egységes követelményrendszer

IoT-rendszerekkel a mindennapok során is gyakran találkozunk, még ha nem is látjuk a rendszert támogató mögöttes komplex infrastruktúrát. Az úthálózatok mentén elhelyezett sebességfigyelő kamerarendszer tipikus példája egy hétköznapi IoT-rendszernek, hiszen különböző érzékelők segítségével vizuális és sebességinformációkat szolgáltat a sebességkorlátot esetlegesen túllépő gépjárművekről. Ezeket az információkat a mögöttes infrastruktúra feldolgozza, és képfeldolgozás (rendszámfelismerés) segítségével automatikusan hozzárendeli a járműtulajdonos adatait. Ez a folyamat manuális módszerekkel nagyságrendekkel több időt venne igénybe.
Az IoT-rendszerek nagy kiterjedésű hálózatai és szolgáltatásai képezik az okosváros piac (Allmendinger – Lombreglia, 2005) alapépítőköveit. Jelenleg a megfelelő minőségű okosváros-megoldások legnagyobb korlátjai a felhasznált inkompatibilis, egyedi tervezésű rendszerkomponensek, valamint az architektúra tervezésére vonatkozó szabványok hiánya, amely a különböző befektetők közötti együttműködést korlátozza.
Ezenkívül nem elhanyagolható az az évek, évtizedek során felhalmozott tudás, amelynek felhasználása (jobban mondva újrahasználása) kritikus lehet egy minőségi szolgáltatás elkészítésében. Gondoljunk például bele, hogy mennyi historikus késési/forgalmi információ áll rendelkezésünkre egy tömegközlekedési szolgáltatás esetén a járművezetők által vezetett naplók alapján. Ezek egy része lehet, hogy nem is került digitalizálásra (csak egy összesítő az adott járatokról vagy napokról).
Szintén példaként említhetjük a budapesti tömegközlekedés új fejlesztését, a BKK Futár (BKK s.a.) alkalmazást és a mögöttes infrastruktúrát. Kombinálva a járműveken elhelyezett szenzorokat (például GPS) és a korábbi évek tapasztalatai (és különböző ütemezési kényszerek (Árgilán et al., 2014)) alapján megtervezett menetrendet, a rendszer valós idejű adatokat szolgáltat a járművek helyzetéről és a várható megállókba való érkezési időkről. Ugyanakkor ezek az információk részben felhasználhatóak lennének egy okos-közlekedésilámpákat irányító rendszer esetén is (a videóalapú forgalom elemzés

mellett (Kanungo – Sharma – Singla, 2014) felhasználva a számtalan tömegközlekedési jármű valós idejű helyzetét).
Régi és új, egymástól nem teljesen függetlenül működő rendszerek integrálásához elengedhetetlen a szabványokra épülő tervezési folyamat. Az interakcióhoz szükséges továbbá a felhalmozott tudás megfelelő jellegű tárolása és cseréje (Sowa 2000, Compton et al., 2012), azonban ezzel a továbbiakban nem foglalkozunk.
A National Institute of Standards and Technology (NIST s.a.) – több partner együttműködésével – elindított egy munkacsoportot, amelynek célja egy referencia-keretrendszer elkészítése IoT-képes okosvárosok számára (NIST-IESCF 2018). IoT-képes okosváros alatt értjük egyrészt a városban elhelyezett fizikaiszenzor-infrastruktúrát, másrészt a szenzorok által szolgáltatott adatok feldolgozására képes mögöttes kiberinfrastruktúrát. Az NIST számtalan területen közreműködik annak érdekében, hogy a nagy jelentőségű fejlődési területeket (mint például az 5G kommunikáció, épület automatizálás) minél hamarabb szabvány szintre emeljék (NIST Impacts s.a., NIST Industry Impacts s.a.). A munkacsoport szerint a keretrendszer célja a következő:

> „A *városi befektetőkkel konzultálva* a munkacsoport összehasonlítja és egységesíti a jelenlegi architekturális törekvéseket; azonosítani fogja a *sarkalatos együttműködési pontokat* a nagyszámú, már telepített architektúrák között, majd elkészít egy *konszenzus keretrendszer dokumentumot* a közös architekturális vonásokról. Ez a dokumentum támpontként fog szolgálni a városoknak, hogy *együttműködésre képes, skálázható* okosváros-megoldásokat telepítsenek, amelyek *megfelelnek a közösség igényeinek*."

A munkacsoport különböző kategóriákat (például energia szektor) és alkategóriákat (energia kereslet, kínálat) állított össze a leggyakoribb okosváros-alkalmazások használati eseteiből. Ezután azonosítottak különböző kontextusokat a kategóriákhoz, például földrajzi tartományokat (lakás, kerület, város, ország stb.), valamint információs és kommunikációs technológiai (ICT) megvalósítási célokat (szenzorok, adat, alkalmazás, adat prezentálása).
A munkacsoport követelményeket fogalmazott meg a különböző kategóriákhoz és kontextusokhoz, figyelembe véve a közöttük lévő kapcsolatokat. A munkájuk egyik közvetlen eredménye egy alkalmazási keretrendszeranalízis-eszköz (NIST-SCA s.a.), amely automatikusan képes absztrakt és specifikus követelmények származtatására a kiválasztott kategóriák és kontextusaiknak különböző aspektusaihoz (például biztonság, titkosság, skálázhatóság) – a legjobb mérnöki gyakorlatokat figyelembe véve és alkalmazva. A hivatkozott (NIST-SCA s.a.) oldalon az egyes követelmények értelmezése is elérhető, így azokat részleteiben nem tárgyaljuk, csak az eszköz alkalmazhatóságát demonstráljuk. Az eszköz egy Microsoft Excel-munkafüzet formájában készült el, amely leírásokat és példákat is szolgáltat a különböző szakterületekhez. Az Excel-munkafüzet felépítését az 1. ábra szemlélteti. A kívánt felhasználási terület megadása után a munkafüzet aspektusonként és felelősségi körök mentén felbontva megadja a tervezéshez szükséges absztrakt (magas szintű) és implementációspecifikus (alacsony szintű) követelményeket. Az energiaszolgáltatás-terület által megkívánt magas szintű követelményeket (aspektusokra és felelősségi körökre bontva) az 1. táblázat foglalja össze.



1. ábra: IoT-képes okosváros követelményfelmérése.
Forrás: a szerző saját szerkesztése

Aspektus	Felelősségi kör	Magas szintű követelmény
Funkcionális	Beavatkozás	• Adatcsere az energiahálózattal • Energiahálózattól kapott adatok feldolgozása, és döntéshozás
	Kommunikáció	• Rendszeren belüli és rendszerek közötti információcsere • Szabványos szenzorhálózat kommunikációs protokollok
	Irányíthatóság	• A rendszer távoli elérhetősége és vezérelhetősége
	Érzékelés	• Perzisztens kommunikáció • Energiahálózattól kapott adatok feldolgozása és döntéshozás
	Megfigyelhetőség	• Azonosítási (autentikációs) mechanizmus • Megfigyelő felület (dashboard)
Üzleti	Minőség	• Információ szolgáltatása az időben történő beavatkozáshoz
	Hasznosság	• Információ szolgáltatása a hatékony terhelés menedzsmenthez
Emberi	Használhatóság	• Olvasható, egyértelmű és aggregált adatok szolgáltatása az emberek számára

Megbízhatóság	Biztonság (Safety)	• Állandó megfigyelés • Információ szolgáltatása az időben történő beavatkozáshoz
	Biztonság (Security)	• Adatokhoz való hozzáférés és közzététel korlátozásának megőrzése • A rendszer módosításának vagy megsemmisítésének megakadályozása • Az adatok megmásíthatatlanságának és eredetiségének biztosítása • Digitális aláírások • Kriptográfia
Időzítés	Logikai idő	• Az események sorrendjének figyelembevétele
	Szinkronizáció	• Adatok továbbítása közös/azonos időskálán • Szenzor szinkronizációs algoritmusok
Adat	Adatszemantika	• Szabványos adatmodellek • Az adatok jelentésének helyes értelmezése
	Adatműveletek	• Különböző forrású adatok harmonizálása • Szabványos elektronikus adat formátumok • Publikus felületek (interfészek)

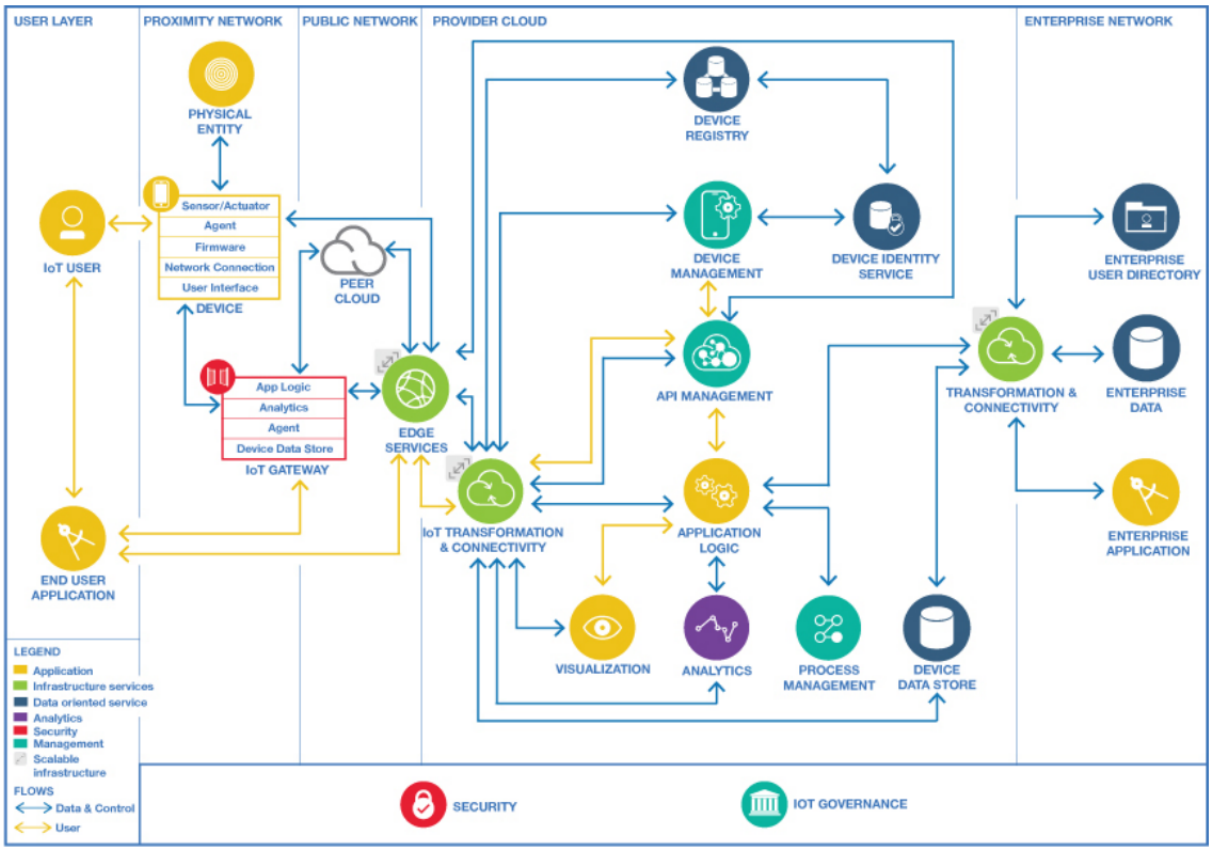
1. táblázat: Az energiaszolgáltatás-kategória magas szintű követelményei

Habár a táblázatban feltüntetett magas szintű követelmények többnyire funkcionalitást írnak elő (például az adatokhoz való hozzáférés és közzététel korlátozásának megőrzése), mégis befolyásolják a rendszer teljesítményaspektusait (egy védelmi szint fenntartása többlet idővel jár a végrehajtási idő szempontjából). Továbbá a következő alfejezetben azt is látni fogjuk, hogy az egyes funkcionális követelményeknek architekturális tervezési következményei vannak, amik tovább nehezítik az előírt teljesítményre tervezést.

2.2. Referenciaarchitektúra IoT-rendszerekhez

Az IoT-rendszereket leginkább úgy szokták jellemezni, mint dedikált eszközök („dolgok”) globálisan összekapcsolt hálózata, amelyek képesek megfigyelni a környezetüket és beavatkozni abba, lehetőleg automatikus módon. Ugyanakkor egy másik közös tulajdonsága ezeknek az eszközöknek a limitált tárolási és számítási kapacitásuk. Ezen limitációk miatt aggodalmak merülhetnek fel az eszközök skálázhatóságával és megfelelő reakcióidejével kapcsolatban. Például időben képesek-e reagálni/beavatkozni egy kritikus környezeti változásra?
A felhőszolgáltatók megoldást adnak ezekre a problémákra (Liu et al. 2016), hiszen praktikusan „végtelen” tárolási és számítási kapacitást kínálnak robusztus és könnyen

skálázható formában (Lorido-Botran – Miguel-Alonso – Lozano 2014). Nyilvánvaló volt, hogy előbb-utóbb megjelennek majd megoldások, amelyek kombinálják az IoT- és felhőalapú számítás világát (Botta et al., 2016). Ugyanakkor ez a lépés a két világ komplexitásait is kombinálta (Kocsis et al., 2013), ezáltal növelve az igényt egy egységes architektúrára felhőalapú IoT-megoldásokhoz.



2. ábra: Felhőalapú IoT-referenciaarchitektúra. Forrás: CSCC 2016

Az igény kielégítésére a Cloud Standards Customer Council (CSCC s.a.) létrehozott egy referenciaarchitektúrát (2. ábra) nagy kiterjedésű IoT-rendszerekhez, amelyek különböző felhőszolgáltatásokra épülnek. Az architektúrának a „Cloud Customer Architecture for IoT" (CCAIoT, CSCC 2016) nevet adták. A kezdeményezés egy teljeskörű IoT-rendszer felépítését lefedi, kezdve az IoT-eszközök felhasználóitól egészen az eszközöket kezelő vállalatokig. Az architektúra alapvető komponensei a következők (egyszerűsítve a 3. ábrán):

- *Felhasználói réteg*, amely magában foglalja az IoT-eszközök használatát lehetővé tevő alkalmazásokat, valamint az alkalmazás felhasználóit. A felhasználói alkalmazások futtatásához jellemzően szükség lehet például okostelefonokra, tabletekre, személyi számítógépekre vagy például specializált irányítópanelekre.
- *Kihelyezett hálózat*, melynek eleme minden olyan entitás, amellyel a rendszer kapcsolatba kerül megfigyelésen vagy beavatkozáson keresztül (például termékek egy gyártósoron), valamint az eszközök, amelyek az IoT-rendszerhez kötik ezen entitásokat. Az eszközök lehetnek szenzorok vagy beavatkozók, valamint speciális IoT-átjárók (gateway), amelyek a rendszer többi részéhez csatolják az eszközöket.

- *Publikus hálózat*, amely kapcsolatot biztosít a különböző kihelyezett hálózatok (és eszközeik) között, jellemzően egy nagy kiterjedésű hálózat (például az internet) segítségével. Szintén idetartoznak a különböző hálózatszéli (edge) szolgáltatások, mint például a tartomány név feloldás, terheléselosztás, stb.
- *Felhőszolgáltatói réteg*, amely alapvető alkalmazásoknak és szolgáltatásoknak ad helyet, mint például adattranszformáció, adattárolás, vizualizáció, IoT-eszköznyilvántartó-szolgáltatások stb.
- *Vállalati hálózat*, ahol azok az alkalmazások helyezkednek el, amelyek üzletspecifikus tudást állítanak elő az IoT-rendszerben végzett megfigyelések alapján, mint például energiafelhasználás-trendek hosszabb távú becslése.

3. ábra: Felhőalapú IoT-referenciaarchitektúra, kivonat. Forrás: a szerző saját szerkesztése

A referenciaarchitektúra kellően általános és rugalmas ahhoz, hogy megfelelő kiindulási alapként szolgáljon IoT-rendszerek széles skálájához. Ugyanakkor szakterület-specifikus problémák esetén szükség lehet az architektúra adaptálására, azaz a ténylegesen szükséges komponensek azonosítására, kiválasztására és megfelelő beállítására. Az architektúra tervezése során kiemelt hangsúlyt kap az IoT-rendszer követelményeinek szisztematikus kiértékelése, ezzel segítve a szükséges referenciaarchitektúra-elemek azonosítását. Ennek azonban alapja egy részletes, szakterület-specifikus követelményrendszer, amely megvalósítástól függetlenül összegyűjti az egyes alkalmazási területek által támasztott igényeket.
Az 2. táblázat bemutatja az NIST eszköze által javasolt követelmények egy részét az energiaszektor területén, valamint egy példa leképezést, amely azonosítja a felhőalapú IoT-referenciaarchitektúra szükséges elemeit a megvalósításhoz. A példa során minden ICT-szint és földrajzi tartomány kiválasztásra kerül. A javasolt követelmények szisztematikus feldolgozásával előállítható a megvalósítandó IoT-rendszer egy kezdeti architektúraterve, amely figyelembe veszi a szakterület legkritikusabb követelményeit.

Okosváros-követelmény	Felhőarchitektúra-komponens
Rendszeren belüli és rendszerek közötti információ cserére való képesség	IoT-átjáró; felhőszolgáltató-hálózat; hálózatszéli szolgáltatások
Szabványalapú szenzorhálózat, kommunikációs protokollok	IoT-átjáró; IoT -transzformációs és -összekapcsolhatósági szolgáltatások; eszközmenedzsment
A rendszer távoli elérése és irányítása	IoT átjáró; hálózat-széli szolgáltatások; programozóifelület-menedzsment (API)
Képesség az energiahálózatból érkező adatok feldolgozására, elemzésére és döntések meghozására	Elemzési szolgáltatások; vizualizáció; vállalati alkalmazások
Különböző forrásból érkező adatok egyesítése	IoT-átjáró; IoT-transzformációs és -összekapcsolhatósági-szolgáltatások; hálózatszéli szolgáltatások
Időben történő adatszolgáltatás a beavatkozáshoz	IoT-átjáró; hálózatszéli szolgáltatások; elemzési szolgáltatások

2. táblázat: Követelmények leképezése architektúrakomponensekre

3. SZOLGÁLTATÁSOK TELJESÍTMÉNY-BENCHMARKOLÁSA

A benchmarkolás bevett gyakorlat, rendszerek vagy rendszerkomponensek teljesítményének számszerűsítésére és objektív összehasonlítására, előre definiált metrikák alapján. A szabványos benchmarkokon alapuló, validált és publikusan elérhetővé tett eredmények jelentős segítséget nyújtanak egy architektúra hardver- és szoftverkomponenseinek kiválasztásához anélkül, hogy saját költségen kipróbálnánk a lehetőségeket.

3.1. A benchmarkok tulajdonságai

Egy jó benchmark (Huppler, 2009) a következő tulajdonságokkal rendelkezik:

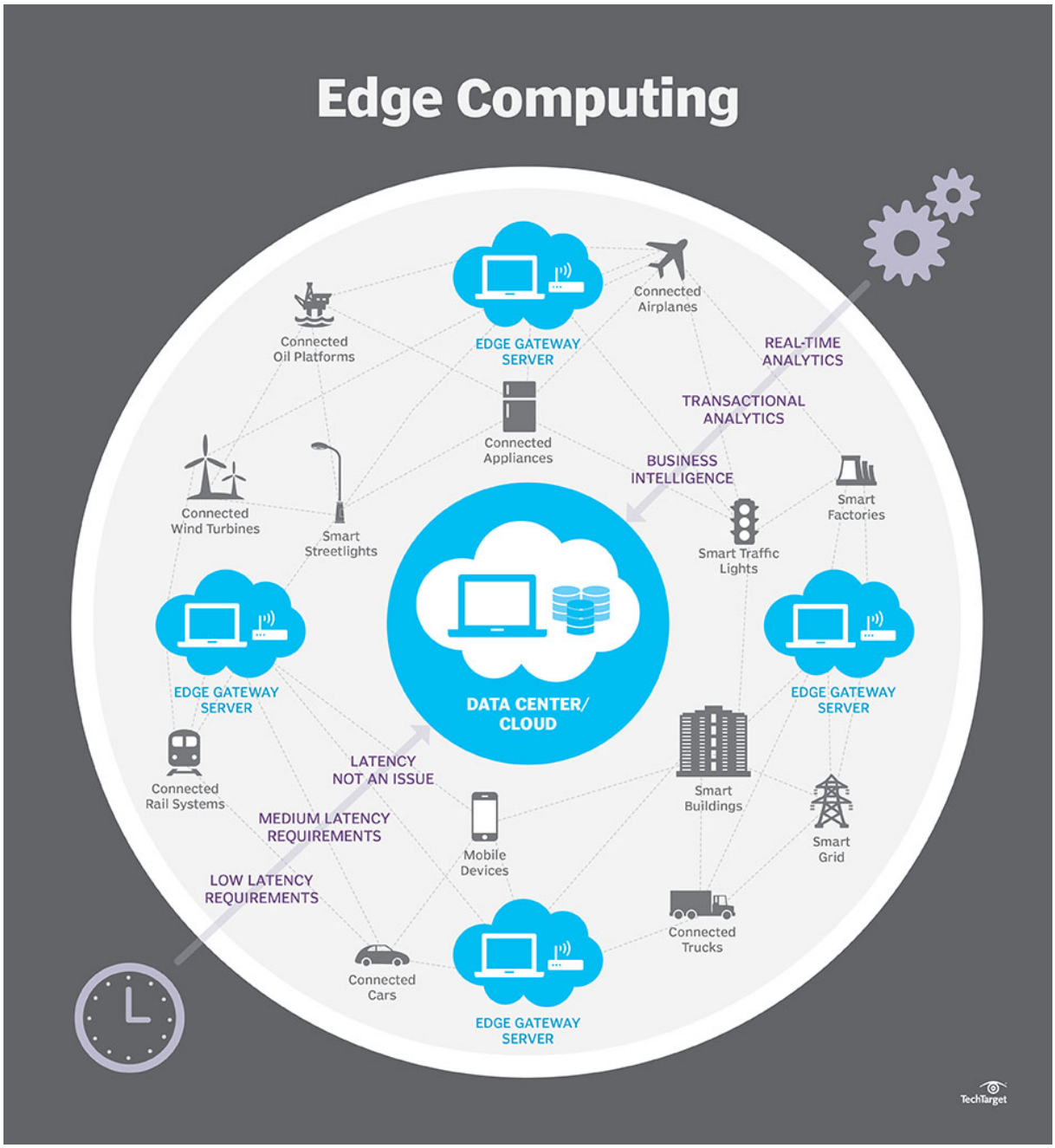
- Gyártófüggetlen, lehetővé téve különböző gyártóktól származó termékek összehasonlítását.
- Jól meghatározott célja van, azaz valós használati eseteket fed le.
- Megismételhető, ezáltal a tesztelés alatt álló rendszer teljesítménye determinisztikus módon felmérhető.
- Egyszerű, általános metrikákat definiál. Ez teszi lehetővé különböző módon megvalósított rendszerek összehasonlítását, egészen addig, amíg ugyanazt a használati esetet fedik le.
- Adaptív, nemcsak egyetlen esetre használható, hanem újrahasználható különböző környezetekben.
- Nyílt szabványokon alapszik, amely azt jelenti, hogy a benchmark publikusan elérhető, specifikációja és implementációja lektorált, szakmailag elbírált, valamint az ipar által összehasonlítási alapként elfogadott.
- Aktívan karbantartott, követi az új technológiai trendeket; szükség esetén frissített, alkalmazkodva ezáltal a tipikus használati esetek fejlődéséhez.

Ilyen tulajdonságokkal bíró benchmark specifikálása és implementálása időigényes, főleg egy olyan rohamosan fejlődő területen, mint az IoT-rendszerek. Annak ellenére, hogy az IoT koncepciója már évek óta létezik, és a legtöbb szervezet létrehozta a saját IoT-megoldását. Az első szabványos IoT-specifikus benchmark csak mostanában – néhány hónappal korábban a jelen mű írásához képest – került publikálásra.
A Transaction Processing Performance Council (TPC) 2017 júliusában jelentette meg az első IoT-rendszerekhez köthető benchmarkját, a TPCx-IoT-benchmarkot (TPC 2018a), amely a referenciaarchitektúra IoT-átjáró-komponensének teljesítményét hivatott felmérni.

Egy IoT-átjárónak több célja is van:

- Összeköti az eszközöket azáltal, hogy a különböző kommunikációs protokollok közötti különbségeket áthidalja.
- Adatokat tárol, ellenőriz, szűr és összesít, ezáltal csökkenti a limitált kapacitású eszközökre eső terhet.
- Adatokat szolgáltat a rendszer többi részének, például a vállalati analízis és döntéshozó komponenseknek.
- A modernebb átjárók már teljes értékű számítási platformként is funkcionálnak, saját operációs rendszerrel. Ez lehetővé teszi, hogy bizonyos irányítási funkciókat az eszközökhöz „közel” valósítsunk meg, növelve így a rendszer válaszidejét, ami különösen fontos kritikus események esetén.

A fenti tulajdonságoknak köszönhetően az IoT-átjárók fontos szerepet töltenek be az architektúrában. Ez a szerep indította el olyan szabványos benchmarkok létrehozásának folyamatát, amelyek az IoT-átjárók teljesítményét hivatottak felmérni. A TPCx-IoT az ipar első szabványos benchmarkja, amely dedikáltan egy IoT-rendszer komponensét célozza meg, jelezve ezzel a terület rohamos fejlődését és fontosságát.
Költséghatékonysági okokból az újonnan telepített IoT-rendszereknek alkalmazkodniuk kell a már meglévő alapinfrastruktúrákhoz. Például az aktuális, valós idejű energiafelhasználást monitorozó és irányító intelligens szolgáltatásoknak integrálniuk kell az előző évtized alatt telepített (okos) fogyasztásmérőket.
Az elsőgenerációs mérők még csak azt a célt szolgálták, hogy egyszerűbbé tegyék a havi leolvasásokat és a számlázási folyamatokat. A mérők következő generációja már részben lehetővé tette a szolgáltatóknak az energiaelosztó-hálózat irányítását. Néhány fogyasztói terhelés, például fűtő eszközök használata, tolerálja az időbeli eltolást a használatuk során (a felfűtés az energiaigényes rész, utána csak adott hőmérsékleten kell tartani a környezetet). Ebből kifolyólag a hasonló tulajdonságú terheléseket az olcsóbb éjszakai árammal elégítették ki, ezáltal kisimítva az energiafelhasználást a terheltebb időszakokban.
Ugyanakkor az IoT-rendszerekhez kapcsolt eszközök számának robbanásszerű növekedése rávilágított néhány problémára. Az eszközök heterogén természete miatt számtalan kommunikációs protokoll van jelen a rendszerben egyszerre. Ezt a problémát továbbmélyíti a szenzorok és beavatkozók limitált tárolási és számítási kapacitása, valamint a korlátozott energiafelhasználás-profiljuk. Mindez szükségessé tette egy újabb komponens, az IoT-átjáró bevonását a kihelyezett hálózatba. Ezáltal a kihelyezett hálózat jelentősebb szerepet kapott az architektúrában, félig szakítva az eddig elképzelt, tisztán felhőalapú okosszolgáltatások világával. Ilyen köztes megoldásokra is léteznek referenciaarchitektúrák, például „fog” és „edge” (4. ábra, (SearchDataCenter s.a)) számításhoz az OpenFog-referenciaarchitektúra (OpenFog Consortium 2017). Az előbbi megközelítés alapgondolata, hogy a rendszer egyes funkcióit különböző „távolságokra” valósítják meg a felhőszolgáltatótól, biztosítva ezzel például az adatfeldolgozás megfelelő sebességét. „Edge” számítás esetén a feldolgozás nagyon közel történik a szenzorokhoz, míg „fog” számítás esetén például a szenzorhálózat és a felhőszolgáltató hálózata „között” elhelyezett rendszer biztosítja az adatfeldolgozást.



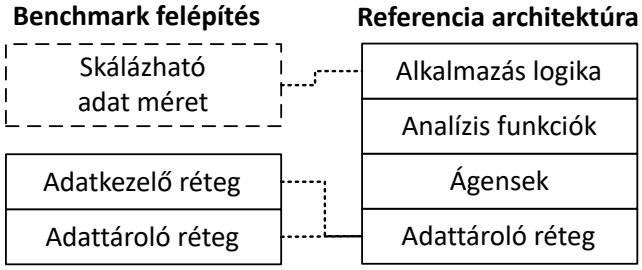
4. ábra: Edge computing. Forrás: SearchDataCenter s.a.

Egy adott környezet szabályozásán túl az IoT-rendszerek egyik fő célja, hogy értékes üzleti tudást származtasson adott környezeten végzett megfigyelések alapján. Annak ellenére, hogy az IoT-átjárók fontos szerepet töltenek be az adatkezelés terén, üzleti szintű analízis ritkán történik ebben a rétegben. Az üzleti tudás előállítása a felhőszolgáltató- és vállalati rétegekben történik. A jelenleg létező egyetlen IoT-átjáró-benchmark ezeket a rétegeket egyáltalán nem fedi le méréseivel. Továbbá jelenleg nincs olyan IoT-specifikus benchmark, amely a felhőszolgáltató- és vállalati rétegek szolgáltatásait célozná meg méréseivel.
Ezt a hiányt kiküszöbölhetjük, ha nem a konkrét felhasználási területre (IoT) összpontosítunk, hanem annak megoldandó problémáira. Az IoT-átjáróhoz hasonlóan a vállalati réteg feladata, hogy az összegyűjtött adatokon komplex elemzési lépések segítségével betekintést nyerjen a rendszer működésébe. Az IoT-rendszerek vállalati

szintű adattárolási és adatelemzési lépései olyan karakterisztikákat mutatnak, amelyek alapján a „Big Data”-problémák kategóriába sorolhatók (Cuzzocrea – Song – Davis, 2011). A „Big Data”-megoldások számtalan variációja fellelhető mind akadémiai, mind ipari területeken, bizonyítva a terület érettségét (Leskovec – Rajaraman – Ullman, 2014). Ennek megfelelően léteznek szabványos benchmarkok, amelyek a „Big Data”-megoldások objektív összehasonlítását teszik lehetővé.
A következő alfejezetek az IoT-átjáró és az adatelemzésért felelős komponensek teljesítménymérésére használható szabványos benchmarkokat mutatják be röviden.

3.2. A TPCx-IoT-benchmark

A TPCx-IoT-benchmark az átjáró adatbefogadási és adatlekérdezési képességeit méri fel. A teljesítménymérést egy realisztikus, elektromos erőművek szenzoraitól származó adathalmazzal végzi el, biztosítva ezáltal az eredmény relevanciáját. A tesztelendő rendszernek egy adatkezelő platformnak kell lennie, amely perzisztens módon tárolja az adatokat, biztosítva továbbá azok replikációját elérhetőségi okokból. A 4. ábra összehasonlítja a benchmark során vizsgált eszköz felépítését a felhőalapú- referenciaarchitektúra IoT-átjáró-komponensének felépítésével.



5. ábra: A benchmark és referencia architektúrák összehasonlítása.
Forrás: a szerző saját szerkesztése

A benchmark által definiált mérések a referenciaarchitektúrának kizárólag az adattároló képességeit fedik le. A további szolgáltatások mérésére nem biztosít lehetőséget, annak ellenére, hogy ezek jelentősen befolyásolhatják az adattároló réteghez érkező tényleges terhelést. Ugyanakkor a kimaradt szolgáltatások hatását szimulálhatjuk azzal, hogy módosítjuk a benchmark által használt adatkészlet méretét. Például az adatkészlet méretének csökkentésével azt a hatást érhetjük el, mintha az átjáró alkalmazás logikája szűrné, illetve aggregálná az adatokat, csökkentve ezáltal az adattároló rétegre jutó terhelést.
A benchmark általános teljesítmény- és ár-teljesítmény-metrikákat határoz meg. A teljesítménymetrika az eszköz tényleges áteresztőképességének felel meg, és a másodpercenként elvégzett adatműveletek (írás vagy olvasás) számaként van definiálva. Az ár-teljesítmény-metrika az egy műveletre eső átlagos költséget adja meg, lehetővé téve a költség szerinti optimalizálást is az eszközök kiválasztásakor.

3.3. A TPCx-BB-benchmark

A szabványos TPCx-BB-benchmark (TPC 2016) [melynek alapjául (Ghazal et al., 2013) szolgált] egy alkalmazásszintű benchmark, amely egy fizikai és egy online bolttal rendelkező kereskedő működését modellezi. Az angolul megfogalmazott 30 komplex lekérdezés olyan használati eseteket fed le, mint például raktárkészlet- menedzsment, jelentéskészítés, árazásoptimalizálás. A lekérdezések strukturált, félig strukturált és

strukturálatlan adatokon is dolgoznak, tehát általánosabb az adatkészlet, mint ami IoT-rendszerek esetén fellelhető.
A benchmark több mérési fázist is tartalmaz, amely során a megadott méretű adathalmazt elhelyezi a mért platformon, lefuttatja egymás után az összes lekérdezést, majd párhuzamosan intéz lekérdezéseket a rendszer felé. A TPCx-IoT-benchmarkhoz hasonló teljesítmény- és ár-teljesítmény-metrikákat határoz meg az alapján, hogy mennyi kérést (például adatlekérdezést) képes a rendszer percenként kiszolgálni, és ez mekkora (például üzemeltetési) költséget jelent.

3.4. A TPCx-HS-benchmark

A TPCx-HS-benchmark (TPC 2018b) célja olyan rendszerek teljesítményének felmérése, amelyek masszívan elosztott adattárolást és adatelérést tesznek lehetővé. Ilyenek például az Apache Hadoop fájlrendszeralapú rendszerek (Apache s.a.). A benchmark több mérési fázis során méri a tesztelt rendszer adatbefogadó képességét, illetve az adatelérhetőség minőségére nyújtott garanciáit. Továbbá az adatmanipulációs teljesítményt is felméri egy sorrendezési algoritmus futtatásával (és az eredmény ellenőrzésével). Az előző benchmarkokhoz hasonlóan a TPCx-HS is teljesítmény- és ár-teljesítmény-metrikákat definiál.
Annak ellenére, hogy a bemutatott „Big Data”-benchmarkok szabványosak, az IoT-rendszerekben betöltött szerepük és alkalmazhatóságuk további megfontolásokat igényel. Mindkét „Big Data”-benchmark tartalmazza a vállalati rétegben lévő rendszerkomponensek adatgyűjtési és adattárolási felmérését. Ugyanakkor ezen feladatok leginkább az IoT-átjárók hatáskörébe tartoznak. Továbbá a „Big Data”-benchmarkok az IoT-rendszerekhez képest sokkal általánosabb felépítésű adatokon dolgoznak. Tehát hiába szolgáltatnak objektív döntési szempontokat a rendszerkomponensek kiválasztásához, a kapott eredményeket semmiképp sem szabad teljesítményre adott garanciaként kezelni! A rendszer megfelelő teljesítményre méretezéséhez dedikált teljesítményhangolásra van szükség, ami alatt a kritikus komponensek felderítését és teljesítményük javítását értjük.

4. KOMPLEX RENDSZEREK TELJESÍTMÉNYELEMZÉSE

A 2. fejezetben bemutatott ajánlások, tervezési módszerek és referenciaarchitektúrák egy egységes kiindulópontként szolgálnak az okosszolgáltatások infrastrukturális hátterét biztosító, komplex IT-rendszerek megtervezéséhez és kiépítéséhez.
A szolgáltatásokkal szemben támasztott funkcionális követelmények mentén kiválaszthatók a működéshez szükséges architektúrakomponensek, megtervezhetők a közöttük felépítendő (kommunikációs) kapcsolatok, illetve telepíthetők a végfelhasználók (legyen szó akár a lakosokról, az üzleti döntéshozókról vagy a rendszer karbantartóiról) számára elkészített alkalmazások. Ezzel szemben az extrafunkcionális követelményeknek (például a teljesítménykritériumoknak) való megfelelés biztosítása jellemzően komplexebb probléma, nagyobb odafigyelést igényel.
Mielőtt részletesen bemutatnánk a teljesítményre tervezés és teljesítményelemzés módszereit, először, a következő alfejezetben áttekintjük a funkcionális követelmények ellenőrzésének alapgondolatait, hogy a későbbiekben érzékeltetni tudjuk a két feladat, a tervezés és az ellenőrzés közötti alapvető különbséget.

4.1. A megfelelőség ellenőrzése

A pontos, helyes és ellentmondásmentes követelmények minden rendszer megkerülhetetlen alapját képezik. Egy rendszer fejlesztésére számtalan módszertan létezik, azonban minden esetben a fejlesztés lépéseit és fázisainak határait a megvalósítandó követelmények vezérlik. A következő alfejezetek a követelmények két jellegzetes típusát mutatják be: funkcionális és extrafunkcionális (vagy más néven nemfunkcionális) követelmények.

4.1.1. Funkcionális követelmények ellenőrzése

A funkcionális követelményeknek való megfelelőség a rendszer fejlesztésének különböző életciklusaiban (mint például a tervezési, implementációs és telepítés fázisokban) folyamatosan validálható és verifikálható. Validálás során az elkészült rendszer minden részének/komponensének (egy adatbázissémától egészen egy kódrészletig az implementációs fázisban) visszakövethetőnek kell lennie egy adott funkcionális követelményhez és fordítva (Post et al., 2009). Ez a követhetőség (traceability) az alapja annak, hogy eldönthető legyen, a rendszer valóban a kívánt funkciókat (és csak azokat) valósítja-e meg.
A validált rendszer ellenőrzésének következő lépése a verifikáció, amely során a funkciók helyes megvalósításának a bizonyítása a cél. A bizonyítás „alapossága” a rendszer felhasználási területétől függ, és általában szoros kapcsolatban áll a rendszer kritikusságával. Míg egy közösségi oldal esetén a hibák legfeljebb a felhasználói elégedettségre vannak hatással, addig egy energiaszolgáltatót működtető infrastruktúrában bekövetkező hibának súlyos következményei lehetnek, mint például emberélet veszélyeztetése. Érthető, hogy egy rendszer helyes működését a két

felhasználási területen élesen különböző alapossággal szükséges verifikálni, amelynek módszereire – kritikus rendszerek esetén – szigorú előírások vannak.
Egy komplex rendszer fejlesztése során jellemzően már létező, kereskedelmileg kapható (úgynevezett commercial off-the-shelf, COTS) komponensekből/szolgáltatásokból építkezünk, amennyiben ez lehetséges. Ez alól kivétel lehet, ha a rendszerben speciális célhardvert, vagy saját, innovatív módszereket alkalmazunk.
A fejlesztési költség esetleges csökkentésén kívül egy másik fontos célja is van a már létező komponensek újrafelhasználásának. Ezt legjobban a kriptográfiában használatos algoritmusokat megvalósító szoftvermodulok újrafelhasználása szemlélteti. Kimondottan ellenjavalt egy ilyen modul saját kezű implementációja ahelyett, hogy egy széles körben elterjedt és nyílt, már elkészített implementációt használnánk. Egy kriptográfiai algoritmus formálisan bizonyított elméleti helyessége mellett kritikus annak helyes, verifikált implementációja. Azonban a szoftverek formális helyességbizonyítása matematikailag nehéz, erőforrásigényes; a gyakorlatban legtöbbször nem kivitelezhető folyamat. Ugyanakkor teszteléssel önmagában nem garantálható az implementáció hibamentessége. A probléma akkor a legsúlyosabb, ha egy implementációs hibát nem a fejlesztő csapat, hanem egy rosszindulatú támadó talál meg, akinek nem áll érdekében felfedni a hibát. Ennek az előfordulási valószínűségét csökkenthetjük, ha széles körben használatos, lehetőleg nyílt forrású implementációkat használunk fel újra. Ilyenkor ugyanis egy kiterjedt fejlesztői közösségnek a feladata – és, ami a legfontosabb, érdeke – az implementáció helyességének garantálása, és az esetleges megtalált hibák azonnali jelzése.
Megjegyzendő, hogy ez a megközelítés sem biztosít 100%-os védelmet a hibák ellen. Erre jó példa a biztonságos internetes kommunikációt lehetővé tévő SSL/TLS-protokoll egyik nyílt implementációja, az OpenSSL-szoftverkönyvtár, amelybe 2012-ben bekerült egy implementációs hiba, amelynek segítségével ki lehetett nyerni a kliens/szerver memóriatartalmának egy korlátozott részét (ami adott esetben érzékeny információkat is tartalmazhat). A hibára (publikusan) csak 2014-ben derült fény. Mivel egy nyílt és széles körben használatos szoftverkönyvtárról van szó, a feltárt hibát számtalan fórum publikálta, biztosítva ezáltal, hogy a felhasználói bázis lehető legnagyobb része értesüljön az implementáció sérülékenységéről, és minél hamarabb megtehesse a szükséges óvintézkedéseket. A hibát még aznap kijavították az OpenSSL-szoftverkönyvtárban.

Az OpenSSL remek példa arra, hogy milyen előnyökkel jár egy széles körben használatos komponens újrafelhasználása:

- A komponens már rendelkezésre áll, így a fejlesztése nem, csak az integrálása igényel a fejlesztők részéről időráfordítást.
- A komponens jól definiált funkcionális követelményeket teljesít (például adat titkosítása valamilyen algoritmus segítségével), amelyek közvetlenül vagy közvetve, de felhasználhatók a rendszer saját funkcionális követelményeinek (részleges) kielégítésére.
- A komponens széles körű felhasználása jellemzően együtt jár az alaposan tesztelt helyes működéssel, amely nem elhanyagolható költséget jelentene egy saját fejlesztésű komponens esetén.
- Szintén a széles körű felhasználásnak köszönhető, hogy az esetleges hibák nagyobb valószínűséggel és gyorsabban kerülnek felderítésre, gyorsítva ezáltal egy komponens „stabilitásának” növekedését.

Az „oszd meg és uralkodj”-elvet alkalmazva a COTS-komponensek felhasználásával jelentősen hatékonyabbá tehető egy olyan komplex IT-rendszer elkészítése, amely megfelel adott funkcionális követelményeknek.

4.1.2. Extrafunkcionális követelmények ellenőrzése

Manapság azonban a funkcionális követelmények mellett ugyanolyan fontos megfelelni bizonyos extrafunkcionális követelményeknek is. Visszatérve a közösségi oldalak példájára – ami szigorú értelemben véve nem tartozik a kritikus infrastruktúrák közé – igenis fontos szempont a felhasználóknál, hogy egy ilyen szolgáltatás az idő legnagyobb részében elérhető legyen, valamint gyorsan kiszolgálja a felhasználói igényeket. A felhasználói elégedetlenség a szolgáltatástól való elpártolást eredményezheti, ami az online reklámok korában jelentős bevételkiesést okozhat a szolgáltatónak [2017-ben a Facebook összbevételének 98%-át, közel 40 milliárd dollárt, a hirdetések adták (The Statistics Portal, 2017)].
A funkcionális és extrafunkcionális (vagy nemfunkcionális) követelmények közötti különbség szemléltetésére a leggyakrabban használt analógia az, hogy míg előbbi arra a kérdésre ad választ, hogy *mit* kell megvalósítania a rendszernek, addig utóbbi azt írja le, hogy ezt *hogyan* valósítsa meg a rendszer. Tipikus példái ennek a gépjárművel szemben támasztott követelmények: funkcionális követelmény, hogy a vezető képes legyen (két pont között) a helyzetét változtatni a járművel; extrafunkcionális követelmény például, hogy a vezető biztonságos módon tudja megtenni ezt az utat, a távolság függvényében megadott időn belül.

A valóság ennél azonban árnyaltabb, az extrafunkcionális követelmények igen sokrétűek lehetnek (Chung – do Prado Leite, 2009). Ebbe a kategóriába tartoznak például (a teljesség igénye nélkül) az alábbi szoftverekhez kapcsolódó általános követelménytípusok:

- használhatóság (usability);
- módosíthatóság (modifiability);
- megbízhatóság (reliability);
- skálázhatóság (scalability);
- konfigurálhatóság (configurability);
- robusztusság (robustness);
- teljesítmény (performance);
- stb.

A teljes követelményspektrum tárgyalása túlmutat a kismonográfia keretein.
A fejezet hátralévő részében egy iteratív és rekurzív teljesítményfelmérési és teljesítményhangolási módszertant mutatunk be, amely szisztematikus módon tárja fel és hárítja el a rendszer teljesítményproblémáit, valamint a származtatott eredmények alapján prediktív modelleket állít fel a rendszer teljesítményének jóslásához.

4.2. Szisztematikus teljesítményelemzés

Az 4.1 alfejezetben bemutatott „oszd meg és uralkodj”-elv alkalmazása az extrafunkcionális követelmények kielégítésére nem annyira magától értetődő, mint a funkcionális követelmények esetén. Vegyünk példaként egy okosmérőórákat használó energiaszolgáltató infrastruktúráját. Funkcionális követelmény lehet a rendszerrel szemben, hogy az időnként mintavételezett óraállásokat a rendszer egy megadott strukturált formában

eltárolja egy adatbázisban, amelyből azok később visszaolvashatók/lekérdezhetők. Számtalan adatbáziskezelő rendszer áll rendelkezésre, amely eleget tud tenni ennek a követelménynek.

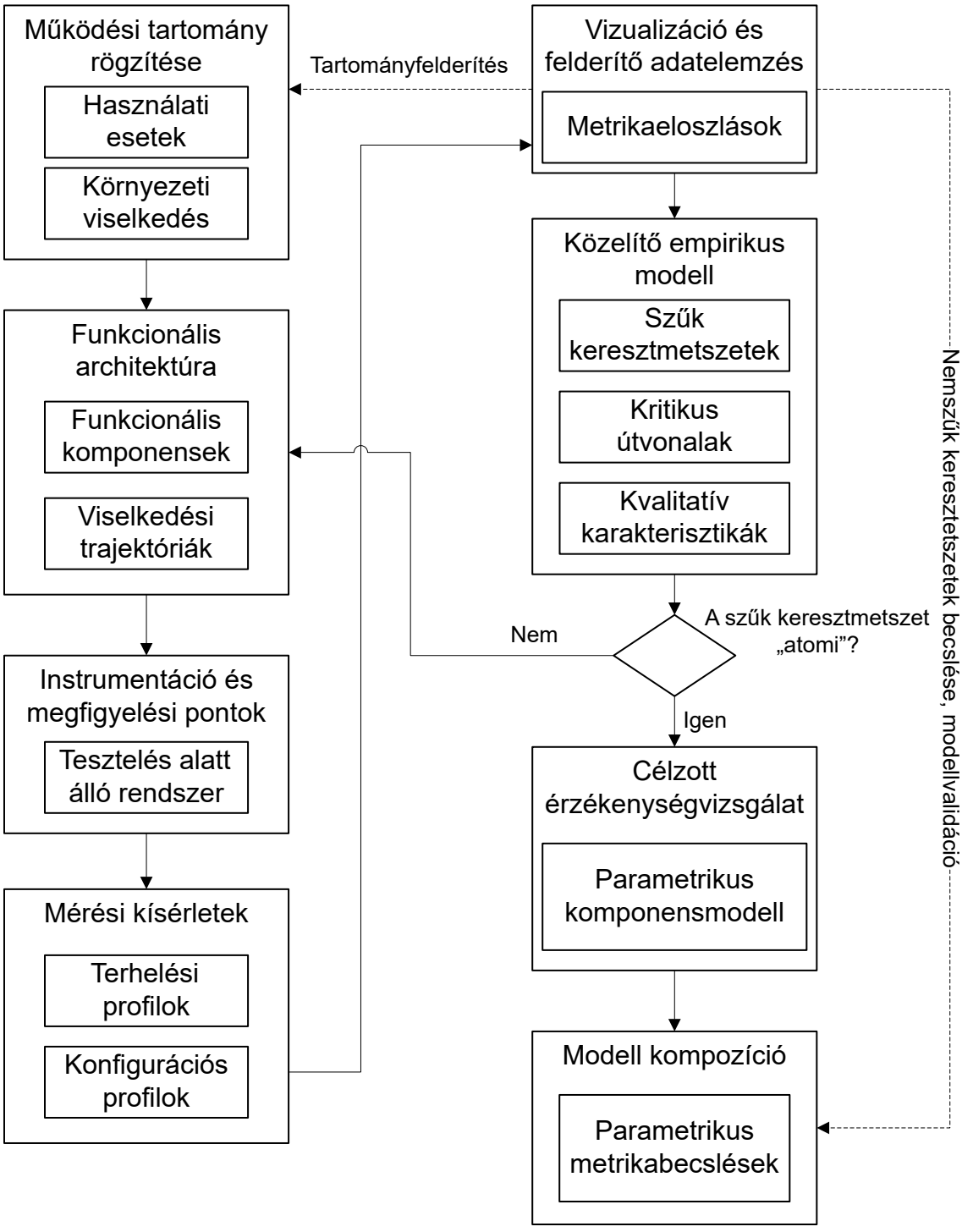
Tegyük fel, hogy további követelmény a rendszerrel szemben, hogy a fogyasztók, az energiahálózat karbantartói, és a stratégiai döntéseket hozó üzletemberek/befektetők is nyomon tudják követni az energiafelhasználás alakulását a hálózatban az őket érintő szinten. Tehát tudni kell adatokat szolgáltatni egy adott háztartáshoz kapcsolódóan, az energiahálózat egy adott régiójára vonatkozóan, valamint általánosságban az energiafelhasználási trendekről. Érezhető, hogy habár a komponens funkcionálisan képes ezen kérdések megválaszolására (hiszen a válaszhoz szükséges adatok rendelkezésre állnak), mégis figyelembe kell vennünk, hogy a különböző kérdések megválaszolása különböző mértékű (akár egyidejű) terhelést jelenthet ugyanazon komponensre.

Tovább bonyolódik a helyzet, ha bizonyos kérdéseket korlátozott időn belül kell megválaszolni. Például az üzemeltetők folyamatosan, közel valós időben (akár másodperces pontossággal) szeretnének friss képet kapni az energiahálózat kihasználtságáról, hogy gyorsan tudjanak reagálni az energiafelhasználási igények megváltozására, például új generátorok hálózatba kapcsolásával (National Grid UK s. a.). Ilyen esetekben a funkcionális és teljesítménykövetelmények együttes figyelembevétele szükséges annak érdekében, hogy a rendszer megfelelően üzemeljen.

Látható, hogy egy komplex rendszer adott teljesítményre méretezését számtalan tényező nehezíti, többek között a megvalósított funkciók komplex egymásra hatása, valamint az időnként megváltozó környezet (például új régió hozzácsatolása az energia hálózathoz), amely a rendszer terhelését befolyásolja. Emiatt nélkülözhetetlen a rendszer teljesítményének alapos, szisztematikus felmérése különböző terhelések (környezeti behatások) mellett (Banga – Druschel, 1999).

A mai, nagy kiterjedésű elosztott rendszerek komplexitása ellehetetleníti a közvetlen, analitikus teljesítménymodellek felállítását. Tehát például nem tudjuk egyszerűen egy zárt képlet segítségével előre megadni a rendszer válaszidejét a párhuzamosan beérkező kérések függvényében. Kompromisszumot kell kötni a teljesítménymodellek pontossága (komplexitása) és felhasználhatósága között.

Az 5. ábrán szemléltetett folyamat célja, hogy szisztematikus módon feltárja a teljesítmény szempontjából kritikus komponenseket, és ez alapján segítsen azok megfelelő konfigurálásában, valamint a későbbi viselkedés előrejelzésében.



6. ábra: Szisztematikus teljesítményfelmérés folyamat.
Forrás: a szerző saját szerkesztése

Fontos megjegyezni, hogy a folyamat iteratív, és minden iterációban egy részrendszer megfigyelhető komponenseinek a teljesítményét térképezi fel és azonosítja a szűk keresztmetszetet. Az első iteráció során ez a részrendszer maga a teljes rendszer és az azt alkotó magas szintű komponensek; a második iteráció során a részrendszer az első iterációban szűk keresztmetszetként azonosított komponens és annak alkomponensei;

és így tovább. Ez a felülről lefelé irányú (*top-down*) finomítás adja a folyamat rekurzív jellegét, és fókuszálja az elemzést a kritikus komponensekre.
A további szakaszokban a folyamat egyes lépései kerülnek részletes bemutatásra.

4.2.1. Működési tartomány rögzítése

A metodológia célja komplex rendszerek teljesítményfelmérésének minél egyszerűbbé tétele. Ennek egyik alappillére a rendszer különböző működési tartományainak (*operating envelope*) rögzítése a rendszeranalízis megkezdése előtt.
Egy kritikus rendszerrel szemben támaszthatók olyan extrafunkcionális követelmények, amelyek valamilyen módon kapcsolódnak, vagy kihatnak a rendszer teljesítményére. Tekintsük például azt a követelményt, miszerint a rendszernek bizonyos erőforráshibák esetén, úgynevezett degradált állapotban is nyújtania kell egy bizonyos minőségű alapszolgáltatást (Strategy&, 2017). Fontos példák erre a telekommunikációs hálózatok, amelyeknek komolyabb vészhelyzetek (például természeti katasztrófák) esetén is biztosítaniuk kell egy alapszintű szolgáltatást, vagy akár egy dedikált hálózatot a katasztrófaelhárítási egységek számára a hatékonyabb koordináció érdekében.
De elég egy egyszerű webáruházat működtető infrastruktúrára gondolni, aminek hirtelen terhelésnövekedéssel kell megbirkóznia, például leárazások vagy ünnepek alkalmával. Jelentős bevételkiesést jelentene az üzemeltetőnek, ha a hirtelen többletterhelés miatt összeomlana a rendszer, és ezáltal már a megszokott felhasználói forgalmat sem tudná kiszolgálni.

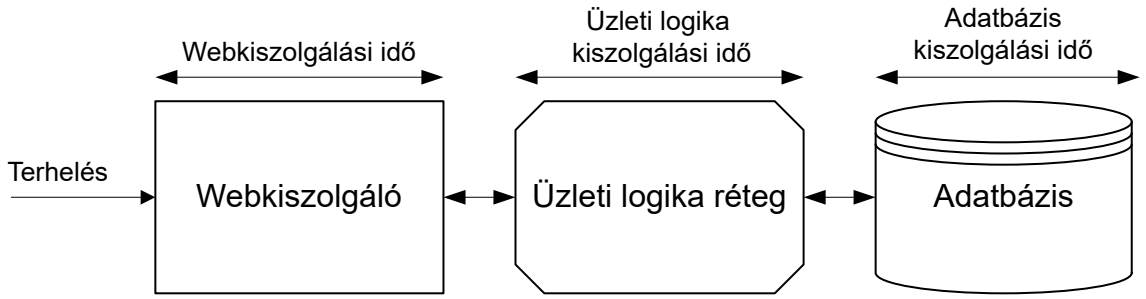
Ha az összes ilyen teljesítményhez kapcsolódó követelménynek való megfelelést egyszerre szeretnénk vizsgálni, az jelentősen elbonyolítaná, átláthatatlanná tenné a rendszer teljesítményelemzését. Ebből kifolyólag az ilyen esetekben mutatott rendszerviselkedéseket, működési tartományokra bontva, külön elemezzük. A működési tartomány tehát nem más, mint a rendszer környezetének olyan nemű korlátozása, amely a rendszert egy kívánt állapotban tartja a teljesítményelemzés alatt. Ezek alapján rendszerünkhöz például a következő működési tartományokat definiálhatjuk:

- hibamentes, hosszú távú, normál terhelés alatti működés;
- hibamentes, tartósan magas terhelés alatti működés (de nincs túlterhelés);
- túlterhelés alatti működés;
- korlátozott számú erőforrásokkal történő működés;
- hibás/degradálódott komponenssel történő működés;
- stb…

A működési tartomány mellett az elemzés során használt rendszerfunkciók is rögzítésre kerülnek mint használati esetek. Ezek jelölik ki a következő lépésben használt (ezáltal figyelembe veendő) komponenseket, amik a további iterációk alapját képezik. Emellett az elemzés fókuszálásában is segítenek, hiszen a működés során nem érintett komponensekkel nem kell foglalkoznunk.
Például, ha egy okos-energiahálózatot működtető rendszer azon képességét vizsgáljuk, hogy milyen sebességgel képes aggregált, historikus adatok előállítására (például adott városok energiahálózatának kihasználtsága az előző órában adott tartományokra összesítve), akkor ez a funkció nem fogja érinteni a végfelhasználókhoz közel elhelyezett IoT-átjárókat, hiszen azokat leginkább a valós idejű események feldolgozására használjuk, mintsem historikus adatok előállítására.

4.2.2. Funkcionális architektúra

Miután behatároltunk egy működési tartományt és a kapcsolódó használati eseteket, a következő lépésben el kell készítenünk a rendszer funkcionális architektúrájának egy közelítő modelljét. Ez a modell a használati esetek által érintett komponenseket és a közöttük lévő kapcsolatokat fogja tartalmazni. Ezek a komponensek képezik a következő lépésekben eszközölt megfigyelések és mérési kísérletek alapjait. Az elemzési folyamat első iterációjában ez az architektúramodell a rendszer főbb, magas szintű komponenseit és azok kapcsolatait fogja tartalmazni.
Egy online bolt esetén például a 6. ábra szemléltet egy lehetséges modellezési felbontást. A terhelést képező felhasználói kérések (amelyek pontos típusa a definiált használati esetektől függ) a feldolgozásuk során a rendszer három fő komponensén haladnak keresztül: a *webkiszolgálón* (amely például a beérkező http-kéréseket kezeli); az üzleti logika rétegen (ahol például előállítjuk a vevőnek ajánlott termékek listáját); valamint az *adatbázis rétegen* (ahol az üzleti logika futtatásához szükséges adatokat tároljuk).



7. ábra: Online bolt magas szintű funkcionális architektúrája.
Forrás: a szerző saját szerkesztése

Fontos megjegyezni, hogy követhetőbbé tehetjük a folyamatot, ha például valamilyen (fél)formális modellezési nyelvet alkalmazunk a funkcionális architektúra leírására, mint például az UML-t (Unified Modelling Language, amely a rendszer különböző nézetinek magas szintű leírását teszi lehetővé változatos diagramok formájában). Ebben az esetben lehetőségünk nyílik a különböző iterációk során elkészített modellek közötti konzisztencia ellenőrzésére is, így megbizonyosodhatunk arról, hogy a felmérés rekurzív, finomítási lépése során (lásd az 4.2.7 szakaszt) nem követtünk-e el valamilyen hibát.

4.2.3. Instrumentáció és megfigyelési pontok

A teljesítményelemzés következő lépéseként, és a mérési kísérletek közvetlen megalapozásaként azonosítanunk kell a rendszer azon aspektusait, amelyet a későbbiekben meg szeretnénk figyelni. A különböző teljesítménymetrikák (például átbocsátás, maximális átbocsátás, végrehajtási idők, kiszolgálási idő) származtatásának alapja a rendszerben zajló folyamatok (például egy kérés végrehajtásának) megfigyelése. Ez alatt jellemzően azt értjük, hogy a végrehajtás kitüntetett pontjaiban feljegyezzük, hogy mennyi idő telt el a folyamat kezdete óta.
Az online bolt példa esetén ez azt jelenti, hogy rögzítjük a kérés beérkezésének az idejét, majd az infrastruktúra egyes rétegeinek a határánál (tehát például amikor a kérés továbbításra kerül a webkiszolgáló rétegből az üzleti logika réteg felé) ismét rögzítjük

az aktuális időt, amiből később kiszámolható többek között a kérés által az egyes rétegekben eltöltött idő, de akár a rétegek „között töltött” idő is, ami a kommunikációs késleltetésnek felel meg. A felhőszolgáltatások esetén (amikor az egyes rétegek már nem egy lokális hálózaton helyezkednek el) a kommunikációs késleltetés sokszor összemérhető az „értelmes” munkával, a kiszolgálással töltött idővel, tehát egyáltalán nem elhanyagolható.

A példa jól szemlélteti, hogy egy kellően jól strukturált funkcionális architektúramodell esetén az érdekes/kritikus megfigyelési pontok többnyire a komponensek határaira fognak esni. Azonban ez önmagában még nem biztosítja a pontos megfigyelhetőséget. Manapság a használatra kész szolgáltatásokat megvalósító, harmadik féltől származó (úgynevezett *third-party*) komponensek a legtöbb esetben képesek saját működésük részletes naplózására. Ebben az esetben egy minimális naplózási konfigurációval rendelkezésünkre fognak állni futás közben az egyes kérések fázisaihoz tartozó időbélyegek.

Ugyanakkor a naplózási funkciók használata körültekintést igényel. Egy maximális átbocsátásra optimalizált komponens esetén az érdekes események naplófájlba írásának ideje a lassabb elérésű merevlemez miatt összemérhető lehet az elvégzett „hasznos” munkáéval. Ennek a teljesítményhatásnak a minimalizálása nem triviális feladat (Chen 2009, Fang et al., 2011; Ghosh – Bhatt – Vaitheeswaran, 2004). Ezenkívül a komponensek ritkán biztosítanak konfigurációs lehetőséget arra, hogy csak bizonyos részfeladatuk (például kérés fogadás/küldés) eseményeit naplózzák. A legtöbb naplózási mechanizmus csak súlyossági szint szerint szűr, tehát megszabhatjuk, hogy például csak a *figyelmeztetés* és/vagy a súlyosabb jellegű eseményeket naplózza. Azonban ez az egész komponensre globálisan vonatkozik.

A komponensek naplózási mechanizmusának alternatívája lehet, ha a komponenseket összekötő kommunikációs csatornákat és a rajtuk keresztül haladó kéréseket figyeljük meg (Dabir – Matrawy, 2007). A harmadik felek által készített, közfelhasználásra szánt szolgáltatások szinte kivétel nélkül kommunikációra valamilyen szabványos protokollt használnak, ami lehetővé teszi, hogy a kommunikáció erre felkészített általános célú eszközökkel megfigyelhető legyen.

Egy harmadik alternatíva a kérések feldolgozási idejének rögzítésére a kódinstrumentáció (Geimer et al., 2009). Ez alatt azt értjük, hogy a komponens kódjába saját utasításokat helyezünk el, jellemzően naplózási céllal – de kódinstrumentáció számos más célra is használható, lásd az aspektus orientált programozás céljait (Kiczales et al., 1997). Az előző két módszerrel szemben ez egy intruzív megoldás, hiszen módosítjuk a komponens programkódját. Ez a tulajdonsága adja a megoldás legnagyobb hátrányát, hiszen ez a módosítás sok esetben nem tehető meg, például kommerciális, zárt forráskódú komponensek esetén. Még ha a komponens nyílt forráskódú is, az instrumentációhoz szükséges a forráskód megfelelő mélységű ismerete, ami miatt komplex komponensek instrumentációja időigényes folyamat lehet. Ugyanakkor, ha mégis lehetséges, akkor egy erős eszköz áll rendelkezésünkre ahhoz, hogy a komponens teljesítményét a lehető legkevésbé befolyásolva (például a merevlemeznél sokkal gyorsabb elérésű memóriába naplózva) információt gyűjtsünk a kérések kiszolgálási idejéről.

Az utolsó átgondolandó lépés ebben a fázisban, hogy a mérés során összegyűjtött adatokat miként fogjuk begyűjteni a különböző komponensek telepítési helyéről, amelyek akár különböző felhőszolgáltatók által biztosított virtuális gépekben helyezkedhetnek el. Szerencsére az adatok (akár automatikus) begyűjtése a funkciógazdag menedzsmentfelületeknek köszönhetően ritkán okoz problémát, így ezzel a továbbiakban nem foglalkozunk.

4.2.4. Mérési kísérletek

Az előző lépések során elvégzett előkészületek után már elvégezhetők a mérési kísérletek. A többes szám (kísérlet*ek*) használatát az indokolja, hogy nem csak egyetlen „helyzetben” mérjük meg a rendszer/komponens teljesítményét, hiszen a végső célunk, hogy a rendszer teljesítménykarakterisztikáját határozzuk meg, és esetleg előre jelezzük, hogyan fog reagálni a változó környezeti terhelésre vagy működési körülményekre. Ennek megfelelően a rendszert több konfigurációban és több terhelési profil hatása alatt figyeljük meg. A mérések során a következő, közel független aspektusokat kell figyelembe venni a rendszerrel kapcsolatban:

- *A terhelés intenzitása*, figyelembe véve az első lépés meghatározott működési tartományt. Azaz ne terheljük túl a rendszert, ha a működési tartományként normál állapotban való működést definiáltunk. Ugyanakkor a megengedett terhelési tartományon belül több intenzitás értékre (100 kérés/mp, 1000 kérés/mp, …) is figyeljük meg a rendszer teljesítményét.
- *A rendelkezésre álló erőforrások*. A legtöbb mai online szolgáltatással szemben elvárás annak valamilyen mértékű skálázhatósága. Ez alatt azt értjük, hogy a rendszer teljesítménye növelhető legyen azáltal, hogy több vagy erősebb számítási erőforrást bocsátunk a rendszer rendelkezésére. Előbbi esetben horizontális, utóbbi esetben vertikális skálázásról beszélünk.

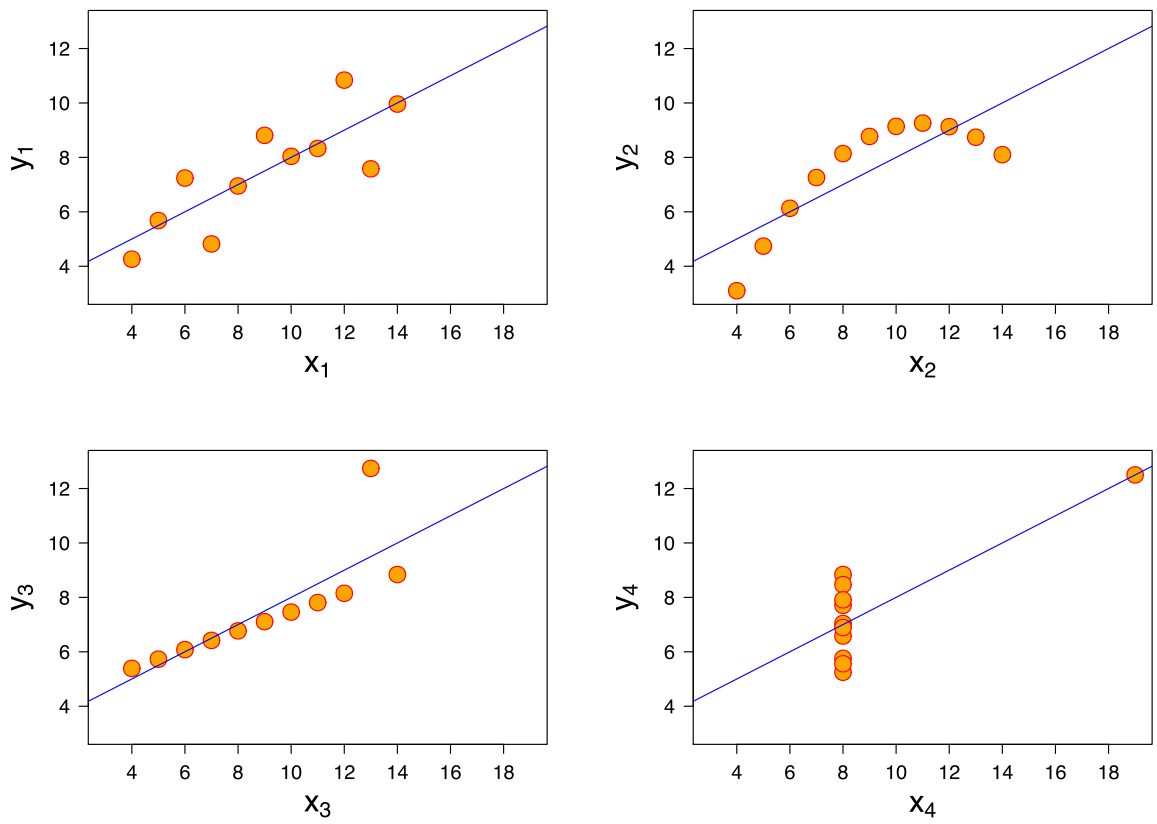
  A horizontális skálázhatóság mint tulajdonság előnye, hogy sokkal könnyebb erőforrásokat többszörözni (például CPU-darabszám növelése), mint egy erőforrás számítási kapacitását növelni (például CPU-magok számának, vagy a magok sebességének növelésével). Az utóbbi, vertikális esetben előbb-utóbb fizikai korlátokba ütközünk. Azonban sokkal könnyebb úgy helyesen implementálni egy szolgáltatást, hogy csak egyetlen CPU-n fut. Ezzel szemben a horizontális skálázhatóság megköveteli a szolgáltatás *jól* párhuzamosított implementációját, ami számtalan kihívást rejthet magában.
- *A rendszer/komponens konfigurációja*. A legtöbb, harmadik fél által implementált szolgáltatás mögötti motiváció az, hogy azt minél többen felhasználhassák. Emiatt az implementáció legtöbbször kellően általános, azaz felhasználás során kellő rugalmassággal testreszabható. Komplex szolgáltatások esetén a konfigurációs pontok száma a százas nagyságrendet is megközelítheti. A komponens beható ismerete nélkül megtalálni az adott terhelést leghatékonyabban kiszolgáló konfigurációt általában reménytelen. Az Oracle például az adatbázis rendszereinek árusítása mellett külön szolgáltatásként kezeli azok megfelelő konfigurálását és karbantartását (Oracle s.a.). A mérésikísérlet-lépés azonban megfelelő arra, hogy szisztematikusan végigpróbáljuk a lehetséges konfigurációkat az egyes terhelés profilokhoz, és kiválasszuk a legmegfelelőbbet.

Az egyes aspektusok értékkészletei önmagukban is széles skálán mozoghatnak. Az aspektusok együttes, teljes kombinatorikus kiértékelése költséges és időigényes folyamat, még akkor is, ha a különböző kiértékelések triviálisan párhuzamosíthatók. Azonban korábbi tapasztalatokra és ajánlásokra alapozva a felmérendő kombinációk száma kezelhető méretekben tartható. Hiszen például egy intenzív terhelés esetén nem érdemes kipróbálni az erőforrásszegény környezeti konfigurációt (hacsak nem ezt írja elő a működési tartomány). Léteznek továbbá paraméteroptimalizációs módszerek

(Michalewicz – Schoenauer, 1996, Akay – Kraboga, 2012), amelyek a kombinatorikus (*grid*) keresésnél intelligensebb módon próbálják megkeresni az optimális konfigurációt.

4.2.5. Vizualizáció és felderítő adatelemzés

Ennél a pontnál már rengeteg információ áll rendelkezésünkre a rendszer teljesítményéről, különböző konfigurációk és terhelések esetén. A metodológiának talán a legfontosabb alappillére a felderítő vizuális adatelemzés (Exploratory Data Analysis, EDA). Az EDA számos módszert és eszköztárat (Morgenthaler, 2009) biztosít arra, hogy könnyebben és jobban megérthessünk, illetve átláthassunk nagyobb adathalmazokat, jelen esetben a rendszer teljesítményjellemzőit változó feltételek mellett.
Az EDA létjogosultságát legkönnyebben Anscombe „négyesével” szemléltethetjük. A négyes négy olyan adathalmazt foglal magában, amelyek egyszerű statisztikai leírói (átlag, szórás, korreláció, lineáris regressziós egyenes stb.) megegyeznek, azonban a 8. ábrára nézve azonnal látjuk, hogy közel sem hasonló adathalmazokról van szó. Az emberi agy lenyűgöző precizitással képes különféle minták teljes spektrumának felismerésére. Az EDA pont ezt a képességünket szeretné kamatoztatni.



8. ábra: Anscombe „négyese”. Forrás: Ravi, 2014

A vizuális adatelemzés egy erőteljesen intuitív folyamat, amely különösen hasznos tisztán megfigyelésalapú rendszeridentifikáció (Ljung, 1998) esetén, amikor is nem ismerjük a rendszert, vagy csak kevés előzetes ismerettel rendelkezünk a rendszer működéséről. Az EDA-t jellemzően az adott felhasználási területek szakértői végzik, akik korábbi tapasztalataikat és a megfigyeléseket kombinálva hoznak létre a megfigyelésekkel kon-

zisztens modelleket, amelyek leírják a rendszer működését. Az EDA által nyújtott intuitív vizualizációs eszköztár lehetővé teszi olyan szakértők bevonását a rendszer elemzésébe, akik nem feltétlenül rendelkeznek mély matematikai (főleg statisztikai) ismeretekkel.
Fontos megjegyezni, hogy az EDA célja nem az, hogy azonnal származtatni lehessen különböző rendszermodelleket, hanem az, hogy erre vonatkozó hipotéziseket állítsunk fel. Az EDA-val felváltva, iteratívan szokták alkalmazni a megerősítő adatelemzést (Confirmatory Data Analysis, CDA), amely során a felállított hipotéziseket, az adathalmaz alapján, statisztikai módszerekkel (például különböző próbákkal) ellenőrzik. Tehát a két módszer szorosan kapcsolódik egymáshoz, önmagukban nem használatosak (Tukey, 1980).
Az EDA másik célja a metodológiában a működési tartományok validációja és felderítése. Ha a teljesítményelemzés elején definiáltuk, hogy a rendszer hibamentes működését elemezzük normál terhelés alatt, akkor a megfigyelt viselkedésnek is ezt illik tükröznie. Ha nem így van, akkor a mérési kísérletek során valószínűleg rosszul választottuk meg a terhelési ráta felső határát, vagy alulméreteztük a rendelkezésre álló erőforrások számát.
Ez esetben a túlterhelést mutató konfigurációkhoz tartozó eredményeket elhagyjuk az adott működési tartomány elemzéséből, és átsoroljuk őket egy másik működési tartományba. Ezáltal egyrészt egy tisztább képet kapunk az aktuálisan vizsgált működési tartományról (nincsenek zavaróan kilógó értékek, úgynevezett *outlier*ek), másrészt elképzelhető, hogy fényt derítettünk egy új működési tartományra (ha esetleg még nem definiáltuk volna az első lépésben).

4.2.6. Közelítő empirikus modell

Az EDA során származtatott eredményeket kombinálva a definiált használati esetekkel és a funkcionális architektúrával, újabb elemzéseket végezhetünk a rendszeren. Többek között végezhetünk szűkkeresztmetszet-analízist (Inakoshi, 2010, Roser – Nakano – Tanaka, 2003), és meghatározhatjuk a rendszer viselkedésének kvalitatív karakterisztikáit (Oriol – Marco – Franch, 2014).
A szűkkeresztmetszet-analízis során a rendszernek azon komponensét keressük, amelynek, ha növeljük a teljesítményét, akkor a teljes rendszer teljesítménye is nő. Azaz a rendszer globális teljesítményének növelését ennek a komponensnek a teljesítménye gátolja. A legtöbb esetben az EDA során egyből nyilvánvalóvá válik, hogy melyik komponens a szűk keresztmetszet. Ha mégsem így lenne, akkor számtalan (akár automatikus) módszer létezik a szűk keresztmetszet felderítésére.
A 6. ábrán az empirikus modellalkotás után egy döntési pont látható. Előfordulhat ugyanis, hogy a szűk keresztmetszetként azonosított komponens még túl magas szintű ahhoz, hogy további vizsgálatok nélkül javítani tudjunk a teljesítményén. A 7. ábrán látható komponensek mindegyike ilyen. Hiába azonosítjuk az üzleti logika réteget szűk keresztmetszetként, ennyiből még nem fog kiderülni, hogy pontosan melyik alszolgáltatással van a probléma. Ebben az esetben a metodológia ismételt, rekurzív végrehajtására kerül sor az 4.2.2 szakasztól kezdve. Ezúttal azonban a teljesítményfelmérés alanya már csak a szűk keresztmetszetként azonosított komponens lesz. A következő szakasz ennek a finomítási lépésnek a kihívásait, és a javasolt megoldásokat tárgyalja.

4.2.7. Komponensmodell finomítása

A szűk keresztmetszetként azonosított komponens finomítása után újra meg kell ismételni a mérési kísérleteket, hiszen ezúttal részletesebb információk birtokába juthatunk a komponenssel kapcsolatban.
Ennek egyik módja, ha a teljes rendszeren megismételjük a mérési kísérleteket, amelyek eredményei már tartalmazni fogják a finomított komponens részletesebb viselkedési jellemzőit. A teljes rendszeren végzett kísérletek adott esetben időigényesek és költségesek lehetnek. Minden mért konfiguráció esetén a rendszert telepíteni kell a megfelelő környezetbe, majd a definiált terhelésprofilokat a teljes rendszeren le kell futtatni.
Erre a problémára megoldásként szolgál, ha a komponenst izolálni tudjuk a rendszer többi részétől, és önmagában tudjuk elemzés alá vetni. Ez a megoldás más IT-területeken már működőképesnek bizonyult, mint például az egységtesztelés (*unit testing*) során használt csonkok (*mock*) esetén (Coelho et al., 2006), vagy a beágyazott rendszerek tesztelésekor gyakran alkalmazott *hardware-in-the-loop* esetén (Lu et al., 2007).
A finomított komponens izolációjára adott megoldásunk is ezeken a módszereken alapszik. A módszer(ek) alapja, hogy a tesztelés (vagy jelen esetben felmérés) alatt álló komponenst nem a valódi, komplex környezetébe helyezik, hanem egy mesterséges, szimulált környezetbe, amely ugyanazokat a gerjesztéseket produkálja a komponens felé, mint eredeti környezete. Ennek köszönhetően egy komplex rendszer helyett elég egy kevésbé komplex „álrendszer” telepítése és működtetése.
Felmerül azonban egy jelentős probléma a módszer alkalmazása során. Míg az eredeti rendszerünk esetén értelmezett terhelés és kérések definíciója intuitív módon adódott (például 100 könyv adatainak lekérdezése másodpercenként), addig a belső komponenseken megfigyelhető terhelés és kérések típusa ettől merőben eltérő lehet. A könyvadatok lekérdezésének példájánál maradva, ugyanez a terhelés az adatbázis szerveren már egy adatlekérdező nyelven – például SQL (Chamberlin – Boyce, 1974) – megfogalmazott, különböző bonyolultságú lekérdezések (*query*) formájában jelenik meg.
Látható, hogy a teljes rendszer határán észlelt gerjesztések akár erős torzításon is keresztüleshetnek, mire az elemzés alatt álló komponenst elérik. Előfordulhat, hogy ezen torzítások mikéntje pontosan ismert, ezáltal a rendszergerjesztések manuálisan transzformálhatók a komponens bemenetén megkövetelt formára. Ez az eset azonban nem jellemző. A köztes szolgáltatások komplexitása általában nem teszi lehetővé, hogy pontosan felmérjük a végbement torzításokat.
A mi javaslatunk a probléma megoldására egy felvétel/visszajátszás (*record/replay*) funkció integrálása az elemzés alatt álló komponens határára. A funkció feladata, hogy pontosan rögzítse a komponenshez érkező kéréseket és az azokra adott válaszokat. Ehhez természetesen el kell végezni egy, *de csak egy* mérési kísérletet a teljes rendszeren, amelynek során rögzítésre kerülnek a kérések-válaszok. Innentől kezdve használható a mesterséges/szimulált környezet, amely ezeket a kéréseket játssza vissza a megadott sorrendben, de tetszőleges időzítéssel.
A módszer hátránya, hogy nem minden eseten elég a kérések visszajátszása. Ha a komponens inicializálásához szükséges egyéb rendszerfunkció is, akkor nem tudjuk tökéletesen izolálni a komponenst. Viszont abban az esetben, ha ez mégis megtehető, akkor vegyük észre, hogy visszavezettük az eredeti elméleti problémát, a gerjesztések transzformációját, egy technikai problémára, a felvétel/visszajátszás funkció implementálására.

4.2.8. Célzott érzékenységvizsgálat

Ha eljutottunk eddig a lépésig, az azt jelenti, hogy kellő mélységben azonosítottunk egy komponenst mint szűk keresztmetszetet. Kereskedelemben rendelkezésre álló, COTS-komponensek esetén a szűk keresztmetszet elhárítása csak a megfelelő konfiguráció megtalálásával történhet, hiszen nem tudunk direkt változtatni a komponensen mint szoftveren. Végső esetben akár szükség lehet egy alternatív, de hasonló szolgáltatást nyújtó komponens integrálására a szűk keresztmetszetet jelentő komponens helyett.
Mivel a komponens szűk keresztmetszet volt, ezért jelentősen befolyásolhatja a rendszer teljesítményét. Emiatt jellemzően ezek a komponensek vannak a további érzékenységvizsgálatok [például miként változik a komponens teljesítménye adott paraméterek megváltozásának hatására (Gokhale – Trivedi, 2002)], illetve a komponensalapú modellezés fókuszában. Ha a legfontosabb szűk keresztmetszeteket azonosítottuk, akkor a teljes rendszer viselkedését teljesítményszempontból, a felépített komponensmodellek kompozíciójából állíthatjuk elő.

4.2.9. Komponensalapú teljesítménymodellezés

A komponensalapú szoftverfejlesztés az évek során már bizonyította hatékonyságát. Kész rendszerkomponensek újrahasználásával növelhető a szoftver minősége, karbantarthatósága és jelentősen csökkenthető a fejlesztési idő. A felhasznált komponensek közös jellemzője, hogy egy jól dokumentált és stabil programozási felületet biztosítnak, miközben maguk is további komponensekre támaszkodnak. Tehát minden komponens esetén felsorolható, hogy a komponens milyen szolgáltatásokat biztosít, illetve milyen egyéb szolgáltatások meglétét várja el ahhoz, hogy működni tudjon. Ez a szemléletmód biztosítja azt, hogy komponenseket – szolgáltatásaik és függőségeik mentén – egy teljes rendszerré illesszünk össze.
A teljesítménymodellezés területén is felismerték a komponensalapú megközelítésben rejlő lehetőségeket (Koziolek, 2010; Grassi – Mirandola, 2004). A biztosított szolgáltatások és elvárt függőségek nézet kellően kifejező és egyszerű ahhoz, hogy megfelelő legyen elemzés szempontjából kezelhető, ugyanakkor matematikailag precíz modellek megalkotásához. A megközelítés előnye, hogy a teljesítménymodellezést nem az alapoktól kell kezdeni, hanem már rendelkezésre állnak a komponensek mint építőkövek modelljei.
A komponensmodellek elkészítése, majd azokból rendszermodell készítése, nagyságrendekkel kezelhetőbb folyamat, mint egyből a teljes rendszermodell előállítása. Egy komponens modellezéséhez jellemzően nem szükséges a teljes rendszer ismerete, ugyanis izolált módon azonosítható (előállítható) és hangolható. Szerencsés esetben a kritikus komponensek (mint például hálózati eszközök, protokoll implementációk) modelljeit már annak fejlesztői elkészítették egy adott formalizmus szerint, így azok újrahasználhatóak, jelentős időt megtakarítva ezzel a modellezési folyamat során.
A rendelkezésre álló komponensmodellek végül integrálásra kerülnek, jellemzően a nyújtott és elvárt szolgáltatásokat használva, a részek közötti kapcsolódási felületként. Emellett számtalan kiegészítő adattal annotálható az elkészült modell, amely például olyan információkat integrál, mint a kimért hálózati késleltetés, válaszidők és jellemző felhasználói viselkedések. A kész rendszermodell elemzésére széles körű statisztikai és lineáris algebrai apparátus áll rendelkezésre (Hoffman – Trivedi – Malek. 2007), azonban ezek áttekintése túlmutat a monográfia keretein.

5. ÖSSZEFOGLALÁS, AZ IOT-RENDSZEREK TERVEZÉSÉNEK LÉPÉSEI

Az okosszolgáltatások és infrastruktúrák alapját képező IoT-rendszerek tervezése kritikusságuk miatt kellő körültekintést igényel. Bemutattunk különböző módszereket és szabványokat, amelyek mentén szisztematikus módon felépíthető egy ilyen IoT-architektúra, amelynek lépései az alábbiak:

- A tervezendő architektúrával szemben támasztott követelmények meghatározása, amelyekhez támpontot az NIST által készített IoT-alapú okosváros referenciamodell szolgáltat.
- Az azonosított követelmények alapján szisztematikus módon kiválaszthatóak az infrastruktúra azon elemei, amelyek segítségével egy robusztus és skálázható rendszert kapunk. Ebben a Cloud Customer Council által kidolgozott felhőalapú referenciaarchitektúra nyújt támpontokat.
- Az architektúra elemeinek azonosítása után az egyes építő komponensek kiválasztására bemutattunk több szabványos benchmarkot (lásd 3. fejezet), amelyek lehetővé teszik a komponensalternatívák objektív és költséghatékony összehasonlítását.
- Végül a kiválasztott komponensekből felépített rendszer teljesítménykövetelményeknek való megfelelőségét vizsgáljuk, amelyre egy szisztematikus teljesítményfelmérési folyamatot mutattunk be, mely iteratív módon feltárja a rendszer teljesítményproblémáit, és igyekszik elhárítani azokat különböző módszerek segítségével.

Az okosalkalmazások (Anttiroiko – Valkama – Bailey, 2014) rohamos fejlődése és terjedése természetes velejárója az informatikai infrastruktúrák és szolgáltatások közműszerű elérhetőségének.
Gazdasági és társadalmi hatásuk egyaránt rendkívül nagy. Akár költségmegtakarítás, akár a felhasználó számára nyújtott komfort (például ügyintézési sebesség és egyszerűség), akár pedig a környezetvédelem szempontjából az intelligencia integrálása minőségi előrelépést ígér.
Ennek gazdasági vetülete a minden korábbinál jobb megtérülési ráta. Meglehetősen nagy keresleti piac alakult ki az ilyen alkalmazások létrehozása és üzemeltetése területén.
Az okosalkalmazások azonban a klasszikus informatikai alkalmazásokhoz képest számos olyan speciális vonással rendelkeznek, amelyek a tervezési és üzemeltetési folyamat fő hangsúlyát nem az implementációra, hanem a követelmények és az architektúra tervezésének fázisára teszik, mi több, szakértői tevékenységet igényelnek.
Fontos vonása ezeknek az új alkalmazásoknak, hogy gyakran meglevő szervezeti és tárgyi infrastruktúrára épülve növelik azok hatékonyságát, anélkül, hogy az alapokhoz az adaptáción túl érdemben hozzá kellene nyúlniuk.
Ez – elsősorban a fizikai világhoz kapcsolódó eszközök esetén – azt jelenti, hogy közel azonos szolgáltatási színvonalat lehet elérni viszonylag korlátozott járulékos

beruházással, mint amilyen egy újonnan kidolgozott és telepített rendszer képessége. Ennek megfelelően a tervezési-fejlesztési metodikának alkalmasnak kell lennie arra, hogy a meglévő szolgáltatásokat eszközként integrálja.
Másik sajátos jellemzője ennek az alkalmazási körnek, hogy várható élettartama jelentősen meghaladja a szokásos informatikai alkalmazásokat, hiszen például egy közlekedési beruházás vagy egy energetikai irányító rendszer élettartama várhatóan évtizedekben mérhető.
A komplex feladatok megoldása rendszerintegrációs alapú, azaz egy-egy ma üzembe helyezett rendszer az életciklusa során várhatóan nemcsak szigetszerűen az eredeti feladatát fogja ellátni, hanem a későbbiekben egy átfogóbb rendszer részévé fog válni. Az úgynevezett "rendszerek rendszere" szemléletben majdan ráépülő integrációs réteg például egy komplexebb feladatot oldhat meg sokaspektusú vezérlést és esetenként optimalizálást megvalósítva. Ennek megfelelően különleges kihívás az, hogy a majdani integráció érdekében a ma tervezett és létrehozott rendszerekbe az együttműködést már eleve be kell tervezni, dacára annak, hogy a későbbi integráció ma még csak körvonalaiban, vagy abban sem ismert.
A fenti követelmények jól mutatják azt, hogy az okosalkalmazások széles spektrumát egységes szemléletben, architekturálisan, az interoperabilitást biztosító módon kell tervezni. A megindult szabványosítási folyamat egyfelől a tervezési metodika, másfelől pedig úgynevezett referenciaarchitektúrák formájában a várhatóan tartós rendszerépítési elveket alkotja meg.
Az okosalkalmazások széles spektruma számára kiemelt jelentőségű az extrafunkcionális tulajdonságok biztosítása, amelyek közül a jelen kismonográfia elsődlegesen a teljesítmény és a teljesítőképesség aspektusaival foglalkozott.
Ennek kiemelt jelentőségét az adja, hogy a jövőbeni integrációs folyamatnál a funkcionalitás viszonylag keveset fog változni, de egy-egy alkalmazásnak természetszerűleg nő a terhelése, ha egyre több ráépülő szolgáltatás használja.
Ennek megfelelően a rövid távú tervezésnek biztosítania kell azt a többletkapacitást, amellyel a szolgáltatás használatának elterjedésével belátható időn belül fellépő igények kielégíthetőek. Olyan megoldásokat kell alkalmazni, amelyek a várhatóan nagy élettartamú alkalmazások teljesítményét, bővíthetőségét hosszabb távon is garantálják.
Természetesen a kellő előrelátás a tervezésben nem abszolút nóvum, elég, ha arra gondolunk, hogy az Andrássy út vagy a Nagykörút tervezésekor mekkora volt az utak forgalmi terhelése...

IRODALOMJEGYZÉK

Akay, Bahriye – Kraboga, Dervis (2012): *A modified artificial bee colony algorithm for real-parameter optimization*. Information Sciences 192 (2012): 120-142. o.

Albino, Vito – Berardi, Umberto – Dangelico, Rosa Maria (2015): *Smart cities: Definitions, dimensions, performance, and initiatives*. Journal of Urban Technology 22, no. 1. 3-21. o.

Allmendinger, Glen – Lombreglia, Ralph (2005): *Four strategies for the age of smart services*. Harvard business review 83, no. 10. 131. o.

Anttiroiko, Ari-Veikko – Valkama, Pekka – Bailey, Stephen J. (2014): *Smart cities in the new service economy: building platforms for smart services*. AI & society 29, no. 3. 323-334. o.

Apache (s.a.): *Apache Hadoop*. Internetes elérhetőség: http://hadoop.apache.org/ (Letöltés ideje: 2020. 09. 02.)

Árgilán, Viktor Sándor – Balogh, János – Békési, József – Dávid, Balázs – Galambos, Gábor – Krész, Miklós – Tóth, Attila (2014): Ütemezési feladatok az autóbuszos közösségi közlekedés operatív tervezésében: Egy áttekintés. ALKALMAZOTT MATEMATIKAI LAPOK 31. 1-40. o.

Balakrishna, Chitra (2012): *Enabling technologies for smart city services and applications*. In: Next Generation Mobile Applications, Services and Technologies (NGMAST), 2012 6th IEEE International Conference. 223-227. o.

Banga, Gaurav – Druschel, Peter (1999): *Measuring the capacity of a Web server under realistic loads*. World Wide Web 2, no. 1-2. 69-83. o.

Botta, Alessio – De Donato, Walter – Persico, Valerio – Pescapé, Antonio (2016): *Integration of cloud computing and internet of things: a survey*. Future Generation Computer Systems, 56. 684-700. o.

Budapesti Közlekedési Központ (s.a.): *BKK Futár alkalmazás*. Internetes elérhetőség: https://bkk.hu/fejleszteseink/futar/ (Letöltés ideje: 2020. 09. 02.)

Byun, Jinsung – Park, Sehyun (2011): *Development of a self-adapting intelligent system for building energy saving and context-aware smart services*. IEEE Transactions on Consumer Electronics 57, no. 1.

Chamberlin, Donald D. – Boyce, Raymond F. (1974): *SEQUEL: A structured English query language*. In: Proceedings of the 1974 ACM SIGFIDET (now SIGMOD) workshop on Data description, access and control. 249-264. o.

Chen, Shimin (2009): *FlashLogging: exploiting flash devices for synchronous logging performance*. In: Proceedings of the 2009 ACM SIGMOD International Conference on Management of data. 73-86. o.

Chung, Lawrence – do Prado Leite, Julio Cesar Sampaio (2009): *On non-functional requirements in software engineering*. In: Conceptual modeling: Foundations and applications, Springer, Berlin, Heidelberg. 363-379. o.

Coelho, Roberta – Kulesza, Uirá – von Staa, Arndt – Lucena, Carlos (2006): *Unit testing in multi-agent systems using mock agents and aspects*. In: Proceedings of the 2006 international workshop on Software engineering for large-scale multi-agent systems, ACM. 83-90. o.

Compton, Michael – Barnaghi, Payam – Bermudez, Luis – GarcíA-Castro, RaúL – Corcho, Oscar – Cox, Simon – Graybeal, John – Hauswirth, Manfred – Henson, Cory – Herzog, Arthur – Huang, Vincent – Janowicz, Krzysztof – Kelsey, W. David – Phuoc, Danh Le – Lefort, Laurent – Leggieri, Myriam – Neuhaus, Holger – Nikolov, Andriy – Page, Kevin – Passant, Alexandre – Sheth, Amit – Taylor, Kerry (2012): *The SSN ontology of the W3C semantic sensor network incubator group*. Web semantics: science, services and agents on the World Wide Web, 17. 25-32. o.

CSCC (2016): *Cloud Customer Architecture for IoT*. Internetes elérhetőség: http://www.cloud-council.org/deliverables/cloud-customer-architecture-for-iot.htm (Letöltés dátuma: 2020. 09. 02.)

CSCC (s.a.): *Cloud Standards Customer Council*. Internetes elérhetőség: http://www.cloud-council.org/ (Letöltés dátuma: 2020. 09. 02.)

Cuzzocrea, Alfredo – Song, Il-Yeol – Davis, Karen C. (2011): *Analytics over large-scale multidimensional data: the big data revolution!*. In: Proceedings of the ACM 14th international workshop on Data Warehousing and OLAP, ACM. 101-104. o.

Dabir, Abes – Matrawy, Ashraf (2007): *Bottleneck analysis of traffic monitoring using wireshark*. In: Innovations in Information Technology, IIT'07. 4th International Conference, IEEE. 158-162. o.

Fang, Ru – Hsiao, Hui-I. – He, Bin – Mohan, C. – Wang, Yun (2011): *High performance database logging using storage class memory*. In: Data Engineering (ICDE), IEEE 27th International Conference. 1221-1231. o.

Geimer, Markus – Shende, Sameer S. – Malony, Allen D. – Wolf, Felix (2009): *A generic and configurable source-code instrumentation component*. In: International Conference on Computational Science. Springer, Berlin, Heidelberg. 696-705. o.

Ghazal, Ahmad – Rabl, Timann – Hu, Minqing – Raab, Francois – Poess, Meikel – Crolotte, Alain – Jacobsen, Hans-Arno (2013): *BigBench: towards an industry standard benchmark for big data analytics.* In: Proceedings of the 2013 ACM SIGMOD international conference on Management of data. 1197-1208. o.

Ghosh, Prasanta – Bhatt, Vadiraja – Vaitheeswaran, Girish (2004): *Database system with improved methods for asynchronous logging of transactions*. U.S. Patent 6,721,765, issued April 13.

Gokhale, Swapna S. – Trivedi, Kishor S. (2002): *Reliability prediction and sensitivity analysis based on software architecture*. In: Proceedings of Software Reliability Engineering, ISSRE 2003. 13th International Symposium, IEEE. 64-75. o.

Grassi, Vincenzo – Mirandola, Raffaela (2004): *Towards automatic compositional performance analysis of component-based systems*. In: ACM SIGSOFT Software Engineering Notes, vol. 29, no. 1. 59-63. o.

Hoffman, Guenther A. – Trivedi, Kishor S. – Malek, Miroslaw (2007): *A best practice guide to resource forecasting for computing systems*. IEEE Transactions on Reliability *56*, no. 4. 615-628. o.

Huppler, Karl (2009): *The art of building a good benchmark*. In: Technology Conference on Performance Evaluation and Benchmarking. Springer, Berlin, Heidelberg. 18-30. o.

Inakoshi, Hiroya (2010): *Method and apparatus for performance bottleneck analysis*. U.S. Patent Application 12/576,944, filed April 15, 2010.

Kanungo, Anurag – Sharma, Ayush – Singla, Chetan (2014): „*Smart traffic lights switching and traffic density calculation using video processing."* In: Recent advances in Engineering and computational sciences (RAECS), IEEE. 1-6. o.

Kiczales, Gregor – Lamping, John – Mendhekar, Anurag – Maeda, Chris – Lopes, Cristina – Loingtier, Jean-Marc – Irwin, John (1997): *Aspect-oriented programming*. In: European conference on object-oriented programming. Springer, Berlin, Heidelberg. 220-242. o.

Knight, John C. (2002): *Safety critical systems: challenges and directions*. In: Proceedings of the 24th International Conference on Software Engineering, 2002. *ICSE 2002.*, IEEE. 547-550. o.

Kocsis, Imre – Pataricza, András – Micskei, Zoltán – Kövi, András – Kocsis, Zsolt (2013): *Analytics of resource transients in cloud-based applications*. International Journal of Cloud Computing 1, 2, no. 2-3. 191-212. o.

Koziolek, Heiko (2010): *Performance evaluation of component-based software systems: A survey*. Performance Evaluation 67, no. 8 (2010): 634-658. o.

Leskovec, Jure – Rajaraman, Anand – Ullman, Jeffrey David (2014): *Mining of massive datasets*. Cambridge university press.

Liu, Sen – Yang, Yang – Qu, Wen Guang – Liu, Yuan (2016): „*The business value of cloud computing: the partnering agility perspective."* Industrial Management & Data Systems 116, no. 6. 1160-1177. o.

Ljung, Lennart (1998): *System identification*. In: Signal analysis and prediction, Birkhäuser, Boston, MA. 163-173. o.

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 12, no. 4. 559-592. o.

Lu, Bin – Wu, Xin – Figueroa, Hernan – Monti, Antonello (2007): *A low-cost real-time hardware-in-the-loop testing approach of power electronics controls*. IEEE Transactions on Industrial Electronics 54, no. 2. 919-931. o.

Michalewicz, Zbigniew – Schoenauer, Marc (1996): *Evolutionary algorithms for constrained parameter optimization problems.* Evolutionary computation 4, no. 1. 1-32. o.

Morgenthaler, Stephan (2009): *Exploratory data analysis*. Wiley Interdisciplinary Reviews: Computational Statistics 1, no. 1. 33-44. o.

National Grid UK (s.a.): *How we balance the electricity transmission system*. Internetes elérhetőség: https://www.nationalgrid.com/uk/about-grid/our-networks-and-assets/how-we-balance-electricity-transmission-system (Letöltés dátuma: 2019. 06. 02.)

NIST (s.a.): *National Institute of Standards and Technology*. Internetes elérhetőség: https://www.nist.gov (Letöltés dátuma: 2020. 09. 02.)

NIST Impacts (s.a.): *Alliance for 5G Networks*. Internetes elérhetőség: https://www.nist.gov/industry-impacts/alliance-5g-networks (Letöltés dátuma: 2020. 09. 02.)

NIST Industry Impacts (s.a.): *Building Automation and Control*. Internetes elérhetőség: https://www.nist.gov/industry-impacts/building-automation-and-control (Letöltés dátuma: 2020. 09. 02.)

NIST-IESCF (2018): *A Consensus Framework for Smart City Architectures*. Internetes elérhetőség: https://s3.amazonaws.com/nist-sgcps/smartcityframework/files/ies-city_framework/IES-CityFrameworkdraft_20180207.pdf (Letöltés dátuma: 2020. 09. 02.)

NIST-SCA (s.a.): *Smart Cities Architecture*. Internetes elérhetőség: https://pages.nist.gov/smartcitiesarchitecture/ (Letöltés dátuma: 2020. 09. 02.)

OpenFog Consortium (2017): *OpenFor Reference Architecture for Fog Computing*. Internetes elérhetőség: https://www.openfogconsortium.org/ra/technical-document-download/ (Letöltés dátuma: 2020. 09. 02.)

Oracle (s.a.): *Maximize Performance for Oracle Database*. Internetes elérhetőség: https://www.oracle.com/support/advanced-customer-support/database/index.html (Letöltés dátuma: 2020. 09. 02.)

Oriol, Marc – Marco, Jordi – Franch, Xavier (2014): *Quality models for web services: A systematic mapping*. Information and software technology 56, no. 10: 1167-1182. o.

Ostroff, Jonathan S. (1992): *Formal methods for the specification and design of real-time safety critical systems*. Journal of Systems and Software 18, no. 1. 33-60. o.

Post, Hendrik – Sinz, Carsten – Merz, Florian – Gorges, Thomas – Kropf, Thomas (2009): *Linking functional requirements and software verification*. In: Requirements Engineering Conference, 2009. RE'09. 17th IEEE International. 295-302. o.

Ravi, Parikh (2014): *Anscombe's Quartet, and Why Summary Statistics Don't Tell the Whole Story*. Internetes elérhetőség: https://heapanalytics.com/blog/data-stories/anscombes-quartet-and-why-summary-statistics-dont-tell-the-whole-story (Letöltés dátuma: 2020. 09. 02.)

Romanovsky, Alexander – Ishikawa, Fuyuki (2016): *eds. Trustworthy Cyber-Physical Systems Engineering*. CRC Press.

Roser, Christoph – Nakano, Masaru – Tanaka, Minoru (2003): *Simulation test bed for manufacturing analysis: comparison of bottleneck detection methods for AGV systems*. In: Proceedings of the 35th conference on Winter simulation: driving innovation. Winter Simulation Conference. 1192-1198. o.

SearchDataCenter (s.a.): *Edge Computing Definition*. Internetes elérhetőség: https://searchdatacenter.techtarget.com/definition/edge-computing (Letöltés dátuma: 2020. 09. 02.)

Sowa, John F. (2000): *Knowledge representation: logical, philosophical, and computational foundations*. Vol. 13. Pacific Grove: Brooks/Cole.

Strategy& (2017): *Mission critical: How GCC telecom operators can enable public safety communications*. Internetes elérhetőség: https://www.strategyand.pwc.com/reports/mission-critical (Letöltés dátuma: 2019.06.02.)

The Statistics Portal (2017): *Facebook's advertising revenue worldwide from 2009 to 2017 (in million U.S. dollars)*. Internetes elérhetőség: https://www.statista.com/statistics/271258/facebooks-advertising-revenue-worldwide/ (Letöltés dátuma: 2020. 09. 02.)

TPC (2016): *TPCx-BB Big Data Benchmark*. Internetes elérhetőség: http://www.tpc.org/tpcx-bb/default5.asp (Letöltés dátuma: 2020. 09. 02.)

TPC (2018a): *TPCx-IoT Benchmark for IoT Gateway Systems*. Internetes elérhetőség: http://www.tpc.org/tpcx-iot/default5.asp (Letöltés dátuma: 2020. 09. 02.)

TPC (2018b): *TPCx-HS V2 Big Data System Benchmark*. Internetes elérhetőség: http://www.tpc.org/tpcx-hs/default5.asp (Letöltés dátuma: 2020. 09. 02.)

Tukey, John W. (1980): We need both exploratory and confirmatory. *The American Statistician 34*, no. 1. 23-25. o.

Zygiaris, Sotiris (2013): *Smart city reference model: Assisting planners to conceptualize the building of smart city innovation ecosystems*. Journal of the Knowledge Economy 4, no. 2. 217-231. o.

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:
Kelemen Dóra

Tördelőszerkesztő:
Friebert Máté

ISBN 978-963-498-110-7 (PDF)