Nagyfelbontású és átlátható termelésinformatika

Új információs kulcstechnológiák beszállító-menedzsment rendszerek számára

PhD Tézisfüzet

Karnok Dávid

Témavezető: Monostori László, akadémikus

Budapesti Műszaki és Gazdaságtudományi Egyetem
Gépészmérnöki kar

Pattantyús Ábrahám Géza Doktori Iskola,
Anyag és gyártástudomány program

Magyar Tudományos Akadémia (MTA)
Számítástechnikai és Automatizási Kutató Intézet (SZTAKI)

Budapest, Magyarország, 2017

1. Bevezetés és motiváció

Napjainkban egy új ipari forradalom kiteljesedése előtt állunk, amit gyakran **Ipar 4.0**-nak (Industry 4.0) [37] vagy **Kiber-fizikai Gyártórendszereknek** (Cyber-Physical Production Systems, CPPS) is neveznek [50]. A számítástudomány eredményeinek egyenrangúvá válása a gyártásban, logisztikában és az ezen belüli folyamatokban egy olyan, hosszú ideje dédelgetett álom, ami most már valóra válhat.

Még ha a számítógépek, szoftverek és algoritmusok a gyártó- és logisztikai rendszerek mindennapos részeit képezték is az elmúlt évtizedekben, fontosságban másodrendű mivoltuk azt jelentette, hogy a fizikai és információs folyamatok között gyakran nem létezett kapcsolat, vagy ha mégis, akkor késleltetésben (az információ késett a termékhez képest vagy fordítva) és minőségben (hiányos, hibás, zajos vagy felesleges adatok) komoly különbségek adódtak. A mindenütt jelenlévő Internet, a vezeték nélküli kommunikációs lehetőségek, valamint a termékeken és komponenseken közvetlenül és gazdaságosan elhelyezhető számítási teljesítmény lehetővé teszik a kiber és fizikai rendszerek kéz-a-kézben megvalósuló működését, amivel nagyobb flexibilitás, jobb válaszidők és általánosan feljavult fő teljesítményindikátorok érhetők el [49].

Az elmúlt évtizedben az elérhető adatok mennyisége és gyakorisága a gyártásban és a logisztikában szinte exponenciálisan növekedett. Az ún. big-data forradalom **nagyfelbontású** [63] információs technológiák és irányítási megoldások létrehozására biztat mind az offline, mind az online – közel valósidejű – adat- és eseményfeldolgozás területén.

A közelmúlt műszaki és tudományos fejlődése számtalan olyan stratégiát és kulcsfontosságú területet határozott meg, amelyek lehetővé teszik a klasszikus ipari termelés és logisztika átalakítását az új CPPS megközelítési és megoldási módokra, mint például – a disszertációhoz kapcsolódó nézőpontból – az **átfogó digitális integráció**, **komplex rendszerek menedzselése** és **erőforrás-hatékonyság**. Ezen felül javaslom a hangsúlyt a rendszerek és komponensek reaktív jellegű (adat- és számítási reaktivitás értelemben vett) megvalósítására helyezni, olyan formában, ahogy azt a Reaktív Kiáltvány (Reactive Manifesto)[1] megfogalmazza: a szoftverek **rugalmas és üzenetalapú** felépítése, ami egyúttal **hibatűrő és időben válaszoló (reszponzív)** is.

A legtöbb vállalat azonban nem tud „máról holnapra” a CPPS-világhoz csatlakozni, mert egy ekkora lépés átfogó, gondos rendszer- és folyamat-tervezést és szigorú változásmenedzsmentet tesz szükségessé. Azon vállalatoknál, amelyekkel együtt dolgoztam, hiába volt naprakész a hardver, ha ehhez hagyományos, elavult szoftver-technológiákkal készült szoftverrendszerek tartoznak, ami akár

[1] http://www.reactivemanifesto.org

5–10 évnyi ún. **technológiai adósság** felhalmozódását is jelenti (pl. táblázatkezelők, több verzióval korábbi nyelveken írt programok, webes szabványokat nem vagy alig támogató böngészők használata).

Az elmúlt tíz évben több ipari–akadémiai kutatási és fejlesztési projektben vettem részt és a különböző partner vállalatok – mérettől, helyszíntől vagy céltól függetlenül – mind hasonló problémákkal küzdöttek. Ezek az alábbi pontokra vezethetők vissza:

- a (pillanatnyi) üzemi vagy logisztikai állapot nem elérhető előfeldolgozott formában és/vagy kellő időben,
- a meglévő vállalatirányítási rendszer (Enterprise Resource Planning, ERP) megvalósítása korlátozott hatáskörű, az ERP rendszer továbbfejlesztése nagyon költséges,
- a szoftver lassú vagy elégtelen (pl. táblázatkezelő) vagy az információfolyam papír- és telefon-alapú, és
- a termelésinformatika korlátozott adatforrásokkal kénytelen dolgozni titoktartási aggályok miatt (pl. vállalati adatszivárgás megakadályozása a bővebb forrásokból kinyerhető új információk lehetőségeivel szemben).

Érdekes módon, ezen kutatásra érdemes problémák látszólag már rendelkeznek megoldással az akadémiai kutatások eredményei és/vagy Információs Technológiák (IT) fejlődése miatt. Az alaposabb áttekintés után azonban kiderül, hogy a meglévő módszerek vagy (1) túl általánosan lettek megfogalmazva és jelentős hézagokat kell leküzdeniük az implementációknak, vagy (2) olyan specifikus tulajdonságokkal rendelkező területen vagy vállalatnál kerültek megvalósításra, amelynek nincs használható analógiája a megcélzott alkalmazási környezetben. Az első pont az akadémiai kutatás ún. fentről-lefelé (top-down) megközelítésének tulajdonítható; még ha az eredeti munka egy konkrét vállalatnál elért konkrét eredményekből is származik, az általánosítás információvesztéssel járt, valamint a kellően részletes példák alkalmi hiánya megnehezíti az eredmények felhasználását. A második eset jellemzően nem siklik el a részletek felett, ám emellett gyakran híján van a mélyebb, átfogó elveknek. Ezen felül az elérhető IT fejlesztések csak lassan kerülnek felhasználásra és a késlekedés látszólag 5–10 évnyi lemaradást jelent; habár az IT projektek 1–2 éven belül befejeződnek, mégis akár 5 évnél is régebbi technológiákat és módszereket alkalmaznak. Ez a lemaradás nem feltétlenül a felmerülő hardver-követelmények miatt adódik, hanem inkább költségvetési okokkal, konzervatív menedzsment-filozófiákkal vagy éppen a személyzet trendekkel és modern technológiákkal való naprakészségének hiányával magyarázható ezen vállalatoknál. Emellett, ahogy az a változás-menedzsmentben ismert, általában közepes vagy jelentős ellenállás tapasztalható a folyamatok, a

teljesítmény és az átláthatóság feljavítását célzó új technológiákkal és megközelítésekkel szemben, amihez nagyban hozzájárul az elbocsátásoktól való vélt vagy valós félelem, ami egy ilyen rendszer éles üzembe állítását követheti. Ez utóbbival szemben a tapasztalat azt mutatja, hogy az Ipar 4.0 szociális aspektusával egybehangzóan [37, p. 55] emberi operátorokra nagyobb szükség van, mint valaha. A korábbi, főleg kézi adatbevitelre és egyszerű adatfeldolgozásra támaszkodó emberi munkakörök átalakulnak olyan magasabb szintű munkavégzéssé, ahol a dolgozók a fejlett, IT-támogatott folyamatok tervezőiként, felügyelőiként és döntéshozóiként tevékenykednek.

Következésképpen, figyelembe véve ezeket a jellemzőket, a megoldásokat inkrementálisan, alulról felfelé építkezve (bottom-up) érdemes alkalmazni és bevezetni. A disszertációban bemutatott új eredmények kitalálása, megtervezése és gyakorlati megvalósítása is ebben a szellemben történt.

A disszertáció mögötti tudományos motiváció az, hogy szükség van olyan új metodológiákra, módszerekre, modellekre és algoritmusokra, amelyek kibővítik vagy akár lecserélik a korábbi változataikat, mivel a (majdnem) állandóan kapcsolatban lévő, egymással kommunikáló és számítást végző entitások világában a korábbi változatok működésképtelenek, nem elég hatékonyak vagy nem üzemeltethetők többé gazdaságosan. A disszertáció a gyártásban és logisztikában előforduló IT környezet néhány kulcsfontosságú eleméhez mutat ilyen újdonságokat. Az eredmények birtokában a problémák megoldása és információ-technológiában történő megvalósítása a jövőben kétségkívül sokkal könnyebb és átláthatóbb lesz.

Érdekes módon, a disszertációban bemutatott kutatások és eredmények (vagy éppen az IT szektor általában) már legalább 10 éve a CPPS irányba mutattak, gyakran tudatosan, máskor nem szándékosan. Az általánosabb Kiber-fizikai Rendszerek (Cyber-Physical Systems, CPS) [39] paradigma gyártásba és logisztikába való bevezetése nem hirtelen, hanem több évnyi fejlődést követően történt, felismerve annak újszerűségét. Véleményem szerint a jövő elsődleges feladata előreláthatólag a CPPS-kompatibilis elgondolások, modellek, megközelítések és algoritmusok felfedezése lesz. A CPPS széleskörű megvalósítását segítő hozzájárulások és a részletek „helyes” kitöltése nagyon fontos aspektus, különösen, ha el akarjuk kerülni az IT-világban nem ritka, zsákutcába vezető megoldásokat.

2. Új tudományos eredmények

2.1. Vevő–beszállító közötti tervezés koordinációja csatornákkal minimális támogató adatmodell segítségével

2.1.1. Bevezetés és irodalmi áttekintés

A termelési hálózatok globális viselkedése a partnerek lokális szándékaiból és akcióiból alakul ki. A készlettervezés egy termelési hálózatban elosztott törekvésnek tekinthető, ahol a folytonosan változó körülmények figyelembe vételével a jövőbeli (várható) végtermék igényt az előállításához szükséges komponensek beszerzésével kell összhangba hozni. Habár a döntéseket helyileg és autonóm módon hozzák meg, a termelési hálózatokban a partnereknek mégis összhangban kell cselekedniük.

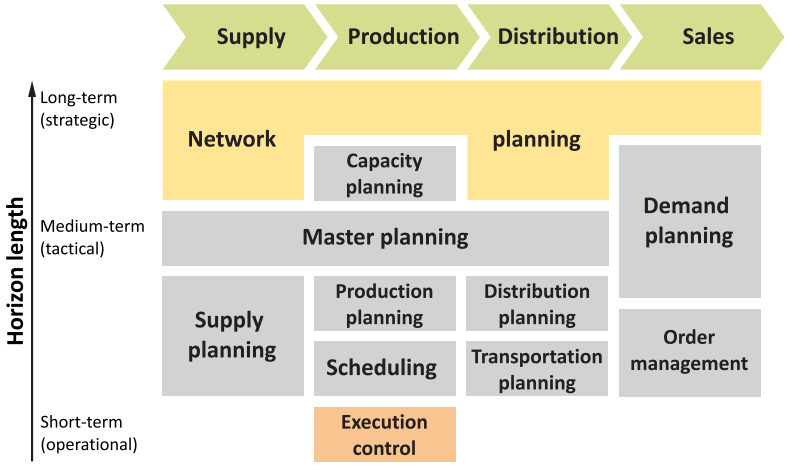
A probléma alapvetően az elosztott tervezésre vezethető vissza: a hálózat tagjai valamilyen jövőbeli eseményt szeretnének befolyásolni a jelenleg rendelkezésükre álló sokrétű információk alapján. Egyes információk biztosnak tekinthetők, mint például a termékekkel, gyártástechnológiákkal, erőforrás-képességekkel vagy értékesítési előzményekkel kapcsolatosak. Azonban az elérhető információk kiemelkedően fontos része hiányos és bizonytalan, mint például az igény-előrejelzések vagy az erőforrások és anyagok várható rendelkezésre állása. Ezen felül további nehézséget jelent, ha az információ elavult, késik vagy hibákkal tűzdelt amiatt, ahogy azt a (kapcsolódó információs) rendszerekben eredetileg rögzítették.

A Termelésinformatika a bizonytalanság és strukturális komplexitás kezelésére már rendelkezik jól bevált megközelítési módokkal. Az **aggregáció** összevonja a termékek, rendelések, igény-előrejelzések, gyártási folyamatok, erőforrás-kapacitások, műveleti- és szállítási-idők részletes információit. Néhány tervezési probléma – úgymint gyártásütemezés, termeléstervezés vagy főtervezés – több és több részlet hosszabb és hosszabb távú horizonton történő összevonásaként van meghatározva [59]. A megoldásokat egy hierarchikus tervezési folyamatban állítják elő, ahol a magasabb szintű megoldások korlátokat támaszthatnak az alacsonyabb szintű problémák megoldásával szemben.

Ennélfogva, a terveknek lehetnek stratégiai, taktikai vagy végrehajtásbeli dimenziói az időbeli távlat (horizont) és részletesség függvényében. Egy adott szinten a dekompozíció szétbontja a tervezési problémákat könnyebben megoldható al-problémákra. Ez a helyzet például a taktikai szinten, ahol a termeléstervezés elválik a beszállítási- vagy terjesztési tervezéstől.

A termelésmenedzsmentben a tervezési funkciók fejlődése egy általános hierarchikus tervezési mátrix kialakulásához vezetett [31, 65]. Az 1. ábra a tipikus stratégiai (hosszútávú), taktikai (középtávú) és végrehajtásbeli (rövidtávú) szintek

tervezési funkcióit mutatja be, amelyek a fő anyag- és információ-áramlás köré szerveződnek. Ezek a funkciók többségében a termelési hálózat minden egyes csomópontjánál megtalálhatók, bár nyilvánvalóan eltérő formában és komplexitásban valósulnak meg.



1. ábra: Stratégiai, takikai és végrehajtásbeli tervezési funkciók mátrixa [4] p. 298 [31] követve.

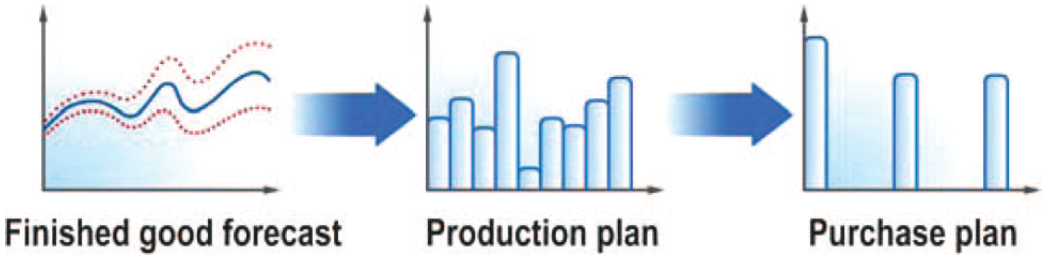
Egy vállalaton belül az elszigetelt tervezési modulok közötti koordináció önmagában is komoly probléma [31, 45]. Azonban ez egy sokkal kritikusabb helyzetet teremt a termelési hálózatban, ahol a megfelelő információcsere az elsődleges előfeltétele a partnerek közötti **kollaborációnak** [43]. Az együttműködésnek általában három szintjét szokták megkülönböztetni. A **koordináció** során a partnerek egy közös cél érdekében működnek együtt, de a résztvevők megtartják önállóságuk nagy részét. A **kooperáció** során a partnerek szorosabban működnek együtt a közös cél érdekében, néha megosztva saját erőforrásaikat is. Végezetül a **kollaboráció** során a partnerek stratégia-szinten eldöntik, hogy a kooperációhoz hasonlóan, de szorosabban és aktívan, tevőlegesen együttműködnek a közös vagy átfedő célok érdekében.

A motiváció itt a termelési hálózatokban előforduló koordinált tervezés elmélete és gyakorlata közötti szakadék áthidalása. A konkrét hátteret egy ipari–akadémiai kutatási és fejlesztési (K+F) projekt szolgáltatta, amely egy egyedi tömeggyártással foglalkozó hálózat teljesítményének fokozását tűzte ki célul [51, 68].

A koordinált beszállítás-tervezéssel kapcsolatos kutatások egészen a készletmenedzsmentig nyúlnak vissza, ahol a legfőbb kérdés, hogy mikor és mekkora rendelést adjanak fel. Tipikusan, korábban olyan központosított, csatorna-alapú koordinációs modelleket fejlesztettek ki, ahol a résztvevő partnerek közül egy

mindig rendelkezett az összes elérhető információval, így optimális döntéseket tudott hozni. A **csatorna**, ebben a környezetben és a disszertációban, egy konkrét anyagnak és a hozzá tartozó információknak a vevő és a beszállító közötti áramlását összefogó logikai-fizikai kapcsolatként értelmeződik, amit önmagában, a többi csatornától viszonylag függetlenül kezelnek a partnerek. A központosított megközelítés a gyakorlatban azonban ritkán megvalósítható, mivel a partnerek a beszállítói láncban különálló, autonóm üzleti és jogi entitásokként tevékenykednek. Ehelyett az ún. folyásirányú (upstream) tervezés a legelterjedtebb kollaborációs forma jelenleg [21]. Mivel a valóságban egyetlen partner sem képes a láncot, vagy éppen az egész hálózatot, egyedül irányítani, egyre nagyobb igény mutatkozik az elosztott irányítási módokra mind determinisztikus, mind sztochasztikus környezetben. [23].

Amikor a sorozatnagyságról döntés születik, a párhuzamos komponensigények aggregálásra kerülnek. Azonban a végleges és tényleges komponensigények gyakran köszönőviszonyban sincsenek az eredeti végtermék igény előrejelzéssekkel (lásd 2. ábra). Az elektronikus információcsere korában ez a szakadék könnyen áthidalható lenne, viszont a résztvevő partnereknek általában nincs motivációjuk az ilyen privát üzleti információk megosztására [29, 68].



2. ábra: A végtermék igény előrejelzés átalakulása a tervezési lépések során [4] p. 299.

A gyakorlatban léteznek olyan információmegosztó megoldások, amelyek támogatják a termelési hálózatokban keletkező rendelések feldolgozását. Egy példa a központosított SupplyOn[2] platform, ami az autó- és egyéb termékek gyártására szakosodott ipar számára készült több európai autóalkatrész beszállító együttműködése révén. A megoldás a (digitális) ún. Elektronikus Adatcsere (Electronic Data Interchange, EDI) formátumot használja, de támogatja a WebEDI szabványt is, amihez technikailag csak internet-hozzáférés szükséges [44]. Egy hasonló megközelítést nyújt a myOpenFactory nevű megoldás [62], ami egy központosított információ-megosztó és -kezelő ügynökséget javasol. Az utóbbi időben tárgyalások zajlanak a termelési hálózatokban megvalósítandó koordinált – és végső fokon akár kooperatív – tervezés érdekében. Ennek elősegítésére a kapcsolódó

[2] http://www.supplyon.com/, legutóbbi hozzáférés: 2017. július

kutatás mellett [40] menetrend is készült a meglévő és felmerülő problémák tanulmányozására [51, 68, 69].

2.1.2. Koordináció egy új, közös platformon keresztül

A központi vevő és a beszállítói hálózata közötti anyagigény és -ellátás koordinálásához a vevőnek meg kell osztania a részletes gyártási tervét és anyagigény-előrejelzését, cserébe minden egyes beszállítónak meg kell osztania a saját gyártókapacitásáról és szállítási terveiről szóló információkat. Ebben a kontextusban a cél, hogy ritkán, vagy egyáltalán ne keletkezzen (előre nem látható) hiány a vevői oldalon, illetve tudható legyen, hogy mikor és mennyi anyagot fognak az egyes beszállítók szállítani, miközben az információk titokban maradnak az ugyanazon hálózatban jelenlévő esetleges versenytársak előtt.

1. tézis: A vevő és beszállítói közötti információcsere során az adatokat a résztvevők naprakészen tartják egy új, közös platform segítségével. A platform egy új, minimális interakciós modellt definiál, ahol:

1. **az interakciós egység az ún. csatorna (ami információkat tartalmaz az anyagról, a beszállítójáról, készletekről és a hiánykezelési szabályokról);**
2. **a felhasználók és csatornák össze vannak rendelve, biztosítva, hogy a különböző beszállítók nem láthatják és módosíthatják egymás adatait;**
3. **hozzáférhető a dinamikus és részletes gyártási, előrejelzési, készletszintbeli és szállítási ütemterv; és**
4. **lehetőség van a problémás csatornákat rugalmasan kiszűrni a kézzel történő beavatkozáshoz.**

A közös platform biztosítja, hogy az a tervezési horizont két szintjén (középtávú, rövidtávú) és az automatizáltság két módjában (fél-automatikus, kézi) a résztvevő felhasználók egy új protokoll szerint ellenőrzik és megerősítik a minden egyes csatornára vonatkozó terveket vagy kiigazítják azokat és felkínálják a platform számára újraértékelés céljából.

Ezen felül a platform garantálja a résztvevő felhasználóknak az egyes csatornáknál, hogy mindkét fél ugyanazt az információt látja, de azt szerepüknek megfelelő módon módosíthatják csak (pl. a vevő meghatározhatja az igény-teljesítési szabályokat, a beszállító pedig a részletes szállítási ütemtervét, de fordítva nem).

Kapcsolódó elsődleges publikációk: [1], [2], [4]
Kapcsolódó további publikációk: [5], [6], [7], [9], [11], [12], [14], [13], [17],

[18], [20]

2.1.3. Eredmények alkalmazása

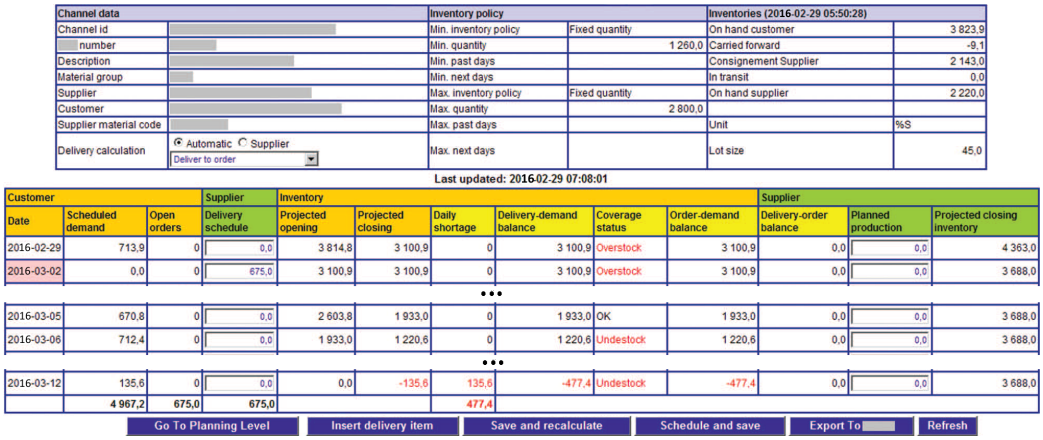
Az ezek alapján kifejlesztett és mindennapos ipari használatba került megoldás, a **Logisztikai Platform (Logistics Platform, LP)** azt az egy, fokális vevőből és számos beszállítóból álló hálózat struktúráját követi, ahol a platform bevezetésre került. Az LP egy szabványos Java Vállalati célú (Java Enterprise Edition, J2EE) webalkalmazás, ami ezáltal mind hagyományos, mind felhő-alapú szerveren is tud üzemelni. Az alkalmazást a vevő intraneten, míg a külső beszállítók virtuális magánhálózaton (Virtual Private Network, VPN) keresztül érhetik el.

Minden felhasználó rendelkezik egy saját csatorna-listával, amit megtekinthet és szerkeszthet. Ez biztosítja, hogy a titoktartás fennmarad a különböző beszállítók esetében, mivel azok csak a saját maguk által szállított anyaghoz kapcsolódó csatornákkal léphetnek interakcióba. Minden ilyen csatornán a vevői és beszállítói oldalról is ugyanazokat az adatokat olvashatják a felhasználók, de a megjelenített adatokat már különböző jogosultságok alapján módosíthatják csak. Például a beszállító felhasználója módosíthatja a szállítási ütemtervet, míg a vevő ezt nem teheti meg. Ezzel szemben csak a vevő állíthatja be a készlet-ellenőrzési szabályokat (lásd 3. és 4. ábrát).

A webalkalmazás az adatokat közvetlen adatbázis-eléréssel a vevő saját, hagyományos ütemező és tervező rendszereiből gyűjti össze, illetve képes ún. Bővíthető Jelölőnyelv (Extensible Markup Language, XML) alapú adatcserére a vevő és a beszállítók ERP rendszereivel is. Az XML-alapú adatcsere a vevőkkel történhet automatikus módon egy ún. biztonságos Egyszerű Objektum-hozzáférés Protokollon (Simple Object Access Protocol, SOAP) keresztül, valamint XML-fájl-feltöltéssel, amely adatok ellenőrzése a feltöltő felhasználói fiókjának kontextusában történik.

Mivel a rendszer alapjául szolgáló adatmodell minimalista részletességű, az alkalmazás ellátható mind a rövidtávú vevői tervezőrendszerből [5] származó, valósidejű termelési adatokkal, mind az ún. mi-lenne-ha típusú termelési-folyamat szimulációk [2] által generált adatokkal. Erre azért van lehetőség, mert a modell nem tesz kikötést a vevő adatainak eredetére. A gyakorlatban, ha ugyanazon az alkalmazáson keresztül történik a valós vagy szimulált adatok kiértékelése, a felhasználói felület jól megkülönböztető módon jelzi ezen utóbbi, eltérő üzemmódot a színezés és/vagy figyelmeztető címkék használatával.

A Logisztikai Platform bevezetése az általános kiszolgálási szintet (hány százalékban sikerült az adott anyagot egy adott napon a tervezett mennyiséghez képest felhasználni) jelentősen növelte. Az 5. ábra egy tipikus komponens kiszolgálási szintjének alakulását mutatja a platform bevezetése előtti és utáni időszakban.



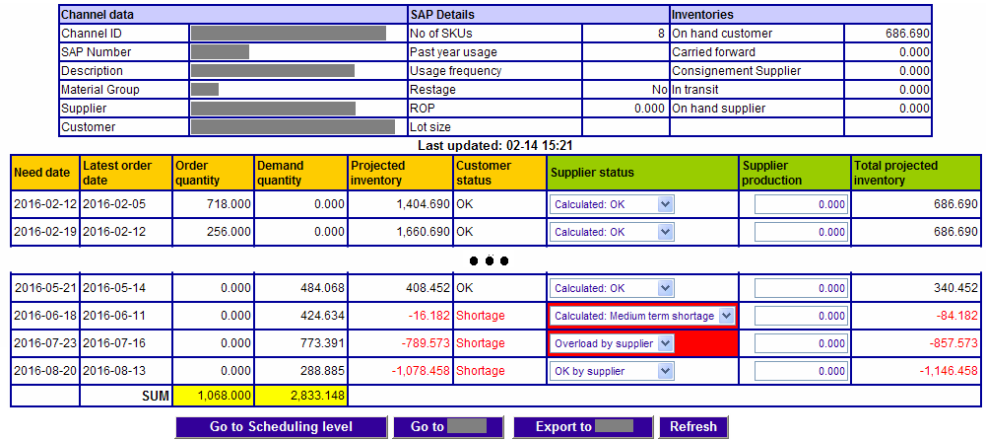
Channel data		Inventory policy		Inventories (2016-02-29 05:50:28)	
Channel id		Min. inventory policy	Fixed quantity	On hand customer	3 823,9
number		Min. quantity	1 260,0	Carried forward	-9,1
Description		Min. past days		Consignement Supplier	2 143,0
Material group		Min. next days		In transit	0,0
Supplier		Max. inventory policy	Fixed quantity	On hand supplier	2 220,0
Customer		Max. quantity	2 800,0		
Supplier material code		Max. past days		Unit	%S
Delivery calculation	Automatic / Supplier; Deliver to order	Max. next days		Lot size	45,0

Last updated: 2016-02-29 07:08:01

Customer			Supplier	Inventory						Supplier		
Date	Scheduled demand	Open orders	Delivery schedule	Projected opening	Projected closing	Daily shortage	Delivery-demand balance	Coverage status	Order-demand balance	Delivery-order balance	Planned production	Projected closing inventory
2016-02-29	713,9	0	0,0	3 814,8	3 100,9	0	3 100,9	Overstock	3 100,9	0,0	0,0	4 363,0
2016-03-02	0,0	0	675,0	3 100,9	3 100,9	0	3 100,9	Overstock	3 100,9	0,0	0,0	3 688,0
•••												
2016-03-05	670,8	0	0,0	2 603,8	1 933,0	0	1 933,0	OK	1 933,0	0,0	0,0	3 688,0
2016-03-06	712,4	0	0,0	1 933,0	1 220,6	0	1 220,6	Undestock	1 220,6	0,0	0,0	3 688,0
•••												
2016-03-12	135,6	0	0,0	0,0	-135,6	135,6	-477,4	Undestock	-477,4	0,0	0,0	3 688,0
	4 967,2	675,0	675,0			477,4						

Go To Planning Level | Insert delivery item | Save and recalculate | Schedule and save | Export To | Refresh

3. ábra: Részletes, ütemezési szintű (rövidtávú) csatorna adatok képernyőpélda.



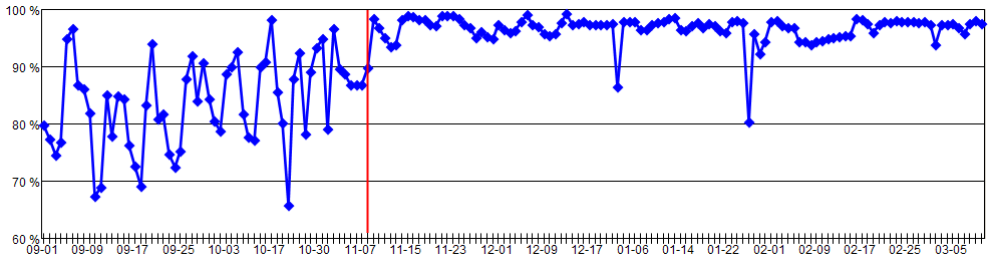
Channel data		SAP Details		Inventories	
Channel ID		No of SKUs	8	On hand customer	686.690
SAP Number		Past year usage		Carried forward	0.000
Description		Usage frequency		Consignement Supplier	0.000
Material Group		Restage	No	In transit	0.000
Supplier		ROP	0.000	On hand supplier	0.000
Customer		Lot size			

Last updated: 02-14 15:21

Need date	Latest order date	Order quantity	Demand quantity	Projected inventory	Customer status	Supplier status	Supplier production	Total projected inventory
2016-02-12	2016-02-05	718.000	0.000	1,404.690	OK	Calculated: OK	0.000	686.690
2016-02-19	2016-02-12	256.000	0.000	1,660.690	OK	Calculated: OK	0.000	686.690
•••								
2016-05-21	2016-05-14	0.000	484.068	408.452	OK	Calculated: OK	0.000	340.452
2016-06-18	2016-06-11	0.000	424.634	-16.182	Shortage	Calculated: Medium term shortage	0.000	-84.182
2016-07-23	2016-07-16	0.000	773.391	-789.573	Shortage	Overload by supplier	0.000	-857.573
2016-08-20	2016-08-13	0.000	288.885	-1,078.458	Shortage	OK by supplier	0.000	-1,146.458
	SUM	1,068.000	2,833.148					

Go to Scheduling level | Go to | Export to | Refresh

4. ábra: Részletes, tervezési szintű (középtávú) csatorna adatok képernyőpélda.



5. ábra: Egy komponens kiszolgálási szintje a Logisztikai Platform (a piros vonal által jelzett) bevezetése előtt és után.

2.2. Adattípus-modellezés heterogén adatforrások strukturális elemzésének és összehasonlításának elősegítésére

2.2.1. Bevezetés és irodalmi áttekintés

A vállalatok hálózatai a szervezeti heterogenitásuk, és a folyamatok ebből eredő átláthatatlansága miatt, gyakran olyan akadályokba ütköznek, amelyek gátolják ezen vállalatok lehetőségeit és üzleti kiteljesedésüket. A több-résztvevős logisztikai hálózatok sem mások ilyen szempontból, amelyeknél a következő említésre méltó, fejlődést akadályozó problémák merülnek fel:

- Habár az interoperabilitás támogatása végett, hálózatszerte szabványos implementációk készülhetnek, ezen szabványok helyenkénti értelmezése vagy éppen a követelményekkel való tényszerű megfeleltetése már eltéréseket mutathat. Az eltérésekért nem mindig lehet a fegyelem hiányát vagy a félreértéseket felelőssé tenni – gyakran a vállalaton belüli adatkezelési és jelentési gyakorlat (beleértve az egyenetlen kapcsolattartást az üzemi szinttel) nem feleltethető meg egy közös specifikációnak.
- Az adatmegosztással kapcsolatos kockázatokból eredő aggályok is jelentősen korlátozhatják, hogy mi kerül végül megosztásra a hálózatban. Bizonyos esetekben a már létező kollaborációs üzleti modellek eleve csak korlátozott átláthatóságot követelnek meg, hogy ezáltal vonzóbbá tegyék a részvételt az újonnan csatlakozó partnerek számára. Egy másik gyakori korlátozó minta az adatmegosztások látszólagos egyszerűsége – gyakran szándékosan és tartósan egyszerű megoldásként jelennek meg, hogy ezáltal kedvezőbb első benyomást kelthessenek, miközben nem foglalkoznak olyan komplex kockázatokkal, mint a masszív, töredékes információgyűjtés (adatbányászat) vagy logikai-inferencia támadások [25].
- Még ha az adat valahol jelen is van a hálózatban, az adat hozzáférhetőségének helye vagy időablaka, részletessége vagy absztrakciós szintje alkalmatlanná tehetik azt az átláthatóság megvalósíthatóságának szemszögéből. A logisztikai hálózatok esetén például a szállítmányozási információk lemaradhatnak a tényleges anyagáram mögött, csorbítva ezáltal a működési döntések hatásosságát.

Néhány ilyen akadály legyőzhető anélkül, hogy ehhez a vállalatok hozzáállásán vagy üzleti modelljén változtatni kellene, csupán azáltal, hogy a hálózatban már meglévő és azonnal rendelkezésre álló, de gyakran szegényesen strukturált adatok kiaknázásra kerülnek.

Egy széles körben alkalmazott megközelítés az adatok (a World Wide Web Konzorcium, W3C, [74] által definiált) XML formátumban történő leírása, ami

segíti az ezzel kódolt dokumentumok mind az ember, mind a számítógép által történő elolvashatóságát. A rendszerkomponensek közötti, rendszerek közötti vagy éppen a vállalatok közötti XML-alapú adatátvitel gyakori működési mód és követelmény a jelenlegi, modern számítógépes és üzlet-üzlet kapcsolatokat igénylő világban.

Az XML-alapú adatfeldolgozás rendkívül fontos ezekben a környezetekben, ezért számtalan funkcionális (program)nyelv és keretrendszer készült az elmúlt években a feladat kezelésére. A legismertebbek az XML-t célzó eszközök közül az olyan lekérdező nyelvek, mint az XQuery [76] és az XPath [71], valamint az ún. Bővíthető Stíluslap-transzformációs nyelv (Extensible Stylesheet Transformation, XSLT) [72], amely utóbbi XML-dokumentumok átalakítási folyamatát írja le deklaratív formában. Sajnos az XML formátum, önmagában és az említett társ-technológiákon keresztül, nem közvetíti vagy kezeli azon objektumok szemantikáját, amelyet az XML-alapú adat meghatároz és megtestesít.

Az XML-dokumentumok jól formázottsági követelményén túl számtalan strukturális, szemantikus és tartalomfüggő definíció hozzárendelésére van mód. Az effajta definíciók előfordulhatnak az XML-dokumentumba közvetlenül beágyazva, mint pl. a Dokumentumtípus-definíció (Document Type Definition, DTD) [70]. A DTD hátránya, hogy nem ad lehetőséget az elemek és attribútumok típusának megadására; minden tartalmat szöveges típusként kell értelmezni. Később a W3C egy nagyobb kifejezőerejű definíciót dolgozott ki az XML Sémaleíró (XML Schema Description, XSD) specifikáció formájában [75]. Néhány szélsőséges esettől eltekintve az XSD lehetővé teszi a fejlesztőnek az XML-dokumentum részletes struktúrájának, valamint szükséges (vagy tiltott) elemeinek és adattípusainak definiálását. Megemlíthető még a RELAX NG [53], ami egy hasonló séma-leíró nyelv és az XML-dokumentumok definiálására és ellenőrzésére szolgál (bár nem annyira elterjedt).

Egy közös tulajdonsága az említett a feldolgozó nyelveknek (XQuery, XPath, XSLT) és keretrendszereknek, hogy nem használják ki a rendelkezésre álló, a feldolgozás alatt lévő dokumentumhoz kapcsolódó (DTD vagy XSD formában adott) strukturális típusinformációkat, emiatt működés közben vagy hibára futnak, vagy nem adnak vissza eredményt, ha egy nem-illeszkedő dokumentum-struktúrával találják magukat szembe. Ezen hátrány leküzdésére néhány bővített lekérdező nyelv és programkönyvtár született.

Az egyik figyelemre méltó példa az XDuce keretrendszer [33], ami egy statikusan tipizált[3] XML feldolgozó programnyelvet biztosít és támogatja a legtöbb funkcionális programozási szerkezetet, úgymint a mintaillesztést és dinamikus típusellenőrzést. A keretrendszer elméleti alapjaiban a reguláris fa-automatára támaszkodik, továbbá (ritkaságként) a nyelv formális definíciója és a típus-biztonsá-

[3] Az adattípus-információk már fordítási időben kötelezően rendelkezésre állnak.

gosságának bizonyítéka is megtalálható a fenti hivatkozásban. Egy másik példa az XMλ keretrendszer [47], amely szintén statikusan tipizált nyelvet használ típus-inferenciával megtámogatva. A nyelv maga polimorfikus[4], magasabbrendű és támogatja a mintaillesztést is. Hátránya az olyen nyelveknek, mint az XDuce és XMλ, hogy látszólag csak DTD-vel rendelkező XML-dokumentumokon hajlandók dolgozni, miközben figyelmen kívül hagyják az XML-dokumentum elemeihez rendelt attribútumokat.

Az XML-típusrendszerek sajnos nem támogatják a modern programozási nyelvekben előforduló típus-polimorfizmust. Az igény arra, hogy például a jól ismert Egyszerű Objektum-hozzáférés Protokoll (Simple Object Access Protocol, SOAP) [73] az általa szállított dokumentum sémájával paraméterezhető legyen, az XML-séma számtalan, akadémiai szintű bővítési kísérletének adott teret. Erre egy példa a XML-séma annotáció-alapú bővítésére épülő megoldás [32].

Az XML-feldolgozó nyelvek számára hasznos lehet az ún. sémaegyszerűsítés és kanonizálás. Az XML-normálforma létrehozása, hasonlóan a relációs adatbázisok normálformáihoz, egy lehetséges irány [22]. Alternatívaként lehetőség van a sémadefiníció elemeinek egyfajta közös struktúrán alapuló, algoritmikusan egyszerűsített reprezentációjára [27].

Az XML-adatok átadása a vállalatok között egy fontos kihívást jelent. Hasonló dokumentum-sémák valószínűleg ugyanazt az adatot szándékoznak leírni az XML adatcsere két végletén. A sémák elemeinek összepárosításához hasonlósági mértékek lettek definiálva, amelyhez például felügyelt gépi tanulást lehet használni [35]. Hasonló lehetőséget nyújt a Cupid általános séma-összehasonlító eszköz [42], ami nyelvészeti-, elem- és struktúra-alapú párosítást végez, és kulcs vagy referencia jellegű korlátokat is használ.

Ha az XML lesz egy, az öröklést[5] is támogató típusrendszer alapja, módszerekre van szükség **két séma viszonyának meghatározásához**. Sajnos a gyakori eszközök és módszerek, mint a fentebb említett Cupid és XML-egyszerűsítő, algoritmusai nem bocsátják rendelkezésre ezt az információt.

2.2.2. Adattípusok modellezése és felettük végzett következtetések

A központi gyár (vevő) és az anyag-beszállítói közötti heterogén, nyers vagy alacsony szintű információs források és fogyasztók integrálásához a résztvevő információs rendszereknek az adatokat szintaktikailag és szemantikailag kompatibilis, (fél-)automatikus és számítási igényben hatékony formában kell kezelniük. Ez

[4]Bizonyos transzformációkat lehetőség van absztraktabb módon, a konkrét résztvevő adattípusok megadása nélkül definiálni.

[5]Öröklés: az adattípusok más adattípusokra épülése és kibővülése újabb elemekkel vagy korlátozásokkal.

például a közös szintaktikai alapot jelentő XML-leírással és a közös szemantikai alapot definiáló XML-sémával biztosítható.

2. tézis: Egy adat-interoperabilitást igénylő feladat megoldásakor az adatforrások és -fogyasztók XML-sémájú leírásait egy új, ún. kanonikus reprezentációra kell átalakítani, amely

1. **egy irányított, potenciálisan köröket tartalmazó gráf, ahol egy levélcsomópont egy konkrét adattípust vagy más csomópontok nevesített tárolóját reprezentálja, és minkét csomópont-fajta meghatározza a tartalma számosságát (kardinalitását) is; és**
2. **lecsökkenti a leírás alapjául szolgáló sémában megadott adattípusokat a gyártásban és logisztikában szokásos szűkebb készletre (pl., csak szám, szöveg, időbélyeg megengedett).**

Az adott kanonikus leírás birtokában három új algoritmus segítségével

1. **meghatározható két séma viszonya (összehasonlítása), úgymint azonosság, inkompatibilitás, vagy hogy az egyik a másik bővítése;**
2. **kigenerálható a két séma egyesített reprezentációja (uniója) szintén kanonikus leírás formában; illetve**
3. **kigenerálható a két séma közös része (interszekciója) kanonikus leírás formában**

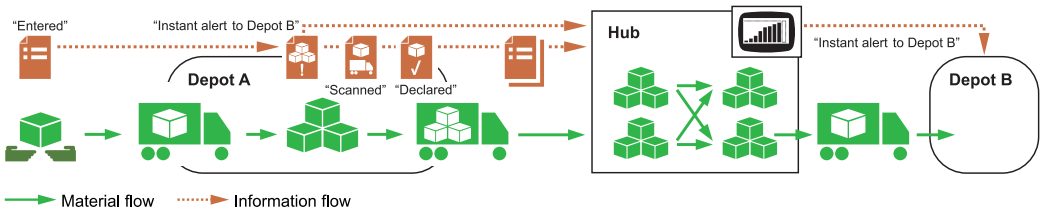
egy ún. koordinált, iteratív-leereszkedő algoritmussal, ami a két gráfon párhuzamosan haladva összepárosítja a csomópontokat (név vagy egyéb, annotációval megadott szemantikus információk alapján), összehasonlítja a csomópontok egyéb paramétereit és aggregálja az egyszerűsített viszonyaikat, ami végül a két gráf átfogó viszonyát eredményezi (egyforma, inkompatibilis, vagy az egyik a másik bővítése) és/vagy felépíti a két sémából származtatott közös vagy egyesített adattípus kanonikus reprezentációját.

Az algoritmusok eredménye a forrás(ok) és fogyasztó(k) közötti adat- vagy esemény-integrációs esetek ellenőrzésére és a (lehetséges) együttműködés szintjének meghatározására használandó, illetve az eredmény javaslattal szolgál az adat- vagy esemény definíciók kiigazításához a jobb elvi illeszkedés érdekében.

Kapcsolódó elsődleges publikációk: [3], [10], [15]
Kapcsolódó további publikációk: [8], [16], [19]

2.2.3. Eredmények alkalmazása

A 2., 3. és 4. tézisek fő alkalmazási területe az **ADVANCE Adatfolyam-Motor (Flow Engine)**, ahol az ADVANCE[6] azon projekt rövidítése, amelynek részeként a javasolt elméleti megoldások és az abból kifejlesztett szoftver próbahasználatba került egy országos logisztikai hálózatban az Egyesült Királyság területén. A hálózat szerkezetileg egy központi raktározó és küldemény-átcsoportosító telephelyből (hub) és több kiválasztott, helyi be- és kiszállító alvállalkozóból állt (depot, spoke), amelyek a bevezetett új rendszerek és funkcionalitások hatására, lépésről-lépésre, tovább tökéletesedtek. Ez az ún. **küllős hálózati szerveződés (hub-and-spoke)** kihasználja azt a hatékonyság-növekedést lehetővé tevő felismerést, hogy az országban lévő különféle feladók csomagjait ugyanarra célterületre kiszállítani sokkal hatékonyabb, ha azt egy dedikált alvállalkozó végzi a járműveivel, mintha minden alvállalkozó saját maga próbálná leszállítani a csomagokat egyenként vagy kis kötegekben. A csomagok és a hozzájuk kapcsolódó (digitális és papíralapú) információk áramlását egy ilyen típusú logisztikai hálózatban a 6. ábra szemlélteti.



6. ábra: Tipikus anyag- és információáramlás elemei egy küllős jellegű logisztikai hálózatban.

Minden egyes postai körzetet (irányítószám-tartományt) kiszolgáló alvállalkozó (depot) összegyűjti a területéről kiinduló csomagokat, majd kamionokra rakva naponta elszállítja azokat a központi raktárba (hub). Ott a csomagokat kirakodják és szétosztják célkörzetek szerint, majd a kamionokat feltöltik olyan csomagokkal, amelyeket arra a körzetre címeztek meg, ahonnan a kamionok jöttek. Ezzel a folyamattal, különösen annak információ-megosztási oldalával kapcsolatban több probléma merült fel korábban:

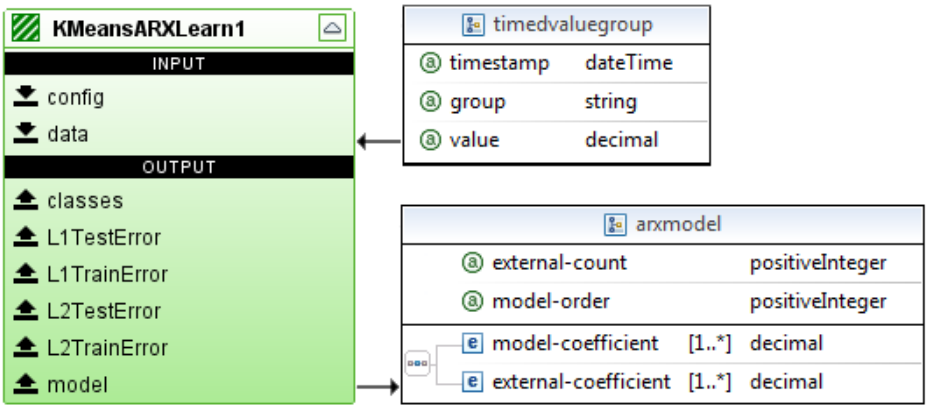
- az alvállalkozók heti szinten változhatnak, megnehezítve az információ-követést,
- az alvállalkozók IT-rendszerei gyakran nem azt a szótárat (fogalmi rendszert) használták, mint a központi raktár (hub) IT-rendszere,

[6]EU FP7, 257398 számú kutatási támogatás, „ADVANCE – Fejlett előrejelzés-elemzés-alapú döntéstámogató motor a logisztika számára (Advanced predictive-analysis-based decision-support engine for logistics").

- a csomagok digitális állapotának követése gyakran nem valós időben történt (főleg az engedékeny, korábbról származó szerződések miatt), alvállalkozónként más-más volt az időbeli lefutása, gyakran élt az automatikus leolvasás (vonalkód) helyett kézi adatbevitellel, és
- a csomagokkal kapcsolatos állapotváltozások, riasztások és értesítések annyi adatot generáltak hálózat szerte, hogy annak egységes kezelése hagyományos módszerekkel lehetetlenné vált.

A projekt célul tűzte ki ezen problémák megoldását egy új rendszer bevezetésével, amelyet **ADVANCE Adatfolyam-Motor**nak (Flow Engine) neveztem el. A Motor a disszertációban bemutatott új tudományos eredmények felhasználásával készült, valamint platformot biztosított az ADVANCE projekt más részein kidolgozott megoldások, úgymint gépi tanulás, folyamat-optimalizálás és valósidejű követés számára.

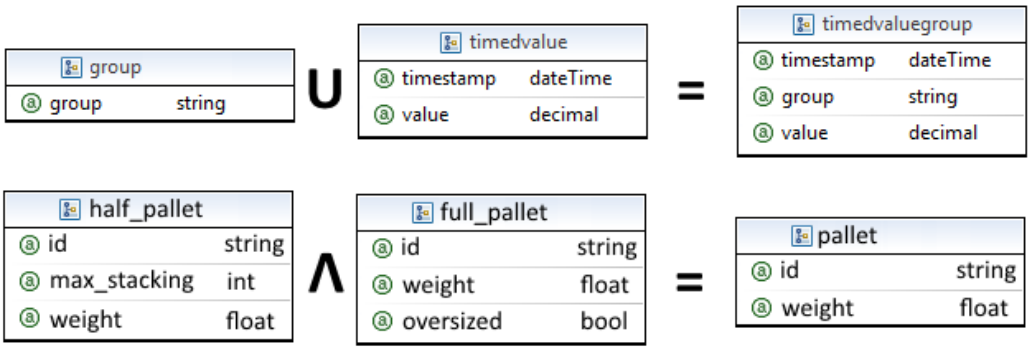
Az ADVANCE Adatfolyam-Motor (Flow Engine) részeként szükség volt a feldolgozó egységek (blokkok) be- és kimenetei adattípusainak definiálására. A gyakorlatban ez a Motorhoz tartozó folyamszerkesztő alkalmazásban (lásd 11. ábra) rövid, szöveges formában, pl. `domén:típusnév`-ként jelenik meg, ami mögött a definíciót egy, a Motor által működtetett típusrendszer-kezelő a `típusnév.xsd` fájlból olvas be. Erre példa az egyik gépi tanulásra szolgáló blokk, `KMeansARXLearn1`[7], amely bemenetként egy `timedvaluegroup` típusú adat-hármast (időbélyeg, érték, csoport) fogad, és kimenetként a megtanult modellt egy összetett adatszerkezetben (`arxmodel`) bocsátja rendelkezésre (7. ábra).



7. ábra: Példa blokk egyedi (és összetett) be- és kimeneti adatstruktúrákkal.

[7]KMeans: K darab középpontot tartalmazó csoportosítás; ARX: autoregresszív modell külső behatásokkal – olyan lineáris modell, ami korábbi értékek, sztochasztikus elemek és külső hatások alapján jelzi előre az értékeket.

Amíg az erősen típusos programozási nyelvek az adattípusok összehasonlításakor az adattípusok által explicit deklarált viszonyokra támaszkodnak, a 2. tézis strukturális információkat használ az adattípusok közötti viszony meghatározásához, illetve az átfogó (egyesített, unió) vagy közös strukturális rész (metszet) generálásához, ahogy azt a 8. ábra példái szemléltetik:



8. ábra: Példa unió, illetve metszet generálására két adattípusból kiindulva.

Ha az egyes adattípusok összetettek, ugyanazon unió/metszet-képzés történik a struktúra minden szintjén rekurzívan.

Az ADVANCE projekt témaköre, illetve az ontológiákkal kapcsolatos általános összetettség okozta nehézségek miatt a 2. tézisben leírt algoritmus az adattípusokban leírt, a kanonikus reprezentációból eredő ún. képesség-leírókhoz kapcsolódó szemantikai elv-párosítást csak korlátozottan valósítja meg azáltal, hogy a XSD-kben csak az alternatív elnevezések listájának annotációját kezeli, illetve maga az algoritmus lehetőséget biztosít a programozó számára[8] az összehasonlításkor egy egyedi programkód végrehajtására, majd annak eredményét figyelembe veszi a végső típusviszony kiszámításakor.

[8]Ez a funkcionalitás nem érhető el a Motor vizuális szerkesztőjéből, mivel a Motor Java nyelven íródott és a Motor szintjén, Java-ban kell leprogramozni a kívánt egyedi funkciót.

2.3. Adattípusok validálása és inferenciája információcsere-hálózatokban

2.3.1. Bevezetés és irodalmi áttekintés

Amikor a beszállítók saját információs rendszereiből származó heterogén adatokkal kell dolgozni, az adatok lekérdezéséhez és feldolgozásához elvárható minimális forma vagy struktúra megkövetelése gyakran nem lehetséges vagy nem kifizetődő a kezdetekben. Azonban gyakran elérhetők bővebb információtartalommal rendelkező adatforrások, amelyeket aztán olyan feldolgozó lépéseken kell átvezetni, amelyek csupán minimális strukturális követelményeket támasztanak az adatforrásaikkal szemben. A minimális és a gyakorlati struktúra közötti különbséget kovarianciának nevezik[9].

Ahol hagyományos, például SOAP alapú, webszolgáltatások kerülnek felhasználásra, a konkrét web-adatforrás típusdefiníciója gyakran rosszul igazodik a feldolgozásban résztvevő lépések bemeneti adattípus-definícióihoz. Emiatt gyakran költséges vagy kézi előzetes adatátalakításra van szükség, vagy magát a feldolgozó logikát kell újraépíteni és hozzáigazítani az adatforrás konkrét adattípus-definícióihoz. Azonban, ha az adatforrás valamilyen eseményt reprezentál, az adatok időbelisége lehetővé teszi olyan feldolgozó lépések létrehozását, amelyek nem igazán törődnek a konkrét adatszerkezettel, hanem magát az adat időbeli elérhetőségét használják fel. Ezek az általános feldolgozó lépések ún. parametrikus típusdefiníciókat használnak, így amikor a feldolgozási hálózat felépül, a résztvevő konkrét adattípusok a hálózat más részeiből lesznek származtatva.

Másfelől, a feldolgozás általános, gyűjtő/konténer típusokat (úgymint listák, és asszociatív tömböket) is használhat, valamint maga a SOAP üzenet is tekinthető egyfajta kompozit/gyűjtőtípusnak egy másik, konkrétabb adattípust átfogva, mint pl. `SOAP<KészletInfo>`, ahol a SOAP boríték (SOAP envelope) a belső, `KészletInfo`-val van paraméterezve. Ez lehetővé teszi azon feldolgozó lépéseknek, amelyeket csak a külső boríték típusa érdekel, hogy figyelmen kívül hagyják a belső adattípust. Így a lépések, gyakorlatilag, képesek lesznek bármilyen tartalmú SOAP üzenettel elboldogulni. A programozási nyelvekben ezt parametrikus polimorfizmusnak nevezik és a fenti példában `SOAP<T>`-vel lehetne jelölni, ahol a `T` ún. típusváltozó bármilyen konkrét típusra hivatkozhat.

Ha az adatfolyam-hálózat állandó (nem parametrikus), konkrét adattípusokkal működik, a feldolgozó lépések közötti kapcsolatok ellenőrzése majdnem triviális a klasszikus Milner [48] algoritmussal, ami egyszerű egyenlőség-vizsgálatot használ a résztvevő adattípusok összehasonlításakor. Azonban kovariancia, típusparaméterezés és parametrikus polimorfizmus megléte esetén az egyenlőség meg-

[9] Hétköznapiasan, ha gyümölcsre várok, akkor elfogadok almát és narancsot is, hiszen mindkettő annak számít.

állapítása nem elég, vagy gyakran lehetetlen. Ezért egy heterogén adatfolyam-hálózat ellenőrzéséhez ún. **típusinferencia** (típus-kikövetkeztetés) algoritmusra van szükség, ami azáltal hitelesíti a hálózatot, hogy minden résztvevő komponensnek konkrét adattípust talál – vagyis a hálózat akkor helyes, ha létezik olyan konkrét típus-hozzárendelés, ami ellentmondásmentes –, vagy éppen ellentmondásra jut a résztvevő paraméteres és/vagy polimorfikus típusok illeszkedési vizsgálata során.

A típusinferencia a programozási nyelvek, típusok és úgy általában, a számítástudomány világának alaposan kikutatott területe. Milner jól ismert munkásságából [48] kiindulva, több száz típusinferencia-algoritmust javasoltak az elmúlt évtizedekben. Némelyikük az egyesítő-algoritmust egészítette ki, míg mások gráfokat használtak az inferencia során [56]. Ehhez kapcsolódóan, a tudományos szakirodalomban többféle típusrendszer vizsgálatát is elvégezték: nemstrukturális résztípus-képzés [57, 60, 61]; strukturális résztípus-képzés [58, 64]; virtuális típusok [34]; típus-kényszerítés (automatikus típusátalakítás, ún. coercions) [67]; korlátos típusok [54]; rekurzió [36, 38]; és polimorfikus típusok [38], csak hogy néhány említésre kerüljön. Sajnos a fenti szakirodalomban megtalálható megközelítések, a követelmények által támasztott tulajdonságok miatt, egyenként, önmagukban vagy részenként kombinálva sem használhatók a célterületen (logisztikai jellegű adatfolyamok esetén) a típusinferencia-követelmények átfogó kezelésére. Ezek az alábbi tulajdonságokból és elvárásokból tevődnek össze:

- résztípus-képzés: az adatforrás több részletet tartalmaz, mint szükséges;
- polimorfikus típusok: gyűjtő/konténer adattípusok használata;
- közös- (metszet) és átfogó-adattípusok (unió): az adattípusok strukturális kombinációja;
- rekurzió: fa-jellegű adattípusok kezelése;
- nincs típus-kényszerítés: kivéve az egész számok automatikus használata valós számot váró számításokban;
- nincs szükség függvény-típusokra: az eszköztár egy egyszerű, Egyesített Modellező Nyelvre (Unified Modeling Language, UML) épülő blokkokból és vezetékekből áll;
- nincs közvetlen strukturális szétbontás, a már említett egyszerű UML eszköztár miatt; és
- nincs változódefiníciós többértékűség[10]: a fordító kezeli az egyedi és egyértelmű típusváltozó címkézést.

[10] Amikor ugyanaz a típusváltozó címke, pl. `T`, többször szerepel a hálózatban, de nem feltétlenül jelöli ugyanazt a konkrét adattípust.

A legtöbb létező algoritmus egy konkrét típusrendszerhez (vagy programozási nyelvhez) van kötve, és olyan tulajdonságokkal rendelkeznek, amelyekre nincs szükség a központi vevő és beszállítói által használt IT-rendszerek közötti adatkoordináció megvalósításához. Egy másik probléma, hogy általában nem lehet csak úgy kiválogatni a szükséges részelemeket a létező algoritmusokból, elhagyva vagy „félrevezetve" a többi elemet, mivel ettől az algoritmusok vagy működésképtelenné vagy inkompatibilissé (logikailag ellentmondásossá) válnak a többi algoritmus elemeivel szemben. Harmadrészt, sok típusinferencia és típusrendszer kombinációja nehezen támogatja az ún. **általános koordinációs lépéseket**. Ezek olyan csomópontok a hálózatot leíró viszony-gráfban, amelyek általában nem tesznek kikötést a rajtuk átfolyó adatok típusára, de a bemenő és kimenő adattípusoknak illeszkedniük kell egymással egy konkrét inferencia esetben. Más körülmények között a feldolgozó lépés ko- és kontravariáns **típuskorlátot** állíthat a be- és kimeneteire; strukturális értelmezésben az alsó korlát (kovariancia) az adattípuson minimálisan elvárható tulajdonság-halmazt jelent, míg a felső korlát (kontravariancia) egy maximális halmazt definiál. Következésképpen, a megcélzott felhasználási területre, az alapoktól kiindulva, új, a létező megoldások által inspirált szerkezeti elemeket is tartalmazó, típusinferencia-algoritmus megtervezése és megvalósítása szükséges.

2.3.2. Új típusinferencia és típusvalidáció megvalósítása egy cserélhető típusrendszer felett

A (központi) gyárban vagy az anyag-beszállítóival létesített adat-kapcsolatban létrehozandó adat- és esemény-feldolgozási láncban a különféle lépések bemenetét olyan formára kell alakítani, ami támogatja a napi-szintű működést és döntéshozatalt, figyelembe véve, hogy maga a feldolgozást végző logika összetétele is gyakori változtatásnak van kitéve. Ez megvalósítható olyan módon, hogy a komplikált adatfeldolgozó gráfban a különféle adatforrás–fogyasztó összekapcsolódások a gráf megváltoztatása esetén újrahitelesítésre kerülnek. A feldolgozó lépések közötti adatfolyamoknál a kapcsolat kovarianciáját is figyelembe kell venni, illetve az általánosabb, ún. generikus koordinációt megvalósító lépéseket is támogatni kell.

3. tézis: Az információátadásban résztvevő, kanonikusan reprezentált adattípusok és a viszonyaikat meghatározó szolgáltatás felhasználásával egy új adattípus-inferencia algoritmus lett kifejlesztve, amit a végrehajtási lépések közötti adatkapcsolatok kiértékelésére és ellenőrzésére használható, támogatva az alábbi adattípus-fajtákat használó feldolgozó lépéseket:

1. **konkrét adattípusok;**
2. **korlátos és korlát nélküli ún. generikus adattípusok; és**

3. **egymásba ágyazott, tároló-jellegű adattípusok, amelyek az (1)-ben és (2)-ben megjelölt típusokból épülnek fel.**

Az algoritmus a résztvevő feldolgozó lépések be- és kimeneteiből származtatott adattípus-definíciók párjainak listáján dolgozik, és iteratív formában

a. **ellenőrzi, hogy a konkrét adattípussal rendelkező párok kovariáns viszonyban vannak-e;**
b. **ellenőrzi, hogy az összetett (egymásba ágyazott) adattípus-pároknál az átfogó külső adattípus illeszkedik-e, majd a belső adattípus-definíciókból új párokat formál és hozzáadja a fenti listához; és**
c. **összegyűjti a generikus adattípusok (típusváltozók) korlátait és ellenőrzi, hogy a kovariáns viszony fennáll-e minden felső és alsó korlát között.**

A típusinferencia-algoritmus kimenete egyfelől a teljes feldolgozó gráf ellenőrzésének sikeressége vagy sikertelensége, másfelől minden egyes adatkapcsolat konkrét vagy származtatott adattípusa, illetve magyarázata annak, hogy hol és miért inkompatibilisek bizonyos kapcsolatok (azáltal, hogy az eltérést a kanonikus leírás segítségével szemlélteti).

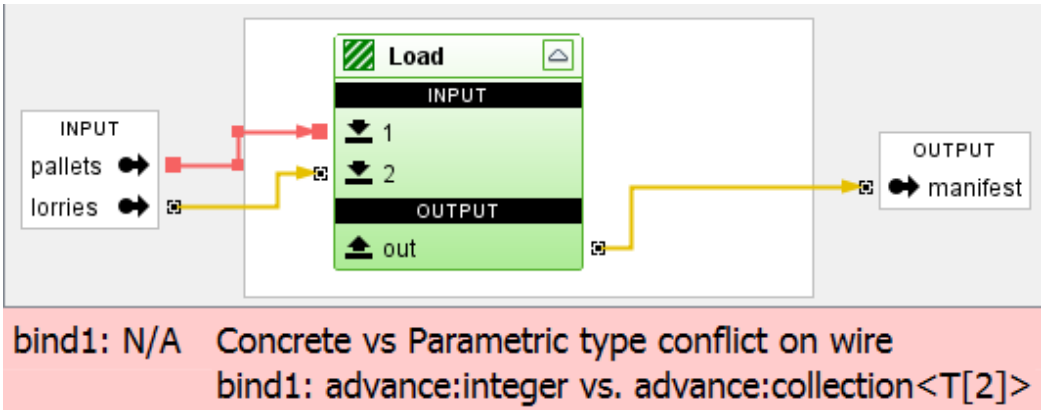
Kapcsolódó elsődleges publikációk: [3], [15]
Kapcsolódó további publikációk: [8], [19]

2.3.3. Eredmények alkalmazása

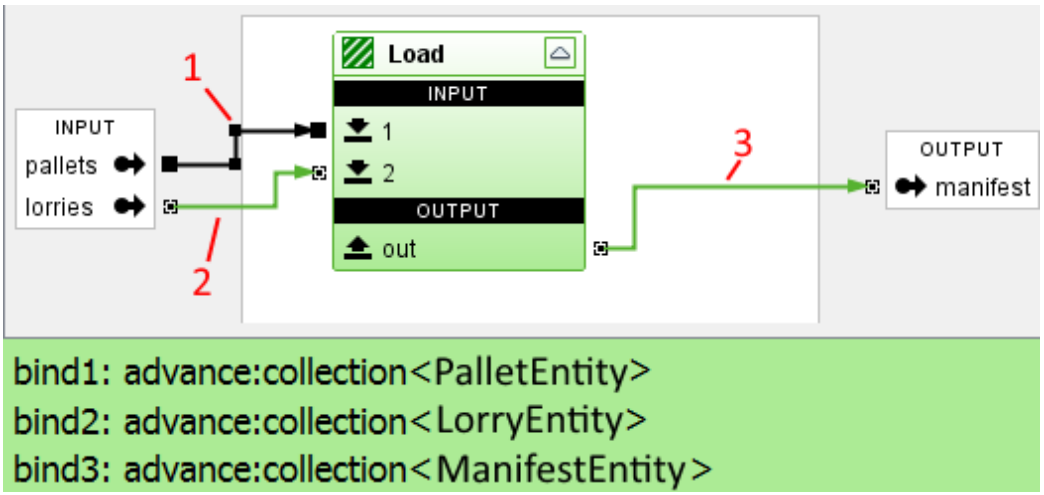
A 2.2.3 fejezetben bemutatott ADVANCE Adatfolyam-Motorban a (vizuálisan) megtervezett és összekapcsolt adatfolyam-hálózatokat ellenőrizni kell típushelyesség szempontjából, mielőtt azokat a Motor végrehajtaná. Az általános, blokkokat és vezetékeket használó szerkesztőkben nagyon könnyű az inkompatibilis be- és kimeneteket véletlenül vagy szándékosan összekötni, kivéve ha a szerkesztő nem ad visszajelzést a lehetséges hibáról a felhasználónak.

Például a Motor grafikus szerkesztője az (irányított) vezetékek piros színével, és kijelölésük esetén rövid hibaüzenettel értesíti a felhasználót az érintett adattípusok összeegyeztethetetlenségéről (9. ábra). Ebben a hálózati összeállításban a `Load` (felrakodás) feldolgozó blokk két bemenetet fogad el: a rendelkezésre álló raklapok és üres kamionok listáit. A blokk kimenete a szállítási jegyzékek listája, ami meghatározza, melyik raklap melyik kamionra került. A 9. ábrán a hálózat egyik külső bemenete, a `pallets` (raklapok) nem egy lista, hanem egy egyszerű szám (valószínűleg az elérhető raklapok darabszáma) a konkrét raklapok listájával szemben. Az ábra azt az esetet is demonstrálja, amikor egy blokk nem tesz megkötést az általa elfogadott adatok típusára, így gyakorlatilag bármivel össze lehet

azt kötni. A blokk egy általános, gyűjtemény jellegű, `T` típusparaméterrel ellátott adattípust vár, – amely a 2-es indexet kapta, mivel ugyanazon típusváltozó többször is előfordul a hálózatban, de nem feltétlenül jelöli ugyanazt a konkrét típust –, ahol a típusparaméter értéke végül a teljes hálózat helyes típus-meghatározása fogja szolgáltatni. A 10. ábra azt az esetet demonstrálja, amikor minden be- és kimenet helyesen van összekötve ebben az (egyszerű) példa-hálózatban.



9. ábra: Típus-inkompatibilitás hibajelzése rossz források összekapcsolása esetén.



10. ábra: Helyesen összekötött be- és kimenetek megjelenése, illetve visszajelzés a vezetékek típusinferencia által meghatározott konkrét adattípusairól.

2.4. Információcsere és -feldolgozás támogatása reaktív adatfolyamokkal

2.4.1. Bevezetés és irodalmi áttekintés

Modern ipari körülmények között nagy mennyiségű adatot kell átvinni és feldolgozni – többségében jelentős késleltetés, feltorlódás és adatvesztés nélkül. Ehhez olyan megoldást kell találni, ami erőforrás-hatékony, nagy sávszélességű és azonnali választ tesz lehetővé. A követelményeknek kiválóan megfelel az ún. **reaktív paradigma**, ahol az adatok (általában) toló-jelleggel [30] áramlanak a komponensek között.

A klasszikus, ún. **húzó-jellegű** adatfeldolgozásban a fogadó oldal kéri a következő adatot a küldő oldaltól szinkron módon, vagyis a fogadónak várnia kell, amíg a kért adat elérhetővé válik számára. Ennek hátránya, hogy a fogadó a várakozás alatt nem végezhet más feladatot, ami gyakran erőforrás-pazarláshoz vezet. Ezzel szemben a **toló-jellegű** működési mód azt jelenti, hogy a fogadó a beérkező adatra reagál, miközben a küldőnek nem kell foglalkoznia az adatátadás sikerességével (jelez és elfelejt, fire-and-forget). Különleges mechanizmusok, mint pl. pufferelés, alkalmazhatók az adatvesztések elkerülésére, tehermentesítve a hálózatot a szükségtelen számítási igény alól. Mindkét, toló- és húzó-jellegű adatfeldolgozás kifejezhető deklaratív formában, ami lehetővé teszi a feldolgozást segítő programkönyvtárnak vagy keretrendszernek, hogy **adaptív módon váltson a két megközelítés között**[11].

A **Reaktív vagy toló-jellegű programozás** [30] nem új paradigma, de csupán a fősodratú programozási nyelvek és programkönyvtárak friss fejlesztései tették alkalmazhatóvá a módszert a gyakorlatban. Egy jól ismert C# [28] nyelvű programkönyvtár az Rx.NET (Reactive Extensions for .NET) [41], ami lehetőséget nyúlt toló-jellegű műveletvégzések kifejezésére és összeláncolására ugyanazon programnyelvi szintű lekérdező szintaxis (Language Integrated Query, LINQ) [46] felhasználásával, amit általában a húzó-jellegű adatfeldolgozásra mindig is használtak. Ezzel szemben, az egyik legszélesebb körben használt programozási nyelv, a Java [55] szinte alig nyújt támogatást az ilyen toló-jellegű műveletvégzéshez. Egy lehetséges megoldás a FlumeJava [24] vagy Frappé [26] (amit eredetileg párhuzamos adatfeldolgozásra terveztek) programkönyvtár valamelyikének átalakítása, de ez nyilvánvalóan fáradságos (és nem túl hatékony) vállalkozás lenne. Ebből következik, hogy egy alapjaiban teljesen új, Java nyelven íródott reaktív keretrendszert kell megtervezni és kifejleszteni.

Gyakran a programozási nyelv szintjén elérhető egyszerűbb vagy kisebb algoritmusokat, feldolgozó logikákat nagyobb, ún. (feldolgozó) **blokkokba** szervezik a magasabb szintű programozási környezetekben (mint pl. a grafikus adatfolyam-

[11] lásd a szerző akarnokd.blogspot.hu/2016/03/operator-fusion-part-1.html blogbejegyzését.

tervezők) való felhasználás céljából. Ezek a blokkok a be- és kimeneteiket csatlakozási pontokkal, ún. **portok**kal definiálják, hasonlóan, ahogy a függvényparamétereket és visszatérési értéket lehet a fősodratú programozási nyelvekben (lásd Java) megadni. A blokkok által biztosított magasabb szintű absztrakció az adatfolyam-hálózatok gazdaságosabb, fejlesztési időben gyorsabb megvalósítását teszik lehetővé. Több blokk-jellegű, adatfolyam-orientált nyelv és programkönyvtár létezik: pl. a LabView G [52] programozási nyelv és környezet, vagy egy viszonylag friss fejlesztés, a Feladat-párhuzamos Adatfolyam Keretrendszer (Task Parallel Library Dataflow Framework, TPL Dataflow) [66]. A TPL Dataflow a többi toló-jellegű megoldással ellentétben sajátos adatelosztási stratégiát használ: egy adatot egy adott pillanatban csak egy blokk birtokolhat, emiatt – gyakran igen kifinomult – tárgyalási protokollokra van szükség az egyes kapcsolódó blokkok között.

Habár ezek a blokk-alapú környezetek hatékonynak bizonyulhatnak a saját célterületükön és használati tartományukban, de minden bizonnyal nehezen tudnák teljesíteni a disszertációban tárgyalt, logisztikai-központú alkalmazási környezet által támasztott követelményeket.

2.4.2. Új keretrendszer információáramlás és -feldolgozás támogatására

A (központi) gyárban, illetve a vele kapcsolatban álló anyag-beszállítók közötti (dinamikus) adat- és eseményfeldolgozó hálózat létrehozásához és végrehajtásához egyszerű, deklaratív adatfolyam-leírásra van szükség, amely formában megadott adatfolyam-hálózatot később a számítógép által hatékonyan végrehajtható formába kell transzformálni.

4. tézis: Az adatfolyam-hálózat modellezésére egy új, XML-alapú deklaratív és egyszerűsített folyamatleíró formátum lett megalkotva, amely formátum csupán négy elemtípust tartalmaz: konstansok, blokkok, be- és kimeneti portok és az ezeket összekötő vezetékek. Az adatfolyam-hálózat által használt adattípusok kanonikus reprezentációjának birtokában, valamint a hálózat helyességének és a kapcsolati gráfjának típusinferenciával meghatározott pontos adattípusainak ismeretében az adott adatfolyam-leíró XML egy új fordító algoritmussal újszerű futásidejű reprezentációra transzformálandó. Az algoritmus bejárja az adatfolyam-leíró összekapcsolódási gráfját, példányosítja a végrehajtó blokkokat, kibontja az összetett blokkokat, beágyazza a konstansokat és összeköti a különféle blokkok be- és kimeneteit. Ez a reprezentáció összeépíthető, reaktív és adaptívan toló- vagy húzó-jellegű számítási- és végrehajtási modellel rendelkezik. A futásidejű környezet, ezt támogatandóan, felelős a be- és kimeneti portok között áramló adatok és események szállításáért és a feldolgozó blokkok által igényelt számítási erőforrások ha-

tékony kihasználásáért.

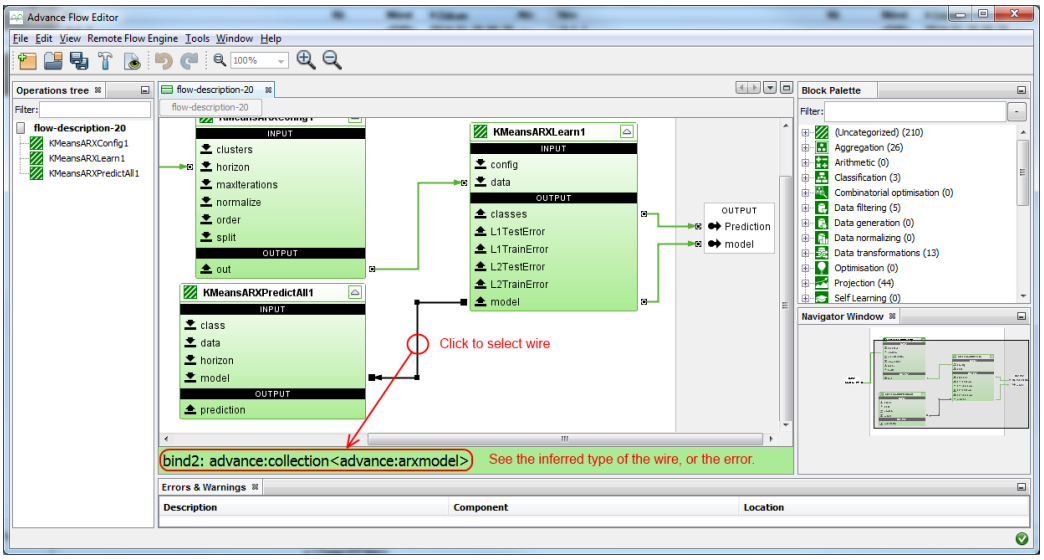
A reaktív jelleg és az átlátható adat- és esemény-interoperabilitás a szervezeti egység határain is túl kell, hogy terjedjen olyan módon, hogy a szolgáltatási végpontok elérésekor az adott hálózat külső be- és kimeneti adattípusainak kanonikus leírása is elérhető legyen magán az adat- és információcserét lehetővé tévő kommunikációs csatornákon túl. Ezen felül a szolgáltatást biztosító környezetnek a végpontokhoz való hozzáféréskor el kell végeznie a szükséges hitelesítési és engedélyezési ellenőrzéseket.

Kapcsolódó elsődleges publikációk: [3], [8], [16]
Kapcsolódó további publikációk: [1], [10], [15],

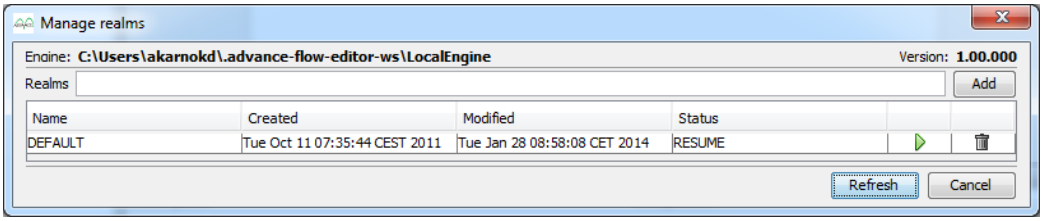
2.4.3. Eredmények alkalmazása

A 2.2.3 fejezetben bemutatott ADVANCE Adatfolyam-Motort egy XML-alapú adatfolyam-leíró nyelven lehet programozni, amelyek lényegében a lefordítandó és végrehajtandó adatfolyam-hálózat forráskódja. A hálózat leírásához az adatfolyam-leíró négy, egyszerű építőelemet biztosít: **konstansok**; **hagyományos blokkok**; **összetett blokkok**, amelyek al-hálózatokat tartalmazhatnak, és **összeköttetések** (avagy vezetékek) a blokkok egymással és/vagy konstansokkal való összekapcsolására. A hagyományos blokkokhoz hasonlóan az összetett blokkok is definiálhatnak be- és kimeneti paramétereket. Ez lehetővé teszi a hálózattervezőnek, hogy egyfajta újrahasználható al-hálózatokat készítsen vagy egyszerűen összefogja egy nagyobb hálózat elemeit könnyebben karbantartható logikai egységekbe. A hálózat-tervezőnek nem kötelező az adatfolyam-hálózatot XML- és szöveges formában megadnia, mivel az ADVANCE Adatfolyam-Motor rendelkezik egy erre a célra testreszabott, UML-stílusú grafikus szerkesztővel (11. ábra). Az Adatfolyam-Motor egy példánya több, ún. végrehajtó környezetet (realm) képes kezelni, amelyekben több, független adatfolyam-hálózat is futhat egyszerre. Új környezet hozzáadására, illetve a meglévő környezetek ellenőrzésére és kezelésére is lehetőség van (12. ábra). Miután egy hálózat tervezése befejeződött, az egyszerűen feltölthető az egyik végrehajtási környezetbe, ahol igény szerint megkezdődhet a végrehajtása (13. ábra). Az aktív adatfolyamok vizsgálatát és hibakeresését is támogatja a Motor. A 14. ábrán bemutatott példában egy rendkívül egyszerű hálózat nyomkövetése látható, amiben csupán egy nyomógomb (eseményforrás) és egy napló-megjelenítő komponens van összekötve egymással. A gomb minden egyes megnyomásakor a napló megjeleníti az esemény időpontját és a benne tárolt értéket. Ezzel egy időben a hibakereső képernyőn látható, hogy melyik végrehajtási környezet melyik komponensének melyik kimenete mikor és milyen eseményt generált. A gyakorlatban az alsó megjelenítő képernyő

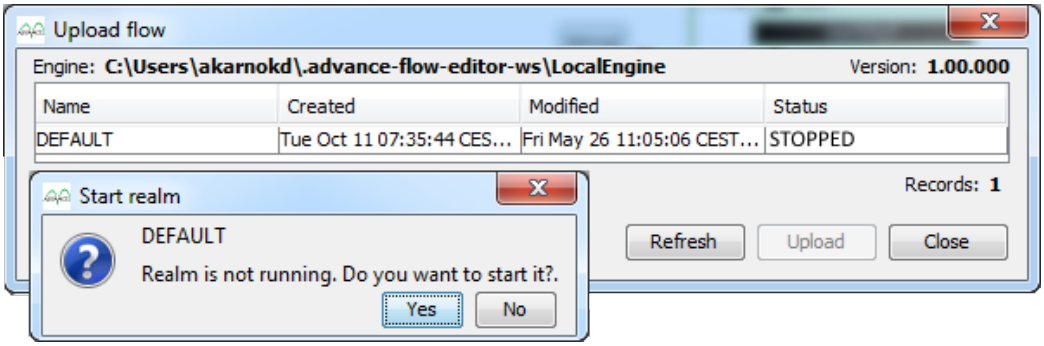
(Advance Block Visualization) nem látszik, mert a Motor által alapértelmezésben rendelkezésre álló legtöbb blokknak nincs vizuális be- vagy kimenete, de ettől függetlenül a hibakereső ablakban ugyanúgy megvizsgálhatók.



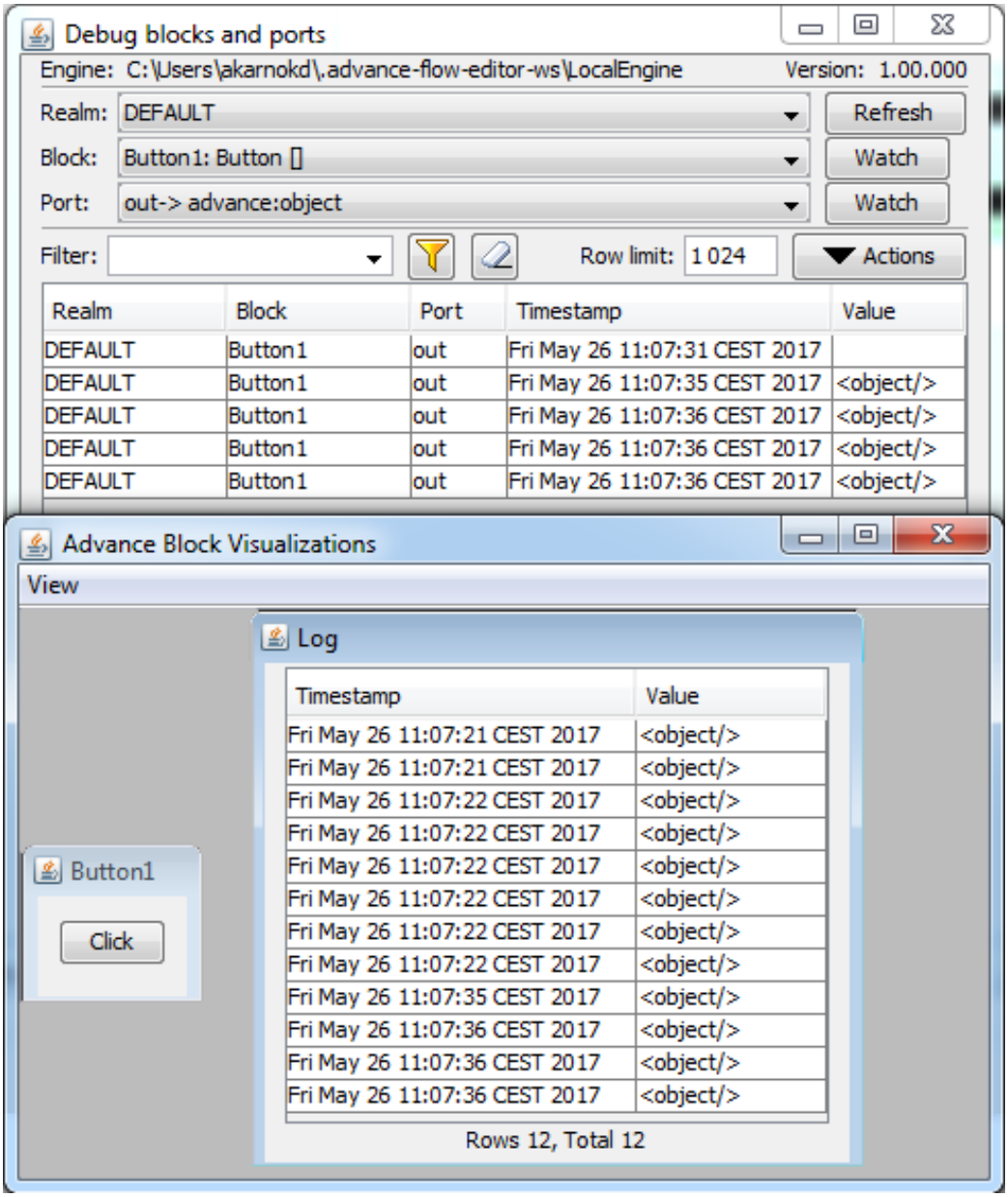
11. ábra: Az ADVANCE Adatfolyam-Motor grafikus szerkesztője.



12. ábra: A Motor egy példánya által kezelt futtató környezetek listája.



13. ábra: Adatfolyam-hálózat feltöltése és végrehajtása a Motor egyik futtató környezetében.



14. ábra: A Motor egyik futtató környezete eseményeinek megjelenítése és hibakeresése egy egyszerű gomb–eseménynapló hálózatban.

3. További kutatási irányok

A modern információ-technológia különösen értékes eredményekkel, technológiákkal szolgál a negyedik ipari forradalom számára, amely most már a felhő (Cloud) és a mindenfelé elérhető okos, számításra képes komponensek által nyújtott lehetőségekkel is élhet, mind a gyártásban, mind a logisztikában.

A fő kihívás, nevezetesen az offline és papíralapú anyagbeszerzés átalakítása fél-automatikus digitális folyamattá, ahogy azt az **1. tézis** demonstrálja, nem a végcél, csupán lépcsőfok egy ember által asszisztált automata logisztikai megoldás felé, amely a résztvevő vállalatok minden szintjéről érkező adatokat képes integrálni. Az integráció ilyen szintje és mértéke segíteni fog a napi jellegű anyagáram menedzselésében, miközben részletes információkat és eszköztárat bocsát rendelkezésre a problémák kézi elhárítására, ha ez az adott üzemi helyzetben szükségessé válik (ún. pilótafülke feladat-menedzsment, cockpit task management). Habár ezt az ötletet a vállalat-integrációs (Enterprise Integration) mozgalom már megfogalmazta, a különbség az, hogy a fő teljesítmény-indikátorok feljavítását alulról-felfelé (bottom-up) irányuló, egyedileg szabott modellek és rendszerek (mint pl. a **Logistics Platform**) tervezésével és implementálásával valósítja meg ahelyett, hogy minden résztvevő fölé egy ERP-t helyezne és a (komplex) problémát föntről-lefelé (top-down) haladva – gyakran szlogenekkel tűzdelt módon, a valóságról nyert (rész-)ismereteket esetleg koncepció jelleggel mellőzve, torzítva – próbálná megoldani.

A (digitális) szolgáltatóiparban ez az alulról-felfelé gondolkodásmód vezetett a **mikroszolgáltatások (microservices)** tervezési elvhez. Az alapötlet az üzleti logika vagy folyamat olyan szolgáltatásokból történő felépítése, amelyeket úgy terveztek (és korlátoztak), hogy kicsi, egyszerű és célirányos feladatokat hajtsanak végre. A korábbi megközelítések, mint pl. a SOAP-alapú webes szolgáltatások összjátékával (service-orchestration) vagy klasszikus vállalati szolgáltatásgyűjtősín (Enterprise Service Bus) megoldásokkal, már nem lehet gazdaságosan vagy elégségesen választ adni az olyan követelményekre, mint a folytonos adatáramlás (streaming) és (közel) valós-idejűség igénye.

A mikroszolgáltatás-alapú architektúrák egy fontos összetevője a reaktív programozási paradigma, amire a **4. tézis** is támaszkodik. Valójában a Reactive4Java-t övező kutatás és fejlesztési munka egyfajta mini programozási forradalmat vitt a mobil (Android) fejlesztési platformra azáltal, hogy a szerzett tapasztalatok az RxJava[12] nevű, kollaboratív, nyílt-forráskódú programkönyvtárban kerültek felhasználásra, amelynek a kódbázisa átlagosan 80%-ban jelen szerző hozzájárulásaiból tevődik össze[13]. Olyan társ-programkönyvtárak segítségével, amelyek

[12] https://github.com/ReactiveX/RxJava

[13] A `git blame` alapján, ez a átlagszám a kb. 65%-ban a korábbi verziókhoz való hozzájárulásokból és a kb. 95%-ban az új verzió (2.x) megtervezéséből és implementálásából áll.

webes / hálózati adatáramlást (streaming) tesznek lehetővé, a paradigma segít a szoftver / szolgáltatás fejlesztőinek abban, hogy az adatfeldolgozó logika deklaratív megfogalmazására koncentrálhassanak, miközben a programkönyvtárak és keretrendszerek lehetővé teszik a kifejlesztett adatfolyamok és üzleti logikák hatékonyabb végrehajtását (felhő-) erőforrás-felhasználás szemszögéből is. Az RxJava sikere (ami főleg abból ered, hogy széles körben elterjesztette és bizonyította a reaktív paradigma hasznosságát) a Reaktív-Folyamok (Reactive-Streams) nevű ipari-szabvány kialakulásához vezetett, amit mostanra várhatóan már minden jelentősebb keretrendszer- és programkönyvtár-szolgáltató felhasznált és támogat saját megoldásaiban, illetve kiindulási alapot jelent a jövőbeli alkalmazások tervezésnél. Ezen a vonulat mentén az **1. tézis** új megközelítési módja (minimális adatmodell és protokoll) kényelmesen illeszkedik ebbe a világba és maga a megoldás viszonylag könnyen átalakítható ilyen formára.

A mikroszolgáltatás tervezéséhez is szükségessé válik olyan megoldások létrehozása, amelyek képesek kezelni az elosztott és reaktív adatfolyamokban felmerülő adattípusok viszonyait. Az elosztottság miatt a konkrét adattípusok többé nem találhatók meg egy helyen, hanem ezen információt rendelkezésre kell bocsátani vagy éppen magával az adatárammal együtt kell elküldeni a fogadó irányába valamilyen módon. Ezen felül a SOAP típusdefiníció rugalmatlansága korlátozza a szolgáltatások szabad igénybevételét, mivel nem támogatja a kovarianciát (vagyis olyan források kezelését, amelyek szerkezete több elemet tartalmaz, mint amennyire szüksége van az igénybe vevőnek). Mivel a legtöbb ipari adat rekordokból vagy statikus adatstruktúrákból áll, a struktúra-alapú típusmegadás, következésképpen a **2. tézis** eredményei által lehetővé tett strukturális összehasonlítás elengedhetetlenné válik az új és modern megoldások tervezése során. (Megjegyzendő, hogy néhány aktuális új programozási nyelv, mint pl. a Pony[14] nyelv, alapjaiban a strukturális adattípusok szükségességének figyelembe vételével lett megtervezve, így megerősítve a tézisben szereplő elgondolás hasznosságát.)

Egy strukturális típusrendszer önmagában nem túl hasznos, működéséhez szükség van egy alkalmazói környezetre is. Azon túl, hogy könnyebbé vált a reaktív rendszerek építése, az igény a dinamikusan létrehozandó, szoftverfejlesztőt nem vagy csak alig igénylő adatfolyamokra továbbra is jelen van. A **3.** és **4. tézisre** épülő **ADVANCE Flow Engine**-hez hasonlóan a különféle felhő-alapú megoldásokat nyújtó vállalatok is biztosítanak vizuális szerkesztő környezetet a megvalósítandó, teljesen reaktív adatfolyamok megtervezéséhez és üzembe állításához, mint például a Spring Cloud Dataflow Server (lásd 24.2 fejezet 11. ábra). Az ilyen szerkesztőknek természetes igénye a típusinferencia megléte, mivel az alacsonyabban elhelyezkedő programnyelvek (mint a Java) fordítói nem képesek besegíteni a képernyőn megjelenített adatfolyam-hálózat helyességének ellenőr-

[14] https://www.ponylang.org/

zésébe. (Ugyan jelen szerző jelentősen hozzájárult a fent említett szoftver alapjául szolgáló Reactor-Core[15] reaktív programkönyvtárhoz, nem állítható, hogy a Spring Cloud Dataflow Server-t támogató típusrendszer és típusinferencia logikát inspirálta-e valamilyen módon az említett ADVANCE Flow Engine.)

Általánosságban kiderült, hogy az Ipar 4.0 által fontossá vált (vagy éppen elvárt) megoldások új és jobb programozási paradigmák létrejöttéhez vezettek a számítási oldalon is, így állítva a reaktív programozási paradigmát középpontba, ami lehetővé teszi az erős alapokra épített szolgáltatások, megoldások és iparág-specifikus üzleti logikák megvalósítását. Azonban a reaktív adatfolyamokkal kapcsolatos kutatások ezzel nem értek véget, mivel az ipari alkalmazásokból származó tapasztalatok új problémák megoldására sarkallnak. Például, az adatáramlás sebessége túlterhelhet egyes adatfeldolgozó elemeket (vagy lépéseket), amely túlzott (számítási) erőforrás használathoz és instabilitáshoz vezethet. A **4. tézis** megoldása a bejövő adatok mintavételezése vagy ún. lapozása volt, mely tudatos adat-eldobást vagy speciális protokoll-tervezést igényel. A problémát, amit **ellennyomásnak (backpressure)** neveznek, végül a Reactive-Streams Alkalmazás-Programozói Felület (Application Programming Interface, API) specifikációja oldotta meg úgy, hogy célzott szabályokkal és elemekkel bővítette az eredeti reaktív API dizájnt[16]. Következésképpen, a jelen szerző az alábbi, további kutatást igénylő problémákat / területeket azonosítja:

- Művelet-fúzió (operator fusion): a reaktív adatfolyamok még hatékonyabb végrehajtása, kiaknázva a szekvenciális lépések tulajdonságait[17].
- Párhuzamosított adatfolyamok: több végrehajtó-szálat tartalmazó processzor hardver kiaknázása[17].
- Reaktív távoli vagy helyi programközi kommunikáció: jobb izoláció, sokkal gördülékenyebb együttműködés az elosztott komponensek között.
- Kétirányú reaktív adatfolyamok.
- Adat-, erőforrás- és számításfüggő életciklusok figyelembe vétele.
- Számítási környezetet figyelembe vevő adatfolyamok.
- Adattartalom nélküli (ún. jelző) vagy egyszerre több adatot átvivő (vektor) adatfolyamok.
- Megbízható adatszállítás többlépéses adatfolyamoknál.
- Késleltetés-érzékeny adatfolyamok és garantált idejű adatszállítás vagy reakciók.

[15] https://github.com/reactor/reactor-core

[16] lásd Reactive-Streams célok.

[17] Az ehhez tartozó fejlesztés és kiértékelés már jelenleg is zajlik az RxJava 2.x és Reactor-Core 3.x programkönyvtárakban.

- Reaktív programkönyvtár, mint szolgáltatás (Reactive-Library-as-a-service): összeválogatható funkcionalitású, automatikusan generált reaktív programkönyvtár.

A disszertációban bemutatott új tudományos eredmények józan és alapvető kiindulási pontot biztosítanak ezen kihívások jövőbeli leküzdésére.

4. Köszönetnyilvánítás

A kutatást az Országos Tudományos Kutatási Alap (OTKA) 113038. számú és a GINOP-2.3.2-15-2016-00002 számú „Ipar 4.0 kutatási és innovációs kiválósági központ” című pályázatai támogatták.

Publikációk

Web-of-Science lektorált folyóiratcikkek

[1] Monostori, L., Ilie-Zudor, E., Kemény, Z., Szathmári, M., and **Karnok**, **D**. Increased transparency within and beyond organizational borders by novel identifier-based services for enterprises of different size. *CIRP Annals – Manufacturing Technology*, 2009, 58(1):417–420. (DOI: 10.1016/j.cirp.2009.03.086), IF: **1.603**, Cites: **5**.

[2] Monostori, L., Kádár, B., Pfeiffer, A., and **Karnok**, **D**. Solution approaches to real-time control of customized mass production. *CIRP Annals – Manufacturing Technology*, 2007, 56(1):431–434. (DOI: 10.1016/j.cirp.2007.05.103), IF: **0.779**, Cites: **20**.

[3] **Karnok**, **D**., Kemény, Z., Ilie-Zudor, E., and Monostori, L. Data type definition and handling for supporting interoperability across organizational borders. *Journal of Intelligent Manufacturing*, 2016, 2:167–185. (DOI: 10.1007/s10845-014-0884-9), IF: **1.142**, Cites: **2**.

[4] Váncza, J., Egri, P., and **Karnok**, **D**. Planning in concert: A logistics platform for production networks. *International Journal of Computer Integerated Manufacturing*, 2010, 23(4):297–307. (DOI: 10.1080/09511921003630092), IF: **0.553**, Cites: **8**.

Idegen nyelvű lektorált folyóiratcikkek

[5] Monostori, L., Váncza, J., Kis, T., Kádár, B., **Karnok**, **D**., Drótos, M., and Egri, P. Real-time, cooperative enterprises: Results of an industry – academia project. *Journal of Machine Manufacturing*, 2009, 49(E1):8–12.

[6] **Karnok**, **D**. and Monostori, L. Novel approaches for better transparency in production and supply chains. *Asian International Journal of Science and Technology in Production and Manufacturing Engineering*, 2009, 2(3):57–68.

Publikációk magyar nyelven

[7] Monostori, L., Váncza, J., Kis, T., Kádár, B., **Karnok**, **D**., Drótos, M., and Egri, P. Valós időben együttműködő vállalatok: egy ipari–akadémiai projekt eredményei. *Gépgyártás*, 2008, 48(4):11–15.

[8] **Karnok**, **D**. Átláthatóság a gyártásban és a logisztikában: egy reaktív megközelítés. *Gépgyártás*, 2015, 55(2):73–77.

Nemzetközi konferencia vagy egyéb cikkek

[9] Egri, P., **Karnok**, **D**., and Váncza, J. Information sharing in cooperative production networks. In: Monostori, L. (editor), *Preprints of the IFAC Workshop on Modelling, Management and Control*. Location: MTA SZTAKI, Budapest, Hungary, 14–16 November, 2007, pages 115–120. (ISBN: 978-963-311-366-0). Cites: **4**.

[10] Ilie-Zudor, E., Ekárt, A., Kemény, Z., **Karnok**, **D**., Buckingham, C., and Jardim-Goncalves, R. Information modelling and decision support in logistics networks. In *5th international conference on experiments process system modeling simulation optimization*. Location: Athens, Greece, 3–6 July, 2013, pages 279–286. (ISBN: 978-618-80527-1-0).

[11] Ilie-Zudor, E., Szathmári, M., Kemény, Z., **Karnok**, **D**., and Monostori, L. Identity-based, item-centric tracking platform for logistics applications. In *Proceedings of the International Workshop on Harbour, Maritime & Multimodal Logistics Modelling and Simulation (HMS2008)*. Location: Campora San Giovanni, Italy, 17–19 September, 2008, pages 10–19.

[12] Monostori, L., Ilie-Zudor, E., Kádár, B., **Karnok**, **D**., Pfeiffer, A., Kemény, Z., and Szathmári, M. Novel it solutions for increasing transparency in production and in supply chains. In: Spath, D., Ilg, R., and Krause, T. (editors), *ICPR 21, Proceedings of the 21st International Conference on Production Research*. Location: Fraunhofer IRB Verlag, Stuttgart, Germany, 7 July – 4 August, 2011, pages 1–6. (ISBN: 978-3-8396-0293-5).

[13] Monostori, L., Váncza, J., Kis, T., Erdős, G., **Karnok**, **D**., and Egri, P. Real-time, cooperative enterprises for customized mass production; challenges and solution approaches. In: Chung, M. J. and Misra, P. (editors), *Proceedings of the 17th IFAC World Congress*. Location: Seoul, South Korea, 6–11 July, 2008, pages 13851–13856. (ISBN: 978-1-605-60758-0).

[14] Monostori, L., Váncza, J., Kis, T., Kádár, B., **Karnok**, **D**., Drótos, M., and Egri, P. Novel it approaches and solutions towards real-time, cooperative enterprises; plenary paper. In *13th IFAC Symposium INCOM 2009, Control Problems in Manufacturing*. Location: Moscow, Russia, 3–5 June, 2009, pages 724–731.

[15] **Karnok**, **D**. and Kemény, Z. Definition and handling of data types in a dataflow-oriented modelling and processing environment. In: Ilie-Zudor, E., Kemény, Z., and Monostori, L. (editors), *MITIP 2012. 14th International Conference on Modern information Technology in the Innovation Processes of the Industrial Enterprises*. Location: MTA SZTAKI, Budapest, Hungary, 24–26 October, 2012, pages 561–574. (ISBN: 978-963-311-373-8).

[16] **Karnok**, **D**. and Kemény, Z. Framework for building and coordinating information flows in logistics networks. In: Ilie-Zudor, E., Kemény, Z., and Monostori, L. (editors), *MITIP 2012. 14th International Conference on Modern information Technology in the Innovation Processes of the Industrial Enterprises*. Location: MTA SZTAKI, Budapest, Hungary, 24–26 October, 2012, pages 551–560. (ISBN: 978-963-311-373-8).

[17] **Karnok**, **D**. and Monostori, L. Logistics platform: developing a distributed, cooperative production and logistics system. In: Váradi, K. and Vörös, G. (editors), *Proceedings of the 6th Conference on Mechanical Engineering*. Location: Budapest University of Technology and Economics, Budapest, Hungary, 29–30 May, 2008, Paper G-2008-C-4. (ISBN: 978-963-420-947-8).

[18] **Karnok**, **D**. and Monostori, L. Novel approaches for better transparency in production and supply chains. In *42nd CIRP Conference on Manufacturing Systems, Sustainable Development of Manufacturing Systems*. Location: Grenoble, France, 3–6 June, 2009.

[19] van Blommestein, F., **Karnok**, **D**., and Kemény, Z. In: Lee, I. (editor), *RFID Technology Integration for Business Performance Improvement*, chapter Meta-data alignment in open Tracking & Tracing systems. IGI Global, July 2014, pages 121–139. (DOI: 10.4018/978-1-4666-6308-4.ch006), (ISBN: 978-1-4666-6308-4).

[20] Váncza, J., Egri, P., and **Karnok**, **D**. Planning in concert: a logistics platform for production networks. In: Maropoulos, P. and Newman, S. (editors), *Proceedings of the 4th International Conference on Digital Enterprise Technology (DET2007)*. Location: University of Bath, Bath, England, 19–21 September, 2007, pages 462–470. (ISBN: 978-0-86197-141-1).

Bibliográfia

[21] Albrecht, M. Supply chain coordination mechanisms: New approaches for collaborative planning, volume 628. Springer Science & Business Media, 2009.

[22] Arenas, M. and Libkin, L. A normal form for XML documents. In *Proceedings of the twenty-first ACM SIGMOD-SIGACT-SIGART symposium on Principles of database systems*, PODS '02. Location: Madison, Wisconsin, 2002, pages 85–96, New York, NY, USA. ACM. (DOI: 10.1145/543613.543625), (ISBN: 1-58113-507-6).

[23] Cachon, G. Supply chain coordination with contracts. In: de Kok, A. and Graves, S. (editors), *Supply chain management: Design, coordination, cooperation. Handbooks in OR and MS*, volume 11. Location: New York, USA, 2003, pages 229–339.

[24] Chambers, C., Raniwala, A., Perry, F., Adams, S., Henry, R. R., Bradshaw, R., and Weizenbaum, N. Flumejava: easy, efficient data-parallel pipelines. In *Proceedings of the 2010 ACM SIGPLAN conference on Programming language design and implementation*, PLDI '10. Location: Toronto, Ontario, Canada, 2010, pages 363–375, New York, NY, USA. ACM. (DOI: 10.1145/1806596.1806638), (ISBN: 978-1-4503-0019-3).

[25] Cho, S.-w. and Pak, M.-s. An integrative view on cyber threat to global supply chain management systems. *Journal of Korea Trade*, 2011, 15(3):55–87.

[26] Courtney, A. Frappé: Functional reactive programming in java. In *In Proceedings of Symposium on Practical Aspects of Declarative Languages. ACM*. pages 29–44, 2001. Springer-Verlag.

[27] Duta, A. C., Barker, K., and Alhajj, R. Ra: An XML schema reduction algorithm, 2006. (pdf)

[28] ECMA International. C# language specification. Standard ECMA-334, 4th Edition, 2006. (pdf)

[29] Egri, P. and Váncza, J. Incentives for cooperative planning in focal supply networks. In *IWES 2006. 6th international workshop on emergent synthesis. Tokyo, 2006.* Location: Tokyo, Japan, 2006, pages 17–24.

[30] Elliott, C. M. Push-pull functional reactive programming. In *Proceedings of the 2nd ACM SIGPLAN symposium on Haskell*, Haskell '09. Location:

Edinburgh, Scotland, 2009, pages 25–36, New York, NY, USA. ACM. (DOI: 10.1145/1596638.1596643), (ISBN: 978-1-60558-508-6).

[31] Fleischmann, B. and Meyr, H. Planning hierarchy, modelling and advanced planning systems. In: de Kok, A. and Graves, S. (editors), *Supply chain management: Design, coordination, cooperation. Handbooks in OR and MS*, volume 11. Location: New York, USA, 2003, pages 457–523.

[32] Hosoya, H., Frisch, A., and Castagna, G. Parametric polymorphism for XML. In *Proceedings of the 32nd ACM SIGPLAN-SIGACT symposium on Principles of programming languages*, POPL '05. Location: Long Beach, California, USA, 2005, pages 50–62, New York, NY, USA. ACM. (DOI: 10.1145/1040305.1040310), (ISBN: 1-58113-830-X).

[33] Hosoya, H. and Pierce, B. C. XDuce: A statically typed XML processing language. *ACM Trans. Internet Technol.*, May 2003, 3(2):117–148. (DOI: 10.1145/767193.767195).

[34] Igarashi, A. and Pierce, B. C. Foundations for virtual types. *Information and Computation*, 2002, 175(1):34 – 49. (DOI: 10.1006/inco.2001.2942). (link)

[35] Jeong, B., Lee, D., Cho, H., and Lee, J. A novel method for measuring semantic similarity for XML schema matching. *Expert Syst. Appl.*, April 2008, 34(3):1651–1658. (DOI: 10.1016/j.eswa.2007.01.025).

[36] Kaes, S. Type inference in the presence of overloading, subtyping and recursive types. In *Proceedings of the 1992 ACM conference on LISP and functional programming*, LFP '92. Location: San Francisco, California, United States, 1992, pages 193–204, New York, NY, USA. ACM. (DOI: 10.1145/141471.141540), (ISBN: 0-89791-481-3).

[37] Kagermann, H. and Wahlster, W. Recommendations for implementing the strategic initiative Industrie 4.0. acatech, National Academy of Science and Technology, Germany, 2013. (pdf)

[38] Kfoury, A. J., Tiuryn, J., and Urzyczyn, P. Type reconstruction in the presence of polymorphic recursion. *ACM Trans. Program. Lang. Syst.*, April 1993, 15(2):290–311. (DOI: 10.1145/169701.169687).

[39] Kim, K.-D. and Kumar, P. R. Cyber–physical systems: A perspective at the centennial. *Proceedings of the IEEE*, 2012, 100(Special Centennial Issue):1287–1308. (DOI: 10.1109/JPROC.2012.2189792).

[40] Li, X. and Wang, Q. Coordination mechanisms of supply chain systems. *European Journal of Operational Research*, 2007, 179(1):1–16. (DOI: 10.1016/j.ejor.2006.06.023).

[41] Liberty, J. and Betts, P. Programming reactive extensions and linq. Apress, 1st edition, 2011. (ISBN: 978-1-4302-3747-1).

[42] Madhavan, J., Bernstein, P. A., and Rahm, E. Generic schema matching with cupid. In *Proceedings of the 27th International Conference on Very Large Data Bases*, VLDB '01. pages 49–58, 2001, San Francisco, CA, USA. Morgan Kaufmann Publishers Inc. (ISBN: 1-55860-804-4).

[43] Maropoulos, P. G., Kotsialos, A., and Bramall, D. G. A theoretical framework for the integration of resource aware planning with logistics for the dynamic validation of aggregate plans within a production network. *CIRP Annals – Manufacturing Technology*, 2006, 55(1):483–488. ID number: ISI:000240073900112

[44] McGowan, M. K. *Electronic Data Interchange (EDI)*, pages 860–868. John Wiley & Sons, Inc., 2007, (DOI: 10.1002/9781118256107.ch55), (ISBN: 9781118256107).

[45] McKay, K. N. and Wiers, V. C. Integrated decision support for planning, scheduling, and dispatching tasks in a focused factory. *Computers in Industry*, 2003, 50(1):5–14. (DOI: 10.1016/S0166-3615(02)00146-X).

[46] Meijer, E. The world according to LINQ. *Queue*, August 2011, 9(8):60:60–60:72. (DOI: 10.1145/2016036.2024658).

[47] Meijer, E. and Shields, M. XMlambda - a functional language for constructing and manipulating XML documents. 2000. (pdf)

[48] Milner, R. A theory of type polymorphism in programming. *Journal of Computer and System Sciences*, 1978, 17:348–375.

[49] Monostori, L. Cyber-physical production systems: roots from manufacturing science and technology. *AT-AUTOMATISIERUNGSTECHNIK*, 2015, 63(10):766–776. (DOI: 10.1515/auto-2015-0066).

[50] Monostori, L., Kádár, B., Bauernhansl, T., Kondoh, S., Kumara, S., Reinhart, G., Sauer, O., Schuh, G., Sihn, W., and Ueda, K. Cyber-physical systems in manufacturing. *CIRP Annals – Manufacturing Technology*, 2016, 65(2):621–641. (DOI: 10.1016/j.cirp.2016.06.005).

[51] Monostori, L., Kis, T., Váncza, J., Kádár, B., and Erdős, G. Real-time, cooperative enterprises for customised mass production. *International Journal of Computer Integrated Manufacturing*, 2009, 22(1):55–68.

[52] National Instruments. Labview system design software. Official website, 2012. (link)

[53] OASIS / RELAX NG Technical Committeee. RELAX NG. Official reference page, 2012. (link)

[54] Odersky, M., Sulzmann, M., and Wehr, M. Type inference with constrained types. *Theory and practice of object systems*, 1999, 5(LAMP-ARTICLE-1999-001):35.

[55] Oracle Technology Network. Oracle Technology Network, Java developers' index page. Official website, 2012. (link)

[56] Palsberg, J. Efficient inference of object types. *Inf. Comput.*, December 1995, 123(2):198–209. (DOI: 10.1006/inco.1995.1168).

[57] Palsberg, J., Wand, M., and O'Keefe, P. Type inference with non-structural subtyping. *Formal Aspects of Computing*, 1997, 9:9–49.

[58] Palsberg, J. and Zhao, T. Type inference for record concatenation and subtyping, 2003. (pdf)

[59] Pochet, Y. and Wolsey, L. A. Production planning by mixed integer programming. Springer Science & Business Media, 2006.

[60] Pottier, F. A framework for type inference with subtyping. In *Proceedings of the third ACM SIGPLAN international conference on Functional programming*, ICFP '98. Location: Baltimore, Maryland, United States, 1998, pages 228–238, New York, NY, USA. ACM. (DOI: 10.1145/289423.289448), (ISBN: 1-58113-024-4).

[61] Pottier, F. Type Inference in the Presence of Subtyping: from Theory to Practice. INRIA, 1998. (pdf)

[62] Schuh, G., Kampker, A., Narr, C., Potente, T., and Attig, P. myOpenFactory. *International Journal of Computer Integrated Manufacturing*, 2008, 21(2):215–221. (DOI: 10.1080/09511920701607766).

[63] Schuh, G., Gottschalk, S., and Höhne, T. High resolution production management. *CIRP Annals – Manufacturing Technology*, 2007, 56(1):439–442. (DOI: 10.1016/j.cirp.2007.05.105).

[64] Simonet, V. and Rocquencourt, I. Type inference with structural subtyping: A faithful formalization of an efficient constraint solver, 2003. (pdf)

[65] Stadtler, H. Supply chain management and advanced planning – basics, overview and challenges. *European Journal of Operational Research*, 2005, 163(3):575–588. (DOI: 10.1016/j.ejor.2004.03.001).

[66] Toub, S. Introduction to tpl dataflow. Microsoft online document, 2011. (link, last accessed: April 2011)

[67] Traytel, D., Berghofer, S., and Nipkow, T. Extending hindley-milner type inference with coercive structural subtyping. In: Yang, H. (editor), *APLAS*, volume 7078 of *Lecture Notes in Computer Science*. pages 89–104, 2011. Springer. (ISBN: 978-3-642-25317-1).

[68] Váncza, J. and Egri, P. Coordinating supply networks in customized mass production: A contract-based approach. *CIRP Annals - Manufacturing Technology*, 2006, 55(1):489 – 492. (DOI: 10.1016/S0007-8506(07)60465-X).

[69] Váncza, J., Egri, P., and Monostori, L. A coordination mechanism for rolling horizon planning in supply networks. *CIRP Annals - Manufacturing Technology*, 2008, 57(1):455 – 458. (DOI: 10.1016/j.cirp.2008.03.105).

[70] World Wide Web Consortium. Document type definition (dtd). W3C recommendation reference page, 1999. (link)

[71] World Wide Web Consortium. XML path language (XPath) version 1.0. W3C recommendation reference page, 1999. (link)

[72] World Wide Web Consortium. XSL transformations (XSLT) version 1.0. W3C recommendation reference page, 1999. (link)

[73] World Wide Web Consortium. SOAP version 1.2. W3C recommendation reference page, 2007. (link)

[74] World Wide Web Consortium. Extensible Markup Language (XML) 1.0 (fifth edition). W3C recommendation reference page, 2008. (link)

[75] World Wide Web Consortium. XML schema version 1.1. W3C recommendation reference page, 2010. (link)

[76] World Wide Web Consortium. XQuery 1.0: An XML query language (second edition). W3C recommendation reference page, 2010. (link)

Adminisztratív információk

Publikációk száma: **20**
Első szerzős publikációk: **7**
Független hivatkozások (2017 Novemberéig): **39**
Teljes impakt faktor: **4.077**

Minimum követelmény	**Tényleges**	**Minimum**	**százalék**
Tézishez kapcsolódó publikációk	20	4	500%
Idegen nyelvő lektorált folyóirat cikkek	241%	200%	121%
– Web of Science cikkek	125%	50%	250%
Magyar nyelvű publikációk	116%	100%	116%
Konferencia vagy egyéb cikkek	11	1	1100%